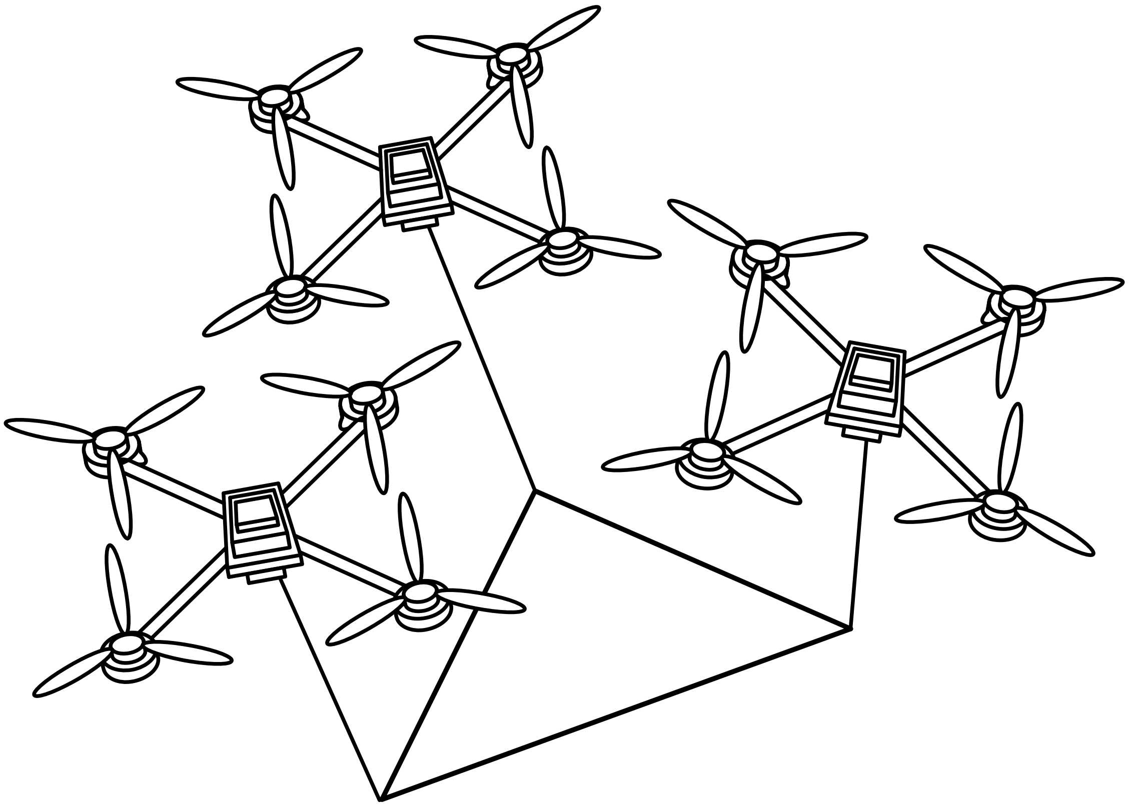
CHALMERS
UNIVERSITY OF TECHNOLOGY



Distributed Model Predictive Control for a Coordinated Multi-Agent System

Simulation and Implementation on a System of Autonomous Quadcopters with Path-Planning for Formation Flying and Collision Avoidance

Master's thesis in Systems, Control and Mechatronics

ALEXANDER WALDEJER
ASHKAN GHODRATI

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Master's thesis EX004/2018

Distributed Model Predictive Control for a Coordinated Multi-Agent System

Simulation and Implementation on a System of Autonomous Quadcopters with Path-Planning for Formation Flying and Collision Avoidance

ALEXANDER WALDEJER
ASHKAN GHODRATI

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2018

Distributed Model Predictive Control for a Coordinated Multi-Agent System
Simulation and Implementation on a System of Autonomous Quadcopters with Path-Planning for Formation Flying and Collision Avoidance
ALEXANDER WALDEJER, ASHKAN GHODRATI

Supervisor: Jonas Kurol, ÅF Technology, Embedded Systems West
Examiner: Balázs Adam Kulcsár, Electrical Engineering

Master's Thesis EX004/2018
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Illustration of a multi-agent system application of three quadcopter together lifting a payload.

Distributed Model Predictive Control for a Coordinated Multi Agent System
Simulation and Implementation on a System of Autonomous Quadcopters with
Path-Planning for Formation Flying and Collision Avoidance

ALEXANDER WALDEJER, ASHKAN GHODRATI
Department of Electrical Engineering
Chalmers University of Technology

Abstract

This thesis investigates how to use Model Predictive Control in a distributed fashion in order to achieve coordinated behaviour in a multi-agent system, where the proposed application is a system of multiple quadcopters which would together lift a hanging payload from one point to another. The coordinated task would translate to formation flying between the agents with the purpose of keeping a desired payload orientation.

The work involves physical modelling of the quadcopter and inverse kinematics of the payload. Further, a control design using a system of three decoupled Model Predictive Controllers in order to achieve reference tracking for altitude, position and attitude. An Extended Kalman Filter is used as an observer for output feedback and disturbance estimation. Model mismatch and external disturbances are compensated by means of feed-forward terms in parallel to each controller. Model Predictive Control input constraints are actively adapted to the feed forward terms. Sensor fusion algorithms have been used in order to estimate the orientation using an Inertial Measurement Unit. Path planning by means of formation flying is introduced to the position controller by augmenting the model as a solution to the coordinated task. As a benefit of using Model Predictive Control, the predicted formation states are shared between the agents in order to improve the tracking performance and compensate for communication delays. Further path planning algorithms are developed for both static obstacle and inter-vehicle collision avoidance. Both algorithms rely on a temporary change in target references to get around the obstacle. The inter-vehicle collision avoidance also relies on actively communicating the states between the agents in either a distributed or decentralized fashion.

The proposed methods for sharing predicted information in a distributed scheme has shown improvement in performance over a decentralized scheme. The performance is evaluated in terms of formation tracking and time of arrival at final target, with different communication delays. Stationary formation flying has been evaluated and introduced a chain reaction between the agents, where the distributed scheme dampened the oscillatory behaviour. The methods for collision avoidance have been demonstrated in simulation. The methods for a single quadcopter have been implemented using low cost hardware exploiting a computational inexpensive approximation of Model Predictive Control.

Keywords: Model Predictive Control, Distributed Control, Decentralized Control, Multi-Agent System, Path-Planning, Formation Flying, Collision Avoidance, Sensor Fusion, Extended Kalman Filter, Feed-Forward Disturbance Compensation.

Acknowledgements

We would first of all like to thank Associate Professor, Balázs Adam Kulcsár at Chalmers University of Technology, Electrical Engineering, for believing in our project. We knew that our project was ambitious and would not be able to complete it without his encouragement. He gave us the freedom needed in order to explore in depth and pursue in the directions we wanted to.

Further, we would like to thank our industrial supervisor, Jonas Kurol at ÅF Technology, Embedded Systems West, for supporting us. He and several of his engineers gave us advice and pushed us forward in the tougher parts of the work. ÅF Technology, Embedded Systems West, gave us also the support for investing in an experimental setup and office space where we could focus on our work.

Finally, we would like to thank our family and girlfriends, for keeping up with us and giving us motivation during the long days and nights. Not only for this final master thesis, but also for supporting us throughout the years of studying. This graduation could never have been completed without the help and support from them. Thank you.

Alexander Waldejer, Gothenburg, January 2018
Ashkan Ghodrati, Gothenburg, January 2018

Contents

List of Figures

List of Tables

Nomenclature

$\hat{\boldsymbol{x}}^p_{j,k\|k+1}$	Predicted states of agent j-th at time k for time $k+1$
$\boldsymbol{\tau}$	Quadcopter torque vector
$\boldsymbol{A}$	Motion model state space A matrix
$\boldsymbol{B}$	Motion model state space B matrix
$\boldsymbol{C}$	Motion model state space C matrix
$\boldsymbol{D}$	Motion model state space D matrix
$\boldsymbol{D}$	Quadcopter disturbance vector
$\boldsymbol{F}_D$	Quadcopter air friction vector
$\boldsymbol{M}^{xy}_{i,c}$	Circle-Circle i-th intersection point
$\boldsymbol{M}^{xy}_{i,l}$	Circle-Line i-th intersection point
$\boldsymbol{q}$	Quaternion vector of orientation
$\boldsymbol{T}$	Quadcopter total thrust vector
$\boldsymbol{u}_i$	Agent i-th input vector
$\boldsymbol{v}_i$	Agent or obstacle i-th velocity vector in XY plane
$\boldsymbol{x}_i$	Agent i-th state vector
$\boldsymbol{x}^m_i$	Agent i-th measured state vector
$\boldsymbol{x}_{i,cref}$	Agent i-th collision avoidance temporary references in XY plane
$\boldsymbol{x}_{i,fref}$	Agent i-th formation references in XY plane
$\boldsymbol{x}^{xy}_{i,obs}$	Obstacle i-th position in XY plane
$\boldsymbol{x}_{i,tref}$	Agent i-th target references in XY plane
$\boldsymbol{x}^{xy}_{j,k}$	Estimated states of agent j-th at time k in XY plane
$\boldsymbol{y}_k$	Full measurement vector in discrete time
$\boldsymbol{y}_{att,k}$	Decoupled attitude measurement vector in discrete time
$\boldsymbol{y}_{pos,k}$	Decoupled position measurement vector in discrete time
$\hat{\boldsymbol{x}}_i$	Agent i-th estimated state vector
$\hat{\boldsymbol{x}}_k$	Estimated full state vector in discrete time
$\hat{\boldsymbol{x}}_{att,k}$	Estimated decoupled attitude state vector in discrete time
$\hat{\boldsymbol{x}}_{pos,k}$	Estimated decoupled position state vector in discrete time
λ_i	Quadcopter i-th connector tension to payload
$\mathbb{R}$	The set of real numbers
ω_ϕ	Angular velocity around reference X axis
ω_ψ	Angular velocity around reference Z axis
ω_θ	Angular velocity around reference Y axis
ϕ	Roll, rotation around reference X axis
ψ	Yaw, rotation around reference Z axis
τ_ϕ	Quadcopter physical input torque around reference X axis
τ_ψ	Quadcopter physical input torque around reference Z axis

τ_θ	Quadcopter physical input torque around reference Y axis
θ	Pitch, rotation around reference Y axis
$\tilde{\boldsymbol{M}}_{i,c}^{xy}$	Circle-Circle i-th intersection candidate point
$\tilde{\boldsymbol{M}}_{i,l}^{xy}$	Circle-Line i-th intersection candidate point
$\tilde{d}_x$	Quadcopter disturbance including mass in world frame X axis
$\tilde{d}_y$	Quadcopter disturbance including mass in world frame Y axis
$\tilde{d}_z$	Quadcopter disturbance including gravitational force and mass in world frame Z axis
b	Quadcopter drag constant
c_m	Quadcopter motor constant
d_x	Quadcopter disturbance in world frame X axis
d_y	Quadcopter disturbance in world frame Y axis
d_z	Quadcopter disturbance in world frame Z axis
g	Gravitational force
I_{xx}	Quadcopter inertia around reference X axis
I_{yy}	Quadcopter inertia around reference Y axis
I_{zz}	Quadcopter inertia around reference Z axis
k	Quadcopter lift constant
k_d	Air friction
L	Quadcopter diagonal center to motor length [m]
L_f	Quadcopter diagonal motor to motor length [m]
m	Quadcopter mass
m_A	Quadcopter motor mass [kg]
m_C	Quadcopter centre arm mass [kg]
m_F	Quadcopter diagonal frame arm mass [kg]
r_A	Quadcopter single frame arm radius [m]
r_C	Quadcopter centre radius [m]
r_F	Quadcopter frame arm intersection radius [m]
r_{fref}	Formation flying reference distance, equal to A_i circle radius
T	MPC Prediction horizon
t_z	Discrete time delay
u_i	Quadcopter motor i-th electrical PWM input
v_x	Velocity in world X axis
v_y	Velocity in world Y axis
v_z	Velocity in world Z axis
x	Position in world X axis
y	Position in world Y axis
z	Position in world Z axis

Abbreviations

CA	Constant Acceleration
CCW	Counter Clock-Wise
CV	Constant Velocity
CW	Clock-Wise
DARE	Discrete Algebraic Ricatti Equation
EKF	Extended Kalman Filter
EMPC	Explicit Model Predictive Control
ESC	Electronic Speed Controller
FMPC	Fast Model Predictive Control
IVCA	Inter-Vehicle Collision Avoidance
KF	Kalman Filter
LLS	Linear Least Squares
LQR	Linear Quadratic Regulator
LTI	Linear Time-Invariant
LTV	Linear Time-Variant
MPC	Model Predictive Control
OCA	Obstacle Collision Avoidance
PID	Proportional–Integral–Derivative
PWM	Pulse Width Modulation
QP	Quadratic Program
RPM	Revolution Per Minute
RPi	Raspberry Pi
UAV	Unmanned Aerial Vehicle

1 Introduction

As computational power is increasing in embedded systems, implementation of *Model Predictive Control (MPC)* schemes have become more feasible on systems with faster dynamics. In traditional *Proportional–Integral–Derivative (PID)* control, the computed control policy does not take into account system dynamics and control actuator constraints. A lot of optimum solutions to control problems require the actuators to perform close to these constraints. Consequently saturation and more heuristic features such as anti-windup functionality are used, and saturation potentially results in nonlinear behaviour that generally are not desirable. However MPC is liberated from these approaches in that constraints are taken into account in a systematic way as part of the control design. It uses a mathematical model of the system at hand in order to predict an optimum trajectory and delivering a control law.

An instance of systems with fast dynamics are *Unmanned Aerial Vehicles (UAVs)*. The area of applications for UAVs are increasing every year where there has been a lot research and development with means of the quadcopter. The manoeuvrability and flexibility of the UAVs has made it popular for specific applications such as aerial photography, surveillance, search and rescue, and logistics such as package delivery. This thesis investigates the particular application of coordinated control of multiple quadcopters in order to perform a task together, where the given task is to lift an object together. Working on coordinated control of multiple quadcopters requires some considerations. It requires formation flying algorithms in order to control the payload position and orientation. UAVs flying close in formation or in an environment with obstacles, are in the risk of colliding, hence introduces the need of *Inter-Vehicle Collision Avoidance (IVCA)* and *Obstacle Collision Avoidance (OCA)* capabilities as safety.

1.1 Aims and Purpose

The aim of this thesis is to design a set of distributed MPC based controllers enabling multiple quadcopters to act in a coordinated fashion; autonomously lifting a payload with a desired attitude by keeping a formation between agents when travelling between a set of reference points. In order to do so, a mathematical quadcopter and payload model is needed to describe the system's overall behaviour. The focus of this thesis is to investigate how the MPC scheme can be used in a distributed fashion in order to optimize coordinated control between the quadcopters. A decentralized control strategy will be used for benchmarking the solution. The complete application will be implemented and verified in a simulated environment. In addition, the main control architecture will be implemented on a single quadcopter in order to evaluate the MPC based controller on a physical system based on low cost hardware. For this system, sensor fusion with respect to orientation estimation is necessary, in addition to state estimation and local indoor position measurement. The proposed methods are applicable to many multi agent systems, where the quadcopter application is on of them. A system of quadcopters with a set of multiple controllers where there are several states and limitations, is a big challenge that exercises advance features within control theory. With that mind, it is worth mentioning that some of the proposed algorithms are tailored for the problem at hand, hence reusability of them among any multi-agent system is not necessarily *plug-and-play*.

1.2 Present Research

Interesting areas of related research are many. [1] proposes how a set of distributed MPC based UAVs keep distance from each other by communicating predicted states which are added directly in to the quadratic cost function. The paper proposes a collision avoidance scheme based on state constraints and benchmarks the performance between a centralized and a distributed control strategy as the UAVs travel between reference two points. [2] introduces the idea of using a leader/follower principle in order to keep formation between three moving quadcopters. [3] presents different approaches on how multiple vehicles can perform leader/follower principle by sharing information such that current and future anticipated motion is taken in to the objective function to provide better stability. [4] describes a 2D model for two UAVs with the leader/follower principle. It uses an adaptive controller in order to improve robustness for unknown disturbances created by the UAVs. [5] talks about introducing dynamic priority based leader/follower principle in order to avoid chain reactions among the UAVs as they approach an obstacle. The idea is that not all of the UAVs need to manoeuvre just because another UAV performs collision avoidance. It also introduces the idea of using virtual leadership, which could be the payload centre point. A detailed 12 state nonlinear quadcopter model is given in [6]. [7] presents an MPC based quadcopter where the full 12 state nonlinear model has been decoupled, simplified and linearized. The payload can be illustrated as an inverted Stewart platform. However the Stewart platform is a rigid structure and does not represent the hanging payload [8]. [9] presents modelling and implementation of multiple cooperative quadcopters lifting a payload using stationary fixed point PID controllers. The approach is similar to the proposed solution of this thesis, however it differs in terms of applied control theory and reference point tracking. The presented model covers static equilibrium of the payload at a fixed position with the necessary wire tensions. The paper proposes an inverse kinematics solution where the robot positions are generated based on desired payload position and orientation, satisfying the constraints of static equilibrium and wire tension. [10] presents another application area where the inverse kinematics model of a hanging payload is used, namely a suspended load crane. The approach is similar to [9]. [11] presents a Lyapunov approach for feedback control of cable suspended robots in 2D, where static equilibrium of connection points and platform dynamics caused by mass, inertia and gravity is described. For the single quadcopter implementation, [12, 13, 14] present different approaches on how to perform sensor fusion, orientation and position estimation using an accelerometer, a gyroscope and a magnetometer, together with or without using an absolute measurement of position. Computational expensive solutions using the Kalman filter are given, however a more computational inexpensive solution to the orientation estimation based on a gradient descent method is presented in [15, 16]. For indoor positioning, a recursive trilateration algorithm is presented in [17], where the position estimation is based on four fixed anchor points in the room measuring the distance to a single mobile tag.

1.3 Scope of Work

The thesis includes physical modelling of the quadcopter dynamics with identification of system parameters, followed by modelling of the payload kinematics which includes estimation of the quadcopter payload connector tension. Further, a distributed MPC based control scheme is developed, that features functionality such as state and disturbance estimation, feed-forward disturbance compensation and actuator constraints which are varying given the disturbance feed-forward terms. Formation flying is introduced by means of path-planning dynamically depending on each quadcopter, satisfying a set of desired distances between each quadcopter. A collision avoidance algorithm is developed based on temporary target reference changes. Orientation estimation is achieved by fusing an accelerometer and gyroscope. The application is implemented in a simulation environment and evaluated between different control strategies for multiple flight scenarios. A real time implementation is developed in *C* language, using low cost hardware and a computational inexpensive *Fast Model Predictive Control (FMPC)* algorithm. The solution is also evaluated given this hardware.

1.4 Limitations

Limitations are set in order to compress the size of work. The following list gives a short description of limitations for this thesis.

- The thesis is not set to develop, nor benchmark the MPC optimization solver.
- The thesis does not include swarm robotics algorithms and where by means of communication, fusion of information is used for *Consensus-seeking* of collective behaviour.
- The thesis is not about Artificial Intelligence (AI) which is a higher level of strategic evaluation.
- The thesis is not related to the mechanical design aspect of a quadcopter, but rather focus on choosing low cost hardware for where the control scheme can be implemented.
- The sensor fusion algorithm is implemented and tuned to combine information from the gyroscope and accelerometer in order to achieve reliable state measurement, however the selected algorithm is not benchmarked among other

available algorithms,
- The focus is not on producing optimized lines of code or algorithms, rather than a working implementation.

1.5 Report Overview

The thesis starts of by presenting related theory in Chapter 2 in order to give the reader necessary background knowledge related to the major theories applied to this work. The methods are presented in Chapter 3 including physical modelling of both the quadcopter and the payload. The control design is given including model simplification, linearization, disturbance compensation, dynamic actuator and algorithms for path-planning of formation flying and collision avoidance. The state and disturbance observer is presented, followed by a sensor analysis, orientation estimation algorithm and verification of the implemented sensor fusion. Mechanical dampening of sensor noise is also given. Finally, model parameter identification is presented. All of the methods are implemented in simulation and on the real time system. Results and discussions are presented Chapter 4, where the multiple coordinated quadcopter scenario is evaluated for different test cases and benchmarked in simulation. On the real time system, the performance of a single flying quadcopter is evaluated given the low cost hardware used. Finally, a conclusion is made in Chapter 5, followed by proposed future work in Chapter 6.

2

Theory

This chapter presents necessary base theories applied in this thesis. The generic MPC problem is given, followed by the a brief overview of the computational less demaning FMPC algorithm. Next, both the Kalman Filter and Extended Kalman Filter are presented. Finally, definitions are given to distinguish between the centralized, decentralized and distributed control strategies.

2.1 Model Predictive Control

MPC is an optimization based regulator in which a problem of optimization is solved at each step to deliver an optimal control law. This noble approach takes into account actuators limits (minimum and maximum) and potentially limit on the rate of change (*the slew rate*) as well as process states constraints in delivering its solution. Consequently it liberates the need for any sort of saturation and all the problems arising with them. Given a model of the plant, it predicts what could be the states of the plant over some future sampling interval, taking into account the possibility of reaching the actuator constraints. There is a computation burden in solving such an optimization problem at every time instant problem. Given a precise model of the plant, it might not be necessary to calculate a new control law as frequent as with more conventional class of controllers such as PID. There is another approach to avoid the heavy computation load, namely *Explicit MPC (EMPC)*, in which the plant is divided into sub regions where the optimal control law is calculated and stored offline. Then, given the estimated states of the plant and parameters, the relevant control law calculated offline is applied online in real time. Such an approach can become challenging in terms of storage. EMPC is not not addressed in this thesis and interested readers can study it further in [18].
This section present the general formulation of MPC [18], followed by an algorithm for faster delivery of the solution to MPC, namely *Fast MPC* [19].

2.1.1 General Formulation

For a *Linear Time-Invariant (LTI)* MPC the *objective* or *cost function* can be expressed as

$$\begin{aligned} V_T(\boldsymbol{x}_k, \boldsymbol{u}_{k:k+T-1}) &= \boldsymbol{x}_T^T \boldsymbol{Q_f} \boldsymbol{x}_T + \sum_{i=k}^{k+T-1} (\boldsymbol{x}_i^T \boldsymbol{Q} \boldsymbol{x}_i + \boldsymbol{u}_i^T \boldsymbol{R} \boldsymbol{u}_i) \\ &\triangleq l_f(\boldsymbol{x}_T) + \sum_{i=k}^{k+T-1} l(\boldsymbol{x}_i, \boldsymbol{u}_i) \end{aligned} \tag{2.1}$$

where the objective V_T is quadratic in $\boldsymbol{x} \in \mathbb{R}^{n\times 1}$ and $\boldsymbol{u} \in \mathbb{R}^{m\times 1}$ with weightings of positive semi-definite $\boldsymbol{Q}$, $\boldsymbol{Q_f} \in \mathbb{R}^n$ and positive definite $\boldsymbol{R} \in \mathbb{R}^m$; all being symmetric matrices. $l(x, u)$ and l_f are referred to as *stage cost* and *final cost* respectively with $l_f(.) \geq 0$ and $l_f(0) = 0$. By rephrasing the control moves as $\mathcal{U} \triangleq u_{k:k+T-1}$, the optimization problem to be minimized from (2.1) can be expressed as

$$\begin{aligned} \min_{\mathcal{U}} \quad & V_T(\boldsymbol{x}_k, \mathcal{U}) \\ \text{subject to} \quad & \boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k, \quad \forall k \in [k, k+T-1] \\ & \boldsymbol{x}_k \in \mathbb{X}, \quad \boldsymbol{u}_k \in \mathbb{U}, \quad \forall k \in [k, k+T-1] \\ & \boldsymbol{x}_T \in \mathbb{X}_f \subseteq \mathbb{X} \end{aligned} \tag{2.2}$$

The sets of $\mathbb{X}$ and $\mathbb{U}$ define the allowed spaces for states and control moves respectively; in addition to $\mathbb{X}_f$ for the allowed space for final state with respect to the physical constraints. These sets are commonly treated as inequality constraint boxes expressed as $\boldsymbol{x}_{k,min} \leq \boldsymbol{x}_k \leq \boldsymbol{x}_{k,max}$. $\boldsymbol{A}$ and $\boldsymbol{B}$ is the state-space representation of the *Motion model* and the *Manipulated variable model* respectively. Given a linear model, a set of convex constraints and the quadratic cost, this optimization problem is referred to as a *Quadratic Program (QP)*. It is worth mentioning that when $i = k$ in (2.1), the objective $\boldsymbol{x}_k^T \boldsymbol{Q} \boldsymbol{x}_k$ is a constant as it is not possible or *too late* to affect it with optmization variables $\mathcal{U}$ and hence it is redundant, but it is kept for notation convenience purposes. As the solution to (2.2) is delivered to be the set $\boldsymbol{u}_{k:k+T-1}$, only the first solution $\boldsymbol{u}_k$ for prediction, is applied as the control move for the *current* time k and the rest is discarded.

2.1.2 Fast Model Predictive Control

Fast Model Predictive Control (FMPC) is a computational less expensive approach which delivers a sub-optimal optimization solution to (2.2) that is *on order of 100 times faster than a method that uses a generic optimizer* [19]. Without going into details, a brief overview of the collection of methods suggested by [19] is presented. The reader is referred to the citation for a more detailed and extensive presentation. A compact form of the QP can be described by

$$\begin{aligned} \min_{\boldsymbol{z}} \quad & \boldsymbol{z}^T \boldsymbol{H} \boldsymbol{z} + \boldsymbol{g}^T \boldsymbol{z} \\ s.t. \quad & \boldsymbol{C}\boldsymbol{z} = \boldsymbol{b}, \\ & \boldsymbol{P}\boldsymbol{z} \leq \boldsymbol{h} \end{aligned} \tag{2.3}$$

with optimization variable of $\boldsymbol{z} \in \mathbb{R}^{n_z \times 1}$ and for some matrices of $\boldsymbol{H} \in \mathbb{R}^{n_z}, \boldsymbol{g} \in \mathbb{R}^{n_z \times 1}$ which are to fit the weightings from (2.1), $\boldsymbol{C} \in \mathbb{R}^{n \times n_z}, \boldsymbol{b} \in \mathbb{R}^{n \times 1}$ to fit the motion model from (2.2) and finally $\boldsymbol{P} \in \mathbb{R}^{l \times n_z}, \boldsymbol{h} \in \mathbb{R}^{l \times 1}$ to fit the inequality constraints of (2.2).
Using an *Infesible Start Primal Barrier Method* the inequality constraints can be replaced with a barrier term in the objective resulting in

$$\begin{aligned} \min_z \quad & z^T H z + g^T z + \kappa \phi(z) \\ s.t. \quad & Cz = b \end{aligned} \tag{2.4}$$

where $\kappa > 0$ is a barrier parameter, and ϕ is the log barrier associated with the inequality constraints given by

$$\phi(z) = \sum_{i=1}^{l} -\log(h_i - p_i^T z) \tag{2.5}$$

where $p_1^T, ..., p_l^T$ are the rows of $\boldsymbol{P}$ and if $\boldsymbol{Pz} \nless \boldsymbol{h}$, then $\phi(z) = \infty$. As κ approaches zero, the solution of (2.4) approaches to (2.3). The problem (2.4) is a convex optimization problem with smooth objective and linear equality constraints, and is solved by Newton's method. The solution of (2.4) is no more than κl suboptimal to (2.3).
By introducing some variations into the infeasible start primal barrier method, the problem of (2.4) can be computed *much faster* than the original problem and *no significant decrease in the quality of the control law (as measurerd by the objective).* Two major variations that are used in the software package of the FMPC algorithm, are the tuning parameters κ and K^{max}. Instead of using a decreasing sequence of values of κ, it is proposed to use a fixed value. This has several advantages among which the number of Newton's steps required are reduced, hence speeding up the approximation. In conventional Newton's method, iteration of Newton's steps are stopped when a limit of iterations, K^{max}, are reached. In a real-time implementation with hard constraints, one might need to settle with the delivered solution in a worst-case time which is K^{max} *times the time per Newton step.*

2.2 Observers

This section presents the theory behind the *Kalman Filter (KF)* which is used for filtering of the raw position estimations provided by the positioning system and the theory behind the *Extended Kalman Filter (EKF)* used for state and disturbance estimation within the control scheme.

2.2.1 Kalman Filter

The Kalman filter is a closed loop linear Gaussian filtering model based on Bayesian filtering equations, including both motion and measurement state space models [20]. The filter proceeds by performing state and certainty predictions based on previous posterior results, before updating the prediction with measurement data. The prediction step is given by

$$\begin{aligned} \hat{\boldsymbol{x}}_{k|k-1} &= \boldsymbol{A}_{k-1}\hat{\boldsymbol{x}}_{k-1|k-1} \\ \hat{\boldsymbol{P}}_{k|k-1} &= \boldsymbol{A}_{k-1}\hat{\boldsymbol{P}}_{k-1|k-1}\boldsymbol{A}_{k-1}^T + \boldsymbol{Q}_{k-1} \end{aligned} \tag{2.6}$$

where the priors $\hat{\boldsymbol{x}}_{k|k-1}$ and $\hat{\boldsymbol{P}}_{k|k-1}$ are the predicted states vector and certainty matrix. The motion models $\boldsymbol{A}_{k-1}$ are either LTI or *Linear Time-Variant (LTV)*. Should the parameters of the model stay the same $A_k = A_{k-1}, \forall k$, then it is an LTI and it is not necessary to specify it with a time index. The update step is given by

$$\begin{aligned} \boldsymbol{v}_k &= \boldsymbol{y}_k - \boldsymbol{H}_k\hat{\boldsymbol{x}}_{k|k-1} \\ \boldsymbol{S}_k &= \boldsymbol{H}_k\hat{\boldsymbol{P}}_{k|k-1}\boldsymbol{H}_k^T + \boldsymbol{R}_k \\ \boldsymbol{K}_k &= \hat{\boldsymbol{P}}_{k|k-1}\boldsymbol{H}_k^T\boldsymbol{S}_k^{-1} \\ \hat{\boldsymbol{x}}_{k|k} &= \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k\boldsymbol{v}_k \\ \hat{\boldsymbol{P}}_{k|k} &= \hat{\boldsymbol{P}}_{k|k-1} - \boldsymbol{K}_k\boldsymbol{S}_k\boldsymbol{K}_k^T \end{aligned} \tag{2.7}$$

where the posterior state vector and certainty matrix $\hat{\boldsymbol{x}}_{k|k}$ and $\hat{\boldsymbol{P}}_{k|k}$ are computed using the innovation vector $\boldsymbol{v}_k$ and its certainty matrix $\boldsymbol{S}_k$, and the Kalman gain $\boldsymbol{K}_k$. In addition, $\boldsymbol{H}_k$ is the measurement model which governs the equations between some measurements and the states. Similarly to the motion model, this measurement model can also be either LTI or LTV. The Kalman filter is an optimal filter as the certainty matrices converge.

2.2.2 Extended Kalman Filter

The Extended Kalman filter is the nonlinear version of the KF, where the filter distribution is based on a Gaussian approximation using the Taylor series [20]. The base principle of computing the prior and posterior is similar to the KF, however the prediction and/or the measurement models are nonlinear, and the state and/or innovation certainty matrices are based on Jacobian matrices computed from the nonlinear models linearized at every sampling instance. The prediction step is given by

$$\begin{aligned}\hat{\boldsymbol{x}}_{k|k-1} &= \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1|k-1}) \\ \hat{\boldsymbol{P}}_{k|k-1} &= \boldsymbol{F}(\hat{\boldsymbol{x}}_{k-1|k-1})\hat{\boldsymbol{P}}_{k-1|k-1}\boldsymbol{F}(\hat{\boldsymbol{x}}_{k-1|k-1})^T + \boldsymbol{Q}_{k-1}\end{aligned} \tag{2.8}$$

where the prior state vector $\hat{\boldsymbol{x}}_{k|k-1}$ is based on the the nonlinear prediction model $\boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1|k-1})$ and where the prior certainty matrix $\hat{\boldsymbol{P}}_{k|k-1}$ is based on the Jacobian matrix $\boldsymbol{F}(\hat{\boldsymbol{x}}_{k-1|k-1})$ linearized around the previous posterior $\hat{\boldsymbol{x}}_{k-1|k-1}$. The update step is given by

$$\begin{aligned}\boldsymbol{v}_k &= \boldsymbol{y}_k - \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}) \\ \boldsymbol{S}_k &= \boldsymbol{H}(\hat{\boldsymbol{x}}_{k|k-1})\hat{\boldsymbol{P}}_{k|k-1}\boldsymbol{H}(\hat{\boldsymbol{x}}_{k|k-1})^T + \boldsymbol{R}_k \\ \boldsymbol{K}_k &= \hat{\boldsymbol{P}}_{k|k-1}\boldsymbol{H}(\hat{\boldsymbol{x}}_{k|k-1})^T\boldsymbol{S}_k^{-1} \\ \hat{\boldsymbol{x}}_{k|k} &= \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k\boldsymbol{v}_k \\ \hat{\boldsymbol{P}}_{k|k} &= \hat{\boldsymbol{P}}_{k|k-1} - \boldsymbol{K}_k\boldsymbol{S}_k\boldsymbol{K}_k^T\end{aligned} \tag{2.9}$$

where the posterior state vector and certainty matrix $\hat{\boldsymbol{x}}_{k|k}$ and $\hat{\boldsymbol{P}}_{k|k}$ are again, computed using the innovation vector and certainty matrix $\boldsymbol{v}_k$ and $\boldsymbol{S}_k$, and the Kalman gain $\boldsymbol{K}_k$. $\boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1})$ is the nonlinear measurement model and $\boldsymbol{H}(\hat{\boldsymbol{x}}_{k|k-1})$ is the Jacobian matrix linearized around the prior $\hat{\boldsymbol{x}}_{k|k-1}$. The EKF is not an optimal filter as it does not converge in terms of the certainty matrix.
The filter is expanded with a term related to the innovation $\boldsymbol{v}_k$. If measurements are unavailable during the update step, the innovation is set $\boldsymbol{v}_k = 0$ such that the state estimation is only based on the prediction using prior state vector $\boldsymbol{x}_{k|k-1}$. The conditional innovation term is given by

$$\boldsymbol{v}_k = \begin{cases}\boldsymbol{y}_k - \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}), & \text{if } \boldsymbol{y}_k \text{ available.} \\ 0, & \text{otherwise.}\end{cases} \tag{2.10}$$

2.3 Control Strategies

This section presents different control strategies on how to approach a multi-agent system with respect to computing the controller locally or centrally and in either cases what sources of information are to be communicated between agents [1]. The principles of centralized, decentralized and distributed control are given, where the two latter are implemented throughout simulation.

2.3.1 Centralized Control

For centralized control, the measured state information $\boldsymbol{x}_i^m$ from multiple agents $i = \{1...\infty\}$, are collected centrally in one controller. The control law of each individual agent $\boldsymbol{u}_i$ is communicated back as illustrated in Figure 2.1. Computation can take place on either of the agents or remotely on a stand-alone computer. This is optimal solution in the sense that it has collected the *current* states of all agents before calculating the control law.

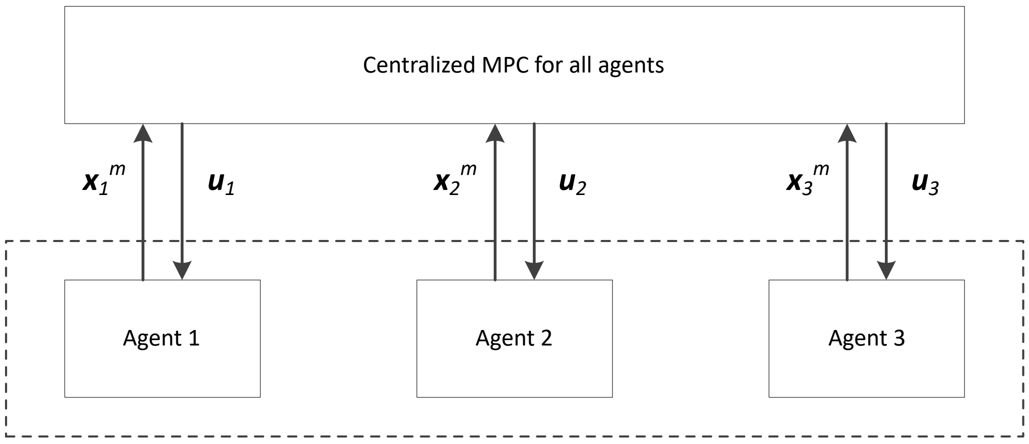


Figure 2.1: Centerlized control architecture

2.3.2 Decentralized Control

For decentralized control, calculation of the control law is performed on each of the agents locally using its own measured state $\boldsymbol{x}_i^m$, as illustrated in Figure 2.2. Sharing information between the agents can be done through communication between the agents, a sensory system on each individual agent which measures and builds up information about the other agents locally, or by using a remote sensory system that centrally communicates the information of all agents to each individual. It is worth mentioning that the control law of each individual is not shared, hence the agents do not know what the other agents are going to do with respect to actuation moves.

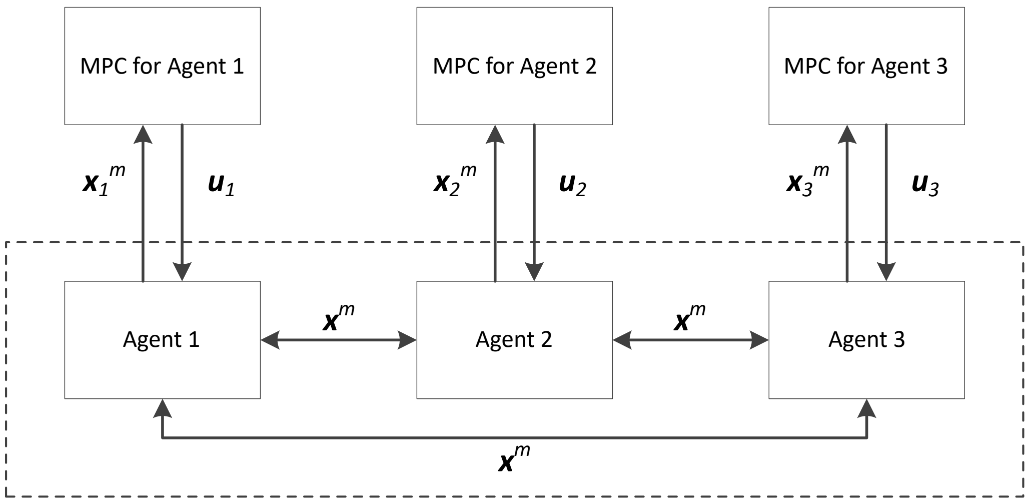


Figure 2.2: Deceneralized control architecture

2.3.3 Distributed Control

For distributed control, the control law is computed locally on each agent similar to the decentralized control strategy. However, the architecture is improved by also sharing the controller information. The MPC predictions $\boldsymbol{x}^p$ and $\boldsymbol{u}^p$ of the states and control moves are shared to acquire better performance as illustrated in Figure 2.3. Since the control law is computed locally, this requires a mean of communication between the agents to share the available information.

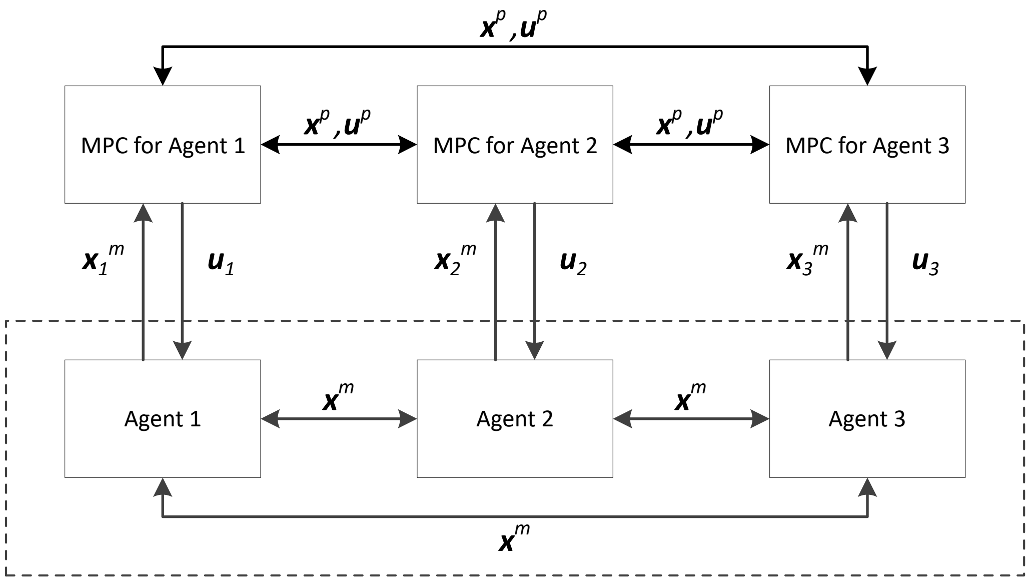


Figure 2.3: Distributed control architecture

As a comparison, since centeralized control can take place at the same node, the control action is optimum given information from the multi-agent system, potentially the latest information. This is in contrast to other two architectures where either the information regarding the control moves is absent or more extensive information regarding states and control moves are subject to noticeable delays in lines of communication. As a comparison between decenteralized and distributed control, one relies on existence of a communication line between the agents in distributed control, while decenteralized control can potentially be exploited by using of a sensory system remote from the agents liberating the demand for a line of communication.

3
Methods

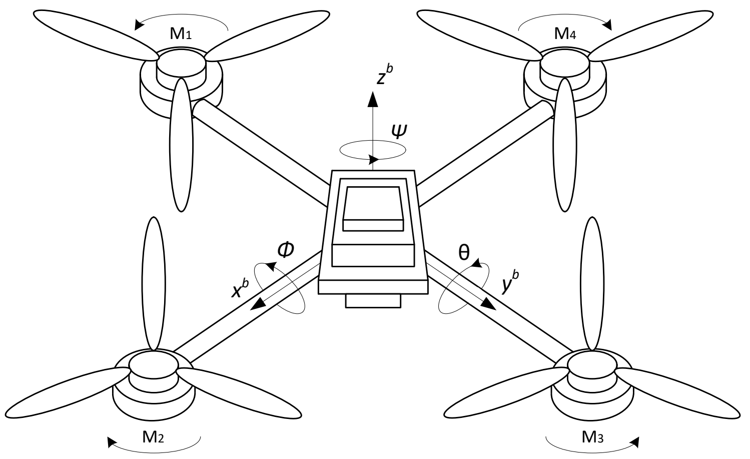
This chapter presents all proposed methods developed for this thesis. It covers quadcopter, payload and disturbance modelling. The proposed control design includes model simplification, linearization and discretization, with disturbance compensation, time-variant input constraints and model analysis with respect to sampling time. Path-planning algorithms are presented for formation flying, collision avoidance and reference ramping. Observer design is given in order to estimate system states and disturbances, followed by a sensor analysis of the hardware used, including sensor fusion, orientation estimation and verification of the implementation. Next, the quadcopter model parameters are identified and presented. Finally, an overview is given of the real time system hardware setup and developed software.

3.1 Model

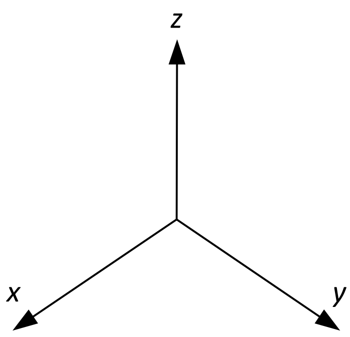
This section covers physical modelling of the quadcopter and the payload. The payload model is used to set the agent formation in the XYZ plane. Further, it returns a disturbance estimate of how much the connector tension between the quadcopter and payload affects the XYZ position.

3.1.1 Quadcopter

The quadcopter design is presented in Figure 3.1. The model is inspired by [6] and [7]. The model is given by 12 states, where the first six states describe the translation of the reference frame in the world frame, while the last six states describes the attitude of the quadcopter between the world frame and the reference frame. System inputs are based on four identical motors M_1-M_4, all facing the same direction, where each pair of diagonal motors spin *Clock-Wise (CW)* and *Counter Clock-Wise (CCW)*. Different combinations of total thrust and diagonal pair wise difference of thrust between the individual motors, creates the translational motion and changes in attitude.



(a) Illustration of quadcopter design



(b) World frame

Figure 3.1: Quadcopter setup with reference and world frame axes x^b, y^b, z^b and x, y, z respectively.

Table 3.1 illustrates how roll, pitch and yaw motion is achieved by creating the torques τ_ϕ, τ_θ and τ_ψ using M_1-M_4.

Table 3.1: Quadcopter roll, pitch and yaw motion summarized.

Angles	Torques	Motors
Positive ϕ (roll)	Positive τ_ϕ	M_3 increase, M_1 decrease, $M_2 = M_4$
Negative ϕ (roll)	Negative τ_ϕ	M_1 increase, M_3 decrease, $M_2 = M_4$
Positive θ (pitch)	Positive τ_θ	M_4 increase, M_2 decrease, $M_1 = M_3$
Negative θ (pitch)	Negative τ_θ	M_2 increase, M_4 decrease, $M_1 = M_3$
Positive ψ (yaw)	Positive τ_ψ	M_1 and M_3 increase, M_2 and M_4 decrease
Negative ψ (yaw)	Negative τ_ψ	M_2 and M_4 increase, M_1 and M_3 decrease
Balanced	Zero	All equal, $M_1 = M_2 = M_3 = M_4$

Table 3.2 illustrates how the translational motion in x, y and z axes is achieved depending on the angles ϕ, θ and ψ.

Table 3.2: Quadcopter translation motion summarized.

Translation	Angles and Input
Positive x	Positive θ with input thrust
Negative x	Negative θ with input thrust
Positive y	Negative ϕ with input thrust
Negative y	Positive ϕ with input thrust
Positive z	Total thrust larger than gravitational force
Negative z	Total thrust less than gravitational force
x and y	Positive and/or negative combination of θ and ϕ with either $\psi = 0$ or $\psi \neq 0$

The model is divided into three parts describing the translation, attitude and actuator dynamics. The equation of translational motion for positions x, y and z in the world frame is given by

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \boldsymbol{R_{\phi\theta\psi}} \boldsymbol{T} + \boldsymbol{F_D} + \boldsymbol{D} \tag{3.1}$$

where m is the system mass, g is gravitational acceleration, $\boldsymbol{R_{\phi\theta\psi}} \in \mathbb{R}^{3\times3}$ is the rotational matrix between the reference and world frame as a function of the angles ϕ, θ and ψ, $\boldsymbol{T}$ is the total thrust vector created by the actuators, $\boldsymbol{F_D}$ is the air friction vector and $\boldsymbol{D}$ is a disturbance vector representing payload tension. The rotational matrix is given by

$$\boldsymbol{R_{\phi\theta\psi}} = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \cos\phi\sin\psi & \sin\psi\sin\phi + \cos\psi\cos\phi\sin\theta \\ \cos\theta\sin\psi & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & \cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{3.2}$$

The air friction and disturbance vectors are given by

$$\boldsymbol{F_D} = -k_d \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \qquad \boldsymbol{D} = \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} \tag{3.3}$$

where k_d is the air friction coefficient, and d_x, d_y and d_z represents disturbances exciting the quadcopter in the inertial frame. The angular velocity dynamics described in the reference frame x^b, y^b and z^b come from the Euler's equations of rigid body motion given by

$$\boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\boldsymbol{I}\boldsymbol{\omega}) = \boldsymbol{\tau} \tag{3.4}$$

where $\boldsymbol{I} \in \mathbb{R}^{3\times3}$ is the diagonal moment of inertia matrix, the $\boldsymbol{\omega} = [\omega_\phi \; \omega_\theta \; \omega_\psi]^T$ vector represents angular velocity around the reference axes, and the $\boldsymbol{\tau} = [\tau_\phi \; \tau_\theta \; \tau_\psi]^T$ vector represents the available input torques. Expanding (3.4) leads to

$$\begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{\omega}_\phi \\ \dot{\omega}_\theta \\ \dot{\omega}_\psi \end{bmatrix} = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} - \begin{bmatrix} (I_{yy} - I_{zz})\omega_\theta\omega_\psi \\ (I_{zz} - I_{xx})\omega_\phi\omega_\psi \\ (I_{xx} - I_{yy})\omega_\phi\omega_\theta \end{bmatrix} \tag{3.5}$$

The attitude relation between the reference frame and the inertia frame is described by the angles ϕ, θ and ψ given by

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} \begin{bmatrix} \omega_\phi \\ \omega_\theta \\ \omega_\psi \end{bmatrix} \tag{3.6}$$

The actuator dynamics given by the thrust vector $\boldsymbol{T}$ from (3.1) is given by

$$\boldsymbol{T} = \sum_{i=1}^{4} f_i \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = k\sum_{i=1}^{4} \omega_i^2 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = k\sum_{i=1}^{4} c_m u_i^2 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.7}$$

where f_i, ω_i and u_i is the thrust, angular velocity and applied voltage of motors $i = \{1...4\}$. The angular velocity is typically related to the motor voltage by means of the motor constant c_m. On the real time system, the motors are directly controlled via a 0-100% *Pulse Width Modulation (PWM)* analogue signal passed on to a set of *Electronic Speed Controllers (ESCs)* which converts the control signal to a suitable voltage level. Therefore, in this the motor constant c_m represents a conversion between the PWM signal and the motor angular velocity. Further, the relation between motor angular velocity and actual thrust is given by the lift constant k, which translates motor angular velocity to thrust. The torque vector $\boldsymbol{\tau}$ from (3.4) is expanded to

$$\begin{aligned} \tau_\phi &= L(-f_1 + f_3) = Lk(-\omega_1^2 + \omega_3^2) = Lkc_m(-u_1^2 + u_3^2) \\ \tau_\theta &= L(-f_2 + f_4) = Lk(-\omega_2^2 + \omega_4^2) = Lkc_m(-u_2^2 + u_4^2) \\ \tau_\psi &= b(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) = bc_m(u_1^2 - u_2^2 + u_3^2 - u_4^2) \end{aligned} \tag{3.8}$$

where L is the distance between the motors and the reference frame origin, and b is the drag constant relating the motor angular velocity to torque around the z^b axis. Individual thrust of each motor is expressed as $f_i,\ i = \{1...4\}$. The relation between the PWM inputs $u_i,\ i = \{1...4\}$, thrust $\boldsymbol{T}$ and torques $\boldsymbol{\tau}$ is expressed in a compact form given by

$$\begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} kc_m & kc_m & kc_m & kc_m \\ -Lkc_m & 0 & Lkc_m & 0 \\ 0 & -Lkc_m & 0 & Lkc_m \\ bc_m & -bc_m & bc_m & -bc_m \end{bmatrix} \begin{bmatrix} u_1^2 \\ u_2^2 \\ u_3^2 \\ u_4^2 \end{bmatrix} \tag{3.9}$$

where the solution for each PWM input u_i is given by

$$\begin{aligned} u_1 &= \sqrt{(TLb - 2b\tau_\phi + Lk\tau_\psi)/(4Lbc_mk)} \\ u_2 &= \sqrt{(TLb - 2b\tau_\theta - Lk\tau_\psi)/(4Lbc_mk)} \\ u_3 &= \sqrt{(TLb + 2b\tau_\phi + Lk\tau_\psi)/(4Lbc_mk)} \\ u_4 &= \sqrt{(TLb + 2b\tau_\theta - Lk\tau_\psi)/(4Lbc_mk)} \end{aligned} \tag{3.10}$$

The complete model results in 12 nonlinear state equations given by

$$\dot{x} = v_x \tag{3.11}$$
$$\dot{y} = v_y \tag{3.12}$$
$$\dot{z} = v_z \tag{3.13}$$
$$\dot{v}_x = -\frac{k_d}{m}v_x + \frac{k\,c_m}{m}(\sin\psi\sin\phi + \cos\psi\cos\phi\sin\theta)(u_1^2+u_2^2+u_3^2+u_4^2) + \frac{d_x}{m} \tag{3.14}$$
$$\dot{v}_y = -\frac{k_d}{m}v_y + \frac{k\,c_m}{m}(\cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi)(u_1^2+u_2^2+u_3^2+u_4^2) + \frac{d_y}{m} \tag{3.15}$$
$$\dot{v}_z = -\frac{k_d}{m}v_z + \frac{k\,c_m}{m}(\cos\theta\cos\phi)(u_1^2+u_2^2+u_3^2+u_4^2) - g + \frac{d_z}{m} \tag{3.16}$$
$$\dot{\phi} = \omega_\phi + \omega_\theta(\sin\phi\tan\theta) + \omega_\psi(\cos\phi\tan\theta) \tag{3.17}$$
$$\dot{\theta} = \omega_\theta\cos\phi - \omega_\psi\sin\phi \tag{3.18}$$
$$\dot{\psi} = \frac{\sin\phi}{\cos\theta}\omega_\theta + \frac{\cos\phi}{\cos\theta}\omega_\psi \tag{3.19}$$
$$\dot{\omega}_\phi = \frac{L\,k\,c_m}{I_{xx}}(-u_1^2+u_3^2) - (\frac{I_{yy}-I_{zz}}{I_{xx}})\omega_\theta\omega_\psi \tag{3.20}$$
$$\dot{\omega}_\theta = \frac{L\,k\,c_m}{I_{yy}}(-u_2^2+u_4^2) - (\frac{I_{zz}-I_{xx}}{I_{yy}})\omega_\phi\omega_\psi \tag{3.21}$$
$$\dot{\omega}_\psi = \frac{b\,c_m}{I_{zz}}(u_1^2-u_2^2+u_3^2-u_4^2) - (\frac{I_{xx}-I_{yy}}{I_{zz}})\omega_\phi\omega_\theta \tag{3.22}$$

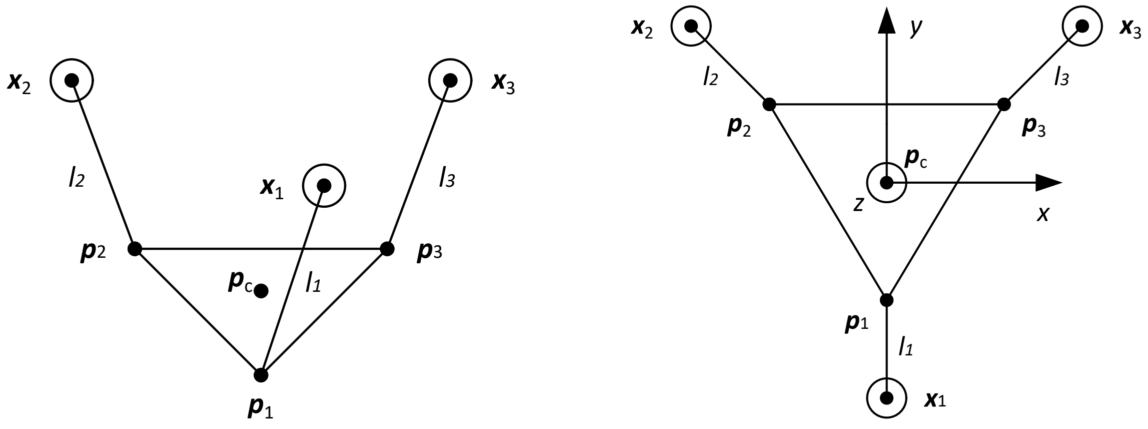
with the complete state and input vectors given by

$$\boldsymbol{x} = [x \;\; y \;\; z \;\; v_x \;\; v_y \;\; v_z \;\; \phi \;\; \theta \;\; \psi \;\; \omega_\phi \;\; \omega_\theta \;\; \omega_\psi]^T \qquad \boldsymbol{u} = [T \;\; \tau_\phi \;\; \tau_\theta \;\; \tau_\psi]^T \tag{3.23}$$

Note that $\boldsymbol{u}$ describes the inputs as thrust and torques, which are the physical inputs to the quadcopter. The electrical inputs u_i represented by PWM are all nonlinear functions of thrust and torques as given in (3.57). Working with the physical inputs is more practical with respect to avoiding the nonlinear terms and linearization.

3.1.2 Payload

The payload model is based on a vertically hanging load which is physically attached to the three agents. The payload dynamics and modelling are based on [9], with inspiration of [10]. The main steps are introduced here, but for the full model derivation, see [9]. By using inverse kinematics, the necessary position of each agent is calculated given a desired payload position and orientation in the XYZ plane. Based on this the agents formation can be determined. Figure 3.2 illustrates the principle of a hanging equilateral triangle shaped load, with agent coordinates $\boldsymbol{x}_i = [x\ y\ z]^T$ attached to connection points $\boldsymbol{p}_i = [x\ y\ z]^T$ via the connector l_i, where $i = \{1...3\}$. In general, static payload position is a result of static equilibrium achieved when the line tensions cancel each other out such that the total forces in x and y directions are zero and force z direction counteracts gravitational disturbance. The outcome from the following model will give a set of quadcopter positions in the world frame which give a desired static payload position and orientation.



(a) Payload from side **(b)** Payload from above

Figure 3.2: Hanging payload with quadcopter connection points P_1, P_2 and P_3.

The dynamics contain orientation and translation of the platform frame with respect to the world frame, and connector tensions with respect to static equilibrium of a certain given initial position and orientation. By specifying initial payload coordinates $\boldsymbol{p}_i$, a desired payload translation $\boldsymbol{T} = [x\ y\ z]^T$ with respect to the payload centre point $\boldsymbol{p}_c = [x\ y\ z]^T$ and a desired orientation $O = \{\theta, \phi, \psi\}$, new platform coordinate points $\tilde{\boldsymbol{p}}_i$ given by

$$\tilde{\boldsymbol{p}}_i = \boldsymbol{T} + \boldsymbol{R}_{\psi\theta\phi}\boldsymbol{p}_i \tag{3.24}$$

where $\mathbf{R}_{\psi\theta\phi}$ is a ZYX rotational matrix given by

$$\boldsymbol{R}_{\theta\phi\psi} = \begin{bmatrix} \cos_\theta \cos_\phi \cos_\psi - \sin_\theta \sin_\psi & -\cos_\psi \sin_\theta - \cos_\theta \cos_\phi \sin_\psi & \cos_\theta \sin_\phi \\ \cos_\theta \sin_\psi + \cos_\phi \cos_\psi \sin_\theta & \cos_\theta \cos_\psi - \cos_\phi \sin_\theta \sin_\psi & \sin_\theta \sin_\phi \\ -\cos_\psi \sin_\phi & \sin_\phi \sin_\psi & \cos_\phi \end{bmatrix} \tag{3.25}$$

Equilibrium is given by the total non zero pitch wrench created by the connector tension equalized by the gravity wrench. Non zero pitch wrench defines zero rotational motions on the connectors, hence it has only straight line motion. The two wrenches $\boldsymbol{\omega}_i$ and $\boldsymbol{g}$ are given by

$$\boldsymbol{w}_i = \frac{1}{l_i}\begin{bmatrix} \boldsymbol{x}_i - \tilde{\boldsymbol{p}}_i \\ \tilde{\boldsymbol{p}}_i \times \boldsymbol{x}_i \end{bmatrix} \qquad \boldsymbol{g} = -mg\begin{bmatrix} \boldsymbol{R}^T_{\theta\phi\psi} e_3 \\ \boldsymbol{p}_c \times \boldsymbol{R}^T_{\theta\phi\psi} e_3 \end{bmatrix} \tag{3.26}$$

where the gravitational force g is represented in x, y and z directions through a rotation and the unit vector $e_3 = [0\ 0\ 1]^T$. By arranging the wrench equations in vector form and introducing the connector tension $\lambda_i \geq 0$, the static equilibrium is given by

$$\begin{bmatrix} \boldsymbol{w}_1 & \boldsymbol{w}_2 & \boldsymbol{w}_3 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = -\boldsymbol{g} \quad \rightarrow \quad \boldsymbol{W}\boldsymbol{\lambda} = -\boldsymbol{g} \tag{3.27}$$

The static equilibrium equation can be simplified by finding the screw matrix $\boldsymbol{S}$ which belongs to the null space of $\boldsymbol{W}$. The screw matrix describes the coordinates and the magnitude of a line in space. The simplification is given by

$$\boldsymbol{W}^T\boldsymbol{S} = 0 \tag{3.28}$$

Finally, the inverse kinematics of the payload results in a set of agent configurations for positions in the x, y and z by solving

$$\boldsymbol{S}^T\boldsymbol{g} = 0 \tag{3.29}$$

The fixed connector length l_i defines a constraint on each agent position $\boldsymbol{x}_i$ within the set of configurations and is given by

$$\|\boldsymbol{x}_i - \tilde{\boldsymbol{p}}_i\| = l_i^2 \tag{3.30}$$

The new, constrained agent position $\tilde{\boldsymbol{x}}_i$ is finally given by normalizing each agent position $\boldsymbol{x}_i$ from the set given in (3.29) with respect to the constraint in (3.30), given by

$$\tilde{\boldsymbol{x}}_i = l_i \frac{\boldsymbol{x}_i}{\|\boldsymbol{x}_i\|} + \tilde{\boldsymbol{p}}_i \tag{3.31}$$

The hanging payload resembles the classic robotic Stewart platform in terms of solving the inverse kinematics problem for the set of connectors given a desired payload position and orientation. However , while the inverse kinematics of the Stewart platform will give finite solutions because of gravitational forces are counteracted by the rigid actuator joints, the inverse kinematics will give infinite solutions for the static equilibrium of the hanging payload. In order to limit down the numerical search of possible agent configurations, a set of constraints are introduced. The maximum difference between each individual tension is given by

$$|\max(\lambda_1 ... \lambda_i) - \min(\lambda_1 ... \lambda_i)| < \lambda_{lim} \tag{3.32}$$

The maximum individual tension of a single wire is given by

$$0 \leq (\lambda_1 ... \lambda_i) \leq \lambda_{max} \quad \text{where} \quad \lambda_{max} = \frac{1}{2}mg \tag{3.33}$$

The maximum altitude difference in the global frame between each individual quadcopter is given by

$$|\max(\tilde{\boldsymbol{x}}_1^z ... \tilde{\boldsymbol{x}}_i^z) - \min(\tilde{\boldsymbol{x}}_1^z ... \tilde{\boldsymbol{x}}_i^z)| < \tilde{\boldsymbol{x}}_{min}^z \tag{3.34}$$

The minimum altitude global quadcopter position in the global frame is given by

$$(\tilde{\boldsymbol{x}}_1^z ... \tilde{\boldsymbol{x}}_i^z) = \tilde{\boldsymbol{x}}_{min}^z \tag{3.35}$$

In addition, the normalized altitude is fixed to given position such that the configuration is constrained to all agents set at equal altitudes.

Examples Two examples are given on how the payload model can be used in order to create initial agent positions in x, y and z, for a desired payload orientation, and how the resulting tension values for each configuration can be used as known disturbances for robustness in the control design. Figure 3.3 presents the scenario where a payload with mass $m = 1$ kg has no orientation angles $O = \{0, 0, 0\}$ and is fixed around the centre point $\boldsymbol{p}_c = [0, 0, 0]^T$. Given the initial payload positions are $\boldsymbol{p}_1 = [-1.7321 \quad -1.0 \quad 0.0]^T$, $\boldsymbol{p}_2 = [1.7321 \quad -1.0 \quad 0.0]^T$ and $\boldsymbol{p}_3 = [0 \quad 2 \quad 0]^T$, and zero translation $T = [0 \quad 0 \quad 0]^T$, with appropriate selected constraints for (3.32) to

(3.35), three different sets of quadcopter configurations satisfy the static equilibrium of the payload.

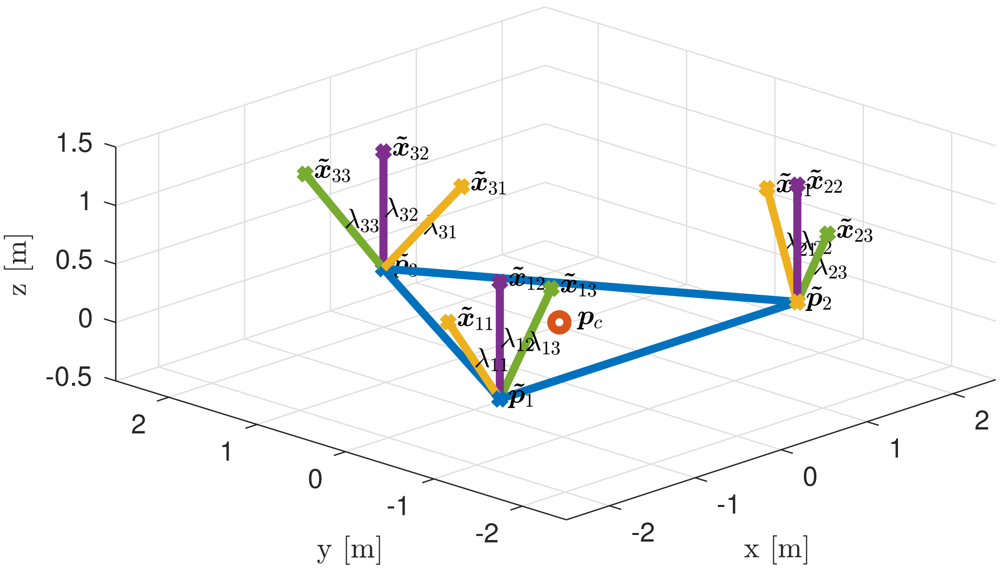


Figure 3.3: Flat hanging payload with three possible quadcopter configurations.

Table 3.3 presents details regarding the resulting quadcopter positions i and connector tensions for each of the three configurations j. Configuration $j = 2$ achieves equal connector tension for all quadcopters when the positions $\tilde{\boldsymbol{x}}_{i2}$ are above each payload connection points. As the quadcopters have different x and y positions in configurations $j = 1$ and $j = 3$, the resulting line tension increases. Since the connector lengths are rigid and cannot be stretched, the resulting z position of quadcopter i, decreases as the x and y positions are different from $\tilde{\boldsymbol{p}}_i$.

Table 3.3: Flat hanging payload example with three different quadcopter configurations with associated tensions.

$\tilde{\boldsymbol{x}}_{ij}$	x_{ij} [m]	y_{ij} [m]	z_{ij} [m]	λ_{ij} [N]
$\tilde{\boldsymbol{x}}_{11}$	-2.3321	1.9779	0.3273	4.0875
$\tilde{\boldsymbol{x}}_{21}$	-1.0	-0.41754	1.4331	4.2169
$\tilde{\boldsymbol{x}}_{31}$	0.8	0.7755	0.7559	4.3258
$\tilde{\boldsymbol{x}}_{12}$	-1.7321	1.7321	0.0	3.27
$\tilde{\boldsymbol{x}}_{22}$	-1.0	-1.0	2.0	3.27
$\tilde{\boldsymbol{x}}_{32}$	1.0	1.0	1.0	3.27
$\tilde{\boldsymbol{x}}_{13}$	-1.1321	1.4862	-0.3273	4.0875
$\tilde{\boldsymbol{x}}_{23}$	-1.0	-1.5816	2.5669	4.2169
$\tilde{\boldsymbol{x}}_{33}$	0.8	0.7755	0.7559	4.3258

For the second scenario, the desired orientation has been set to $O = \{0, \pi/12, \pi/8\}$. Figure 3.4 illustrates the static tilted payload with three different quadcopter orientations.

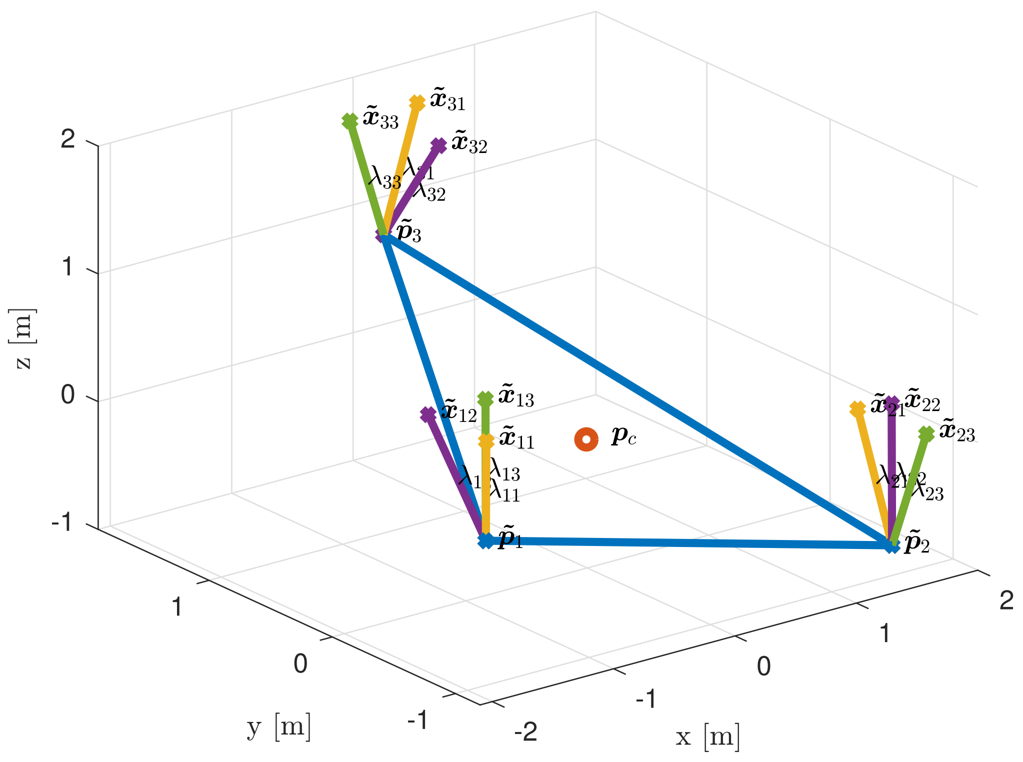


Figure 3.4: Tilted hanging payload with three possible quadcopter configurations.

Again, details from the results are presented in Table 3.4. A variation of tensions are achieved for different configurations.

Table 3.4: Tilted hanging payload example with three different quadcopter configurations with associated tensions.

$\tilde{\boldsymbol{x}}_{ij}$	x_{ij} [m]	y_{ij} [m]	z_{ij} [m]	λ_{ij} [N]
$\tilde{\boldsymbol{x}}_{11}$	-2.0078	1.5365	0.4754	2.9661
$\tilde{\boldsymbol{x}}_{21}$	-1.1596	-0.6815	1.8478	3.5471
$\tilde{\boldsymbol{x}}_{31}$	1.0215	0.1515	0.7001	2.6658
$\tilde{\boldsymbol{x}}_{12}$	-2.0078	1.8097	0.1981	3.7264
$\tilde{\boldsymbol{x}}_{22}$	-0.6882	-0.6882	1.4005	3.3674
$\tilde{\boldsymbol{x}}_{32}$	1.0215	0.1249	1.6337	2.4403
$\tilde{\boldsymbol{x}}_{13}$	-1.5364	1.6115	-0.0793	3.2985
$\tilde{\boldsymbol{x}}_{23}$	-0.6882	-1.1662	1.8478	2.5058
$\tilde{\boldsymbol{x}}_{33}$	1.0215	0.1515	1.7001	3.4033

The results from the payload modelling is used as a reference generator for desired formation flying which is needed to keep the payload orientation during flight. The payload coordinates of $\tilde{\boldsymbol{p}}_1$, $\tilde{\boldsymbol{p}}_2$ and $\tilde{\boldsymbol{p}}_3$ can be used to find initial and final target references $\boldsymbol{x}_{i,tref} = \tilde{\boldsymbol{x}}_i = [x\ y]^T$, where the distance between $\tilde{\boldsymbol{x}}_1$, $\tilde{\boldsymbol{x}}_2$ and $\tilde{\boldsymbol{x}}_3$, can be used as desired formation distance r_{fref}. More on formation flying is presented in 3.3.1.

Disturbance Free flying without the payload will be subject to disturbance caused by gravitational force, air resistance and physical model mismatch from the real system. When the payload is attached, the quadcopter will have the additional tension forces λ from (3.27). This known disturbance is mapped in to appropriate disturbance components d_x, d_y and d_z from (3.14), (3.15) and (3.16), by normalizing the vector between the current quadcopter position in $\boldsymbol{x}$ and the payload connection point $\tilde{\boldsymbol{p}}$. This vector is equal to the tension value λ which is then multiplied in such that the resulting components are given by

$$\boldsymbol{D} = \lambda \frac{\tilde{\boldsymbol{p}} - \boldsymbol{x}}{\|\tilde{\boldsymbol{p}} - \boldsymbol{x}\|} \tag{3.36}$$

where $\boldsymbol{D} = [d_x\ d_y\ d_z]^T$ from (3.3). Further, in order to use this information in the control design, (3.37) is rewritten to

$$\begin{aligned} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} &= \frac{1}{m}\boldsymbol{R}_{\phi\theta\psi}\boldsymbol{T} + \frac{1}{m}\boldsymbol{F_D} - \frac{1}{m}\begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + \frac{1}{m}\boldsymbol{D} \\ &= \frac{1}{m}\boldsymbol{R}_{\phi\theta\psi}\boldsymbol{T} + \frac{1}{m}\boldsymbol{F_D} + \tilde{\boldsymbol{D}} \end{aligned} \tag{3.37}$$

where the disturbances for gravitational force g, and connector tension and model mismatch $\tilde{\boldsymbol{D}}$ are augmented and given by

$$\tilde{\boldsymbol{D}} = \begin{bmatrix} \tilde{d}_x \\ \tilde{d}_y \\ \tilde{d}_z \end{bmatrix} = \begin{bmatrix} \frac{d_x}{m} \\ \frac{d_y}{m} \\ -g + \frac{d_z}{m} \end{bmatrix} \tag{3.38}$$

To summarize, the disturbance model $\tilde{\boldsymbol{D}}$ contains information regarding model mismatch such as inaccurate parameters, connector tension and gravitational force.

3.2 Control Design

This section covers control design, including model simplification, linearization and discretization. Solutions for disturbance compensation are proposed using feed-forward and integrator action. Finally, by using time-variant input constraints in the controllers, the actuator saturation limits are not exceeded.

3.2.1 General Overview

The complete nonlinear model has been decoupled such that three individual controllers each keep track on position, attitude and altitude respectively. The control scheme is presented in Figure 3.5. MPC for position control delivers necessary reference angles ϕ_{ref} and θ_{ref} to MPC attitude, in parallel to a desired ψ_{ref}. Attitude controller delivers necessary torques τ_ϕ, τ_θ and τ_ψ to achieve reference tracking which again, steers true position towards references x_{ref} and y_{ref}. The position controller has no information regarding current heading ψ, hence the attitude references ϕ_{ref} and θ_{ref} are rotated with respect to ψ prior to passing them in to MPC attitude. In parallel to position and attitude control, an altitude controller keeps track on desired z_{ref} by delivering total thrust $\boldsymbol{T}$ to the system, counteracting gravitational and disturbance forces on the system. Finally, $\boldsymbol{T}$, τ_ϕ, τ_θ and τ_ψ are converted to PWM signals $u_i,\ i = \{1...4\}$ which are passed on to the quadcopter.

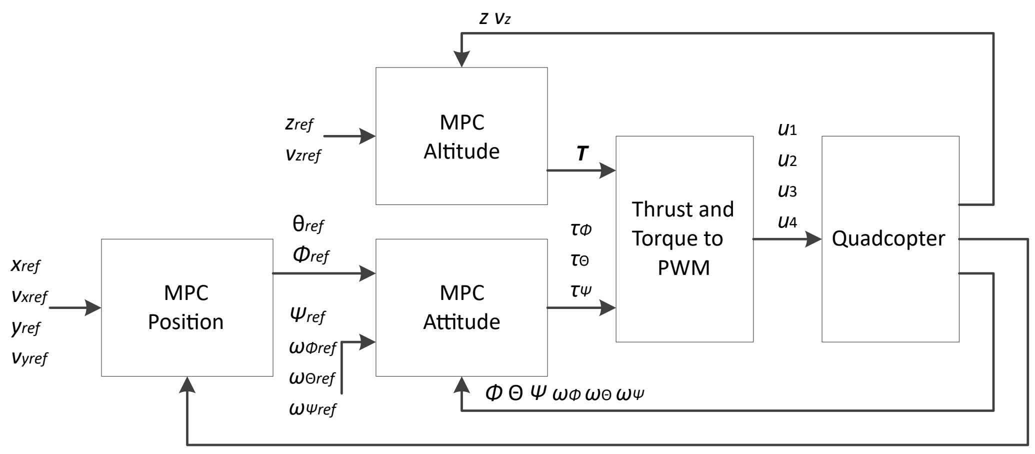


Figure 3.5: Block diagram of MPC scheme.

3.2.2 Model Simplification and Linearization

All three controllers are based on three LTI state space models derived from the complete nonlinear model presented in (3.11) to (3.22). The state space models in the form $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$, are found using the Jacobian matrix evaluated around equilibrium points where there is no translational or rotational motion and with zero input to the system. In addition, prior to solving the Jacobian matrix, a small angle approximation is applied to the motions models given by

$$\sin\angle \approx \angle \qquad \cos\angle \approx 1 - \frac{\angle^2}{2} \qquad \tan\angle \approx \angle \tag{3.39}$$

where $\angle$ is replaced by the attitude angles ϕ, θ and ψ.

Altitude controller For the altitude model, the equations presented in (3.13) and (3.16) together with the disturbance component $\tilde{d_z}$ from (3.38), are written in compact form in terms of total thrust $\boldsymbol{T}$ and with the small angle approximation. The model is given by

$$\begin{aligned} \dot{z} &= v_z \\ \dot{v}_z &= -\frac{k_d}{m} v_z + \left(1 - \frac{\theta^2}{2}\right)\left(1 - \frac{\phi^2}{2}\right) G \end{aligned} \tag{3.40}$$

where $G = \frac{1}{m}\boldsymbol{T} + \tilde{d_z}$, is the system input. The equilibrium points for the simplified model are given by

$$\boldsymbol{x}_{alt,0} = [z \;\; v_z]^T = [0 \;\; 0]^T \qquad \boldsymbol{u}_{alt,0} = G = 0 \tag{3.41}$$

where the resulting linearized state space model is given by

$$\begin{bmatrix} \dot{z} \\ \dot{v}_z \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{k_d}{m} \end{bmatrix} \begin{bmatrix} z \\ v_z \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} G \tag{3.42}$$

Details regarding the control input shows that when a total force $\boldsymbol{T}$ equal to the total disturbance force $\tilde{d_z}$ in z direction is applied to the system, the quadcoptor will hover. In fact, the input G forms the base of disturbance compensation for altitude control which is presented in (3.2.4).

Position controller Further, the position equations presented in (3.11), (3.12), (3.14) and (3.15) are also written in compact form in terms of total thrust $\boldsymbol{T}$ from (3.7) and with the small angle approximation. The model is given by

$$\begin{aligned} \dot{x} &= v_x \\ \dot{v}_x &= -\frac{k_d}{m} v_x + \frac{1}{m}\left(\psi\phi + \left(1 - \frac{\psi^2}{2}\right)\left(1 - \frac{\phi^2}{2}\right)\theta\right) g + \frac{1}{m} d_x \\ \dot{y} &= v_y \\ \dot{v}_y &= -\frac{k_d}{m} v_y + \frac{1}{m}\left(\left(1 - \frac{\phi^2}{2}\right)\psi\theta - \left(1 - \frac{\psi^2}{2}\right)\phi\right) g + \frac{1}{m} d_y \end{aligned} \tag{3.43}$$

where $g = \frac{\boldsymbol{T}}{m}$ decouples the input $\boldsymbol{T}$ from the equations by introducing the constant driving gravitational disturbance force g. The driving thrust for position is based on the assumption of zero offset reference tracking in a hovering z position. As the quadcopter is kept hovering, the input thrust $\boldsymbol{T}$ of altitude controller will cancel out the disturbance g. In order to reach a desired x and y position, the quadcopter needs an angle in θ and ϕ respectively, hence as a result of model decoupling, the inputs are now set to be θ and ϕ. Equilibrium points for the simplified and decoupled model are given by

$$\boldsymbol{x}_{pos,0} = [x \;\; v_x \;\; y \;\; v_y]^T = [0 \;\; 0 \;\; 0 \;\; 0]^T \qquad \boldsymbol{u}_{pos,0} = [\theta \;\; \phi]^T = [0 \;\; 0]^T \tag{3.44}$$

where the resulting linearized state space model is given by

$$\begin{bmatrix} \dot{x} \\ \dot{v_x} \\ \dot{y} \\ \dot{v_y} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{k_d}{m} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{k_d}{m} \end{bmatrix} \begin{bmatrix} x \\ v_x \\ y \\ v_y \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ g & 0 \\ 0 & 0 \\ 0 & -g \end{bmatrix} \begin{bmatrix} \theta \\ \phi \end{bmatrix} \tag{3.45}$$

Attitude controller Finally, the attitude equations presented in (3.17) to (3.22) are written in compact form in terms of torques τ_ϕ, τ_θ and τ_ψ and with the small angle approximation. The model is given by

$$\begin{aligned} \dot{\phi} &= \omega_\phi + \omega_\theta (\phi\theta) + \omega_\psi \left(\left(1 - \frac{\phi^2}{2}\right) \theta \right) \\ \dot{\omega}_\phi &= \frac{1}{I_{xx}} \tau_\phi - \left(\frac{I_{yy} - I_{zz}}{I_{xx}} \right) \omega_\theta \omega_\psi \\ \dot{\theta} &= \omega_\theta \left(1 - \frac{\phi^2}{2}\right) - \omega_\psi \phi \\ \dot{\omega}_\theta &= \frac{1}{I_{yy}} \tau_\theta - \left(\frac{I_{zz} - I_{xx}}{I_{yy}} \right) \omega_\phi \omega_\psi \\ \dot{\psi} &= \frac{\phi}{\left(1 - \frac{\theta^2}{2}\right)} \omega_\theta + \frac{\left(1 - \frac{\phi^2}{2}\right)}{\left(1 - \frac{\theta^2}{2}\right)} \omega_\psi \\ \dot{\omega}_\psi &= \frac{1}{I_{zz}} \tau_\psi - \left(\frac{I_{xx} - I_{yy}}{I_{zz}} \right) \omega_\phi \omega_\theta \end{aligned} \tag{3.46}$$

where the driving inputs are the torques τ_ϕ, τ_θ and τ_ψ. As the position controller passes the calculated input angle as a reference to the attitude controller, reference tracking in position depends on reference tracking in attitude. Equilibrium points are given by

$$\boldsymbol{x}_{att,0} = [\phi \;\; \omega_\phi \;\; \theta \;\; \omega_\theta \;\; \psi \;\; \omega_\psi]^T = [0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0]^T \qquad \boldsymbol{u}_{att,0} = [\tau_\phi \;\; \tau_\theta \;\; \tau_\psi]^T = [0 \;\; 0 \;\; 0]^T \tag{3.47}$$

where the resulting linearized state space model is given by

$$\begin{bmatrix} \dot{\phi} \\ \dot{\omega}_\phi \\ \dot{\theta} \\ \dot{\omega}_\theta \\ \dot{\psi} \\ \dot{\omega}_\psi \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \omega_\phi \\ \theta \\ \omega_\theta \\ \psi \\ \omega_\psi \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{I_{xx}} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix} \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \tag{3.48}$$

The references θ_{ref} and ϕ_{ref} passed from position controller into the attitude controller do not take the current heading ψ in to account. The position controller assumes zero heading when calculating θ_{ref} and ϕ_{ref}. As a result of this, the references are adjusted according to the heading. The adjustment is a 2D rotation in XY plane given by

$$\begin{bmatrix} \bar{\phi}_{ref} \\ \bar{\theta}_{ref} \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} \phi_{ref} \\ \theta_{ref} \end{bmatrix} \tag{3.49}$$

where the resulting $\bar{\phi}_{ref}$ and $\bar{\theta}_{ref}$ are the ψ adjusted references passed to the attitude controller.

3.2.3 Discretization

For implementation purpose, the linear controller models (3.42), (3.45) and (3.48) are all discretized using the approximation $\boldsymbol{x}_{k+1} = (\boldsymbol{I} + \Delta t A)\boldsymbol{x}_k + \Delta t B \boldsymbol{u}_k$, where $\boldsymbol{I}$ is an identity matrix and Δt is the time step between each sample. The discrete time state space model for altitude is given by

$$\begin{bmatrix} z_{k+1} \\ v_{z,k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 - \Delta t \frac{k_d}{m} \end{bmatrix} \begin{bmatrix} z_k \\ v_{z\,k} \end{bmatrix} + \begin{bmatrix} 0 \\ \Delta t \end{bmatrix} G_k \tag{3.50}$$

The discrete time state space model for position is given by

$$\begin{bmatrix} x_{k+1} \\ v_{x,k+1} \\ y_{k+1} \\ v_{y,k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 - \Delta t \frac{k_d}{m} & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 - \Delta t \frac{k_d}{m} \end{bmatrix} \begin{bmatrix} x_k \\ v_{x\,k} \\ y_k \\ v_{y\,k} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ g\Delta t & 0 \\ 0 & 0 \\ 0 & -g\Delta t \end{bmatrix} \begin{bmatrix} \theta_k \\ \phi_k \end{bmatrix} \tag{3.51}$$

The discrete time state space model for attitude is given by

$$\begin{bmatrix} \phi_{k+1} \\ \omega_{\phi,k+1} \\ \theta_{k+1} \\ \omega_{\theta,k+1} \\ \psi_{k+1} \\ \omega_{\psi,k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \phi_k \\ \omega_{\phi,k} \\ \theta_k \\ \omega_{\theta,k} \\ \psi_k \\ \omega_{\psi,k} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ \Delta t I_{xx}^{-1} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \Delta t I_{yy}^{-1} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \Delta t I_{zz}^{-1} \end{bmatrix} \begin{bmatrix} \tau_{\phi,k} \\ \tau_{\theta,k} \\ \tau_{\psi,k} \end{bmatrix} \tag{3.52}$$

3.2.4 Disturbance Compensation

Disturbance compensation is necessary for the robustness of the overall system in order to cope with model mismatch during control prediction and also to compensate for the payload and the gravitational force. As a result of the model linearization, the disturbance terms are gone and the controller models are no longer aware of the disturbances. However, these same terms are added back again using the idea of feed forward running in parallel to the control signal. This is the case for both altitude and position control, where the nonlinear model describes the disturbance forces $\tilde{\boldsymbol{D}}$ from (3.38).

Altitude controller For altitude control, the input signal $G = \frac{1}{m}\boldsymbol{T} + \tilde{d}_z$ from (3.40), contains the feed forward disturbance compensation $\tilde{d}_z$. The total thrust applied to the quadcopter is given by

$$\boldsymbol{T} = m\left(G - \tilde{d}_z\right) \tag{3.53}$$

where $\tilde{d}_z$ will provide a constant base thrust which cancels out the disturbances.

Position controller Position control has a similar design to the disturbance compensation for altitude control, where in addition to the driving force g from (3.45) which is multiplied by the input angles θ and ϕ, the disturbance forces $\tilde{d}_x$ and $\tilde{d}_y$ are added in terms of feed forward. The disturbance compensated angle references which are then passed to the attitude controller are given by

$$\begin{aligned}\tilde{\theta}_{ref} &= \theta_{ref} + \frac{m}{g}\tilde{d}_x \\ \tilde{\phi}_{ref} &= \phi_{ref} + \frac{m}{g}\tilde{d}_y\end{aligned} \tag{3.54}$$

where $\tilde{\theta}_{ref}$ and $\tilde{\phi}_{ref}$ are the new input signals. The estimated disturbances out from the observer are defined as $\tilde{d}_x = \frac{d_x}{m}$ and $\tilde{d}_y = \frac{d_y}{m}$. In order to only feed forward the true estimated disturbances d_x and d_y, g and m are cancelled out on the disturbance terms in (3.54) when multiplied by the input dynamic matrix $\boldsymbol{B}$ of (3.45).

Attitude controller Finally for attitude control, the model (3.46) does not specify any disturbances. However integrators are introduced running in parallel to τ_θ and τ_ϕ to compensate for unevenly distributed mass such as battery, which causes the quadcopter to tilt towards the heavier side which results in a attitude reference offset. The offset compensated torques are given by

$$\begin{aligned}\tilde{\tau_\theta} &= \tau_\theta + k_i \int_k^{k+\Delta t} \left(\theta - \bar{\theta}_{ref}\right) \Delta t \\ \tilde{\tau_\phi} &= \tau_\phi + k_i \int_k^{k+\Delta t} \left(\phi - \bar{\phi}_{ref}\right) \Delta t\end{aligned} \tag{3.55}$$

where $\theta - \bar{\theta}_{ref} = \theta_{err}$ and $\phi - \bar{\phi}_{ref} = \phi_{err}$, which are the reference errors for attitude control, $\bar{\phi}_{ref}$ and $\bar{\theta}_{ref}$ are the ψ adjusted references from (3.49), and k_i is the integral gain. To summarize the disturbance compensation, the control scheme from Figure 3.5 is updated with feed forward functionality and integrator action, and presented in Figure 3.6.

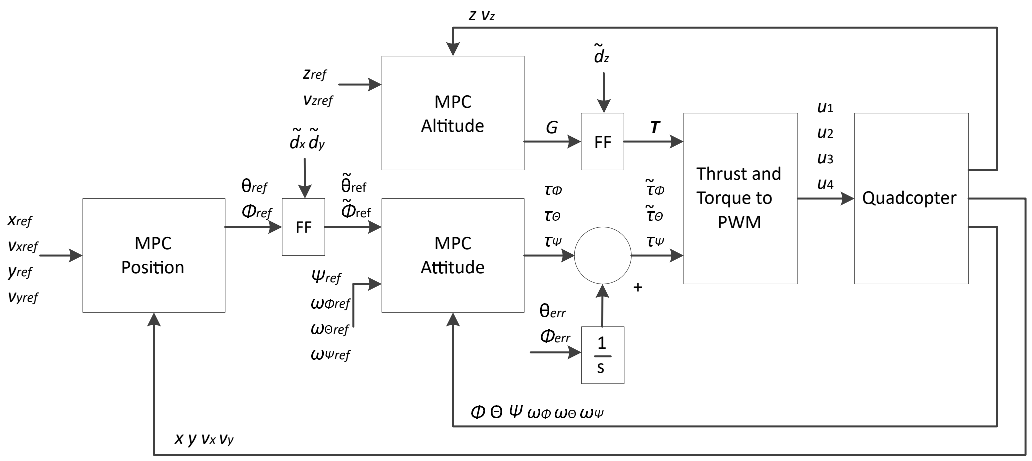


Figure 3.6: Block diagram of MPC scheme with disturbance compensation using feed forward and integration.

3.2.5 Time-Variant Input Constraints

The quadcopter has physical actuator constraints where the input signals saturate at PWM 0% and 100%. There is a risk of exceeding such saturation limits when using feed forward disturbance compensation. since the delivered control action is a sum of both the MPC input and the disturbance term. In order to exploit the input constraints within the MPC such that optimal control actions are delivered, the feed-forward terms are passed in to the MPC as time-variant input constraints. This procedure is carried out prior to calculating the control action since the feed-forward disturbance term is estimated earlier within the observer. From (3.57) the PWM is a function of the total thrust $\boldsymbol{T}$ from MPC altitude and torques τ_ϕ, τ_θ and τ_ψ from MPC attitude. Among the three controllers, position controller is computed first as it delivers the input angles ϕ and θ, which are after feed forward disturbance compensation and ψ adjustment passed to MPC attitude as references as $\bar{\phi}_{ref}$ and $\bar{\theta}_{ref}$. The MPC position model is based on a small angle approximation, hence the controller has input constraints equal to $[-10, 10]$ degrees. This constraint will also dampen aggressive translational motion. In order to not exceed the limit angles, the input constraints are functions of the feed forward contributions $\tilde{d}_x$ and $\tilde{d}_y$ from (3.54) and given by

$$\begin{aligned} \theta_{max} &= \frac{10\pi}{180} - \frac{m}{g}\tilde{d}_x & \theta_{min} &= -\frac{10\pi}{180} - \frac{m}{g}\tilde{d}_x \\ \phi_{max} &= \frac{10\pi}{180} - \frac{m}{g}\tilde{d}_y & \phi_{min} &= -\frac{10\pi}{180} - \frac{m}{g}\tilde{d}_y \end{aligned} \tag{3.56}$$

The attitude controller input torques are the second to be computed. This MPC has no input constraints. Priority of computing the torques prior to thrust is important with the reason that thrust is a necessary base in order to apply torque to the quadcopter. The torques are differences in PWM applied to the motor speed controllers. For the scenario where the quadcopter is thrusting at 100% in positive z direction, there will be no room for attitude control since the differences in applied PWM signal will be physically saturated. Hence, MPC altitude will have input constraint based on the current computed torque from MPC attitude. A result of this coupling is that there will always be a base thrust provided by MPC altitude in order for MPC attitude to apply differences in PWM. MPC altitude is last to compute necessary thrust. The time-variant constraints are based on the PWM, torque and thrust relations from (3.37), the altitude feed forward disturbance compensation from (3.53) and the minimum and maximum saturation for PWM. The set of PWM signals u_i for $i = \{1...4\}$ from (3.37) with input thrust $T = m(G - \tilde{d}_z)$, is given by

$$\begin{aligned} u_1 &= \sqrt{(m(G-\tilde{d}_z)Lb - 2b\tau_\phi + Lk\tau_\psi)/(4Lbc_mk)} \\ u_2 &= \sqrt{(m(G-\tilde{d}_z)Lb - 2b\tau_\theta - Lk\tau_\psi)/(4Lbc_mk)} \\ u_3 &= \sqrt{(m(G-\tilde{d}_z)Lb + 2b\tau_\phi + Lk\tau_\psi)/(4Lbc_mk)} \\ u_4 &= \sqrt{(m(G-\tilde{d}_z)Lb + 2b\tau_\theta - Lk\tau_\psi)/(4Lbc_mk)} \end{aligned} \tag{3.57}$$

In order to translate the minimum and maximum PWM signal to equivalent altitude input G, the set of equations are solved with respect to G with the two minimum and maximum scenarios $u_i = 0, 100$ %. This returns two sets of equations, where the lower bound G_{min} is selected as the maximum value of the set representing $u_i = 0$ % and where the upper bound G_{max} is selected as the minimum value of the set representing $u_i = 100$ %. The selection is basically doing a worst case scenario between all motor inputs u_i in order to make sure that none of the four values exceed the minimum or maximum constraint. The set of equations given by

$$G_{min} = \max \begin{cases} \frac{2b\tau_\phi - Lk\tau_\psi + Lb\tilde{d}_z m}{Lbm} \\ \frac{2b\tau_\theta + Lk\tau_\psi + Lb\tilde{d}_z m}{Lbm} \\ \frac{-2b\tau_\phi - Lk\tau_\psi + Lb\tilde{d}_z m}{Lbm} \\ \frac{-2b\tau_\theta + Lk\tau_\psi + Lb\tilde{d}_z m}{Lbm} \end{cases} \quad G_{max} = \min \begin{cases} \frac{2b\tau_\phi - Lk\tau_\psi + 40000Lbc_m k + Lbm\tilde{d}_z}{Lbm} \\ \frac{2b\tau_\theta + Lk\tau_\psi + 40000Lbc_m k + Lbm\tilde{d}_z}{Lbm} \\ \frac{-2b\tau_\phi - Lk\tau_\psi + 40000Lbc_m k + Lbm\tilde{d}_z}{Lbm} \\ \frac{-2b\tau_\theta + Lk\tau_\psi + 40000Lbc_m k + Lbm\tilde{d}_z}{Lbm} \end{cases} \tag{3.58}$$

To summarize, by adding time-variant input constraints to the control scheme, the decoupled controllers become aware of the feed forward disturbance compensation, thus the calculated control signal will always be with respect to the currently remaining input range available for each particular controller.

3.2.6 Distributed and Decentralized Control Strategy in Coordinating Tasks

When performing a coordinated task between multiple agents such as formation flying, it is a merit to share more information between the agents related to the task at hand. Such information can range from what the remote agents states are, which is more crucial in context of formation flying, to what the decided control policies are, i.e. what the remote agents are planning on doing. In addition to this, the MPC scheme benefits by enabling sharing predicted information.
In multi agent system, communication delays are inevitable and is worth to be taken into account for when in context of sharing information between the agents. If the shared information is subject to large delays, they might not be worthy. The distributed control strategy mentioned in Chapter 2.3, has the opportunity to share not only current measurement states $\boldsymbol{x}_k^m$, but also predicted information $\boldsymbol{x}_{k+1:k+p}^p$. One can use such predictions to cancel out the effect of delays. For instance, instead of using received information with t_d seconds of delay, one can use the same delayed package but instead, the predicted information with a horizon of t_d seconds into the future.
As path-planning algorithms are carried out using the translational states (mainly in XY plane), the position controller is set to perform long predictions with an horizon $p = 20$ steps. Packages received from each remote agent can be acquainted with time stamps to notify about the potential time delay t_d. Consequently the path-planning algorithms would select prediction horizon information accordingly. Figure 3.7 illustrates the distributed MPC scheme between two agents and how it benefits from sharing predicted information in the presence of communication delay. Needless to say, this is expandable for any number of agents.

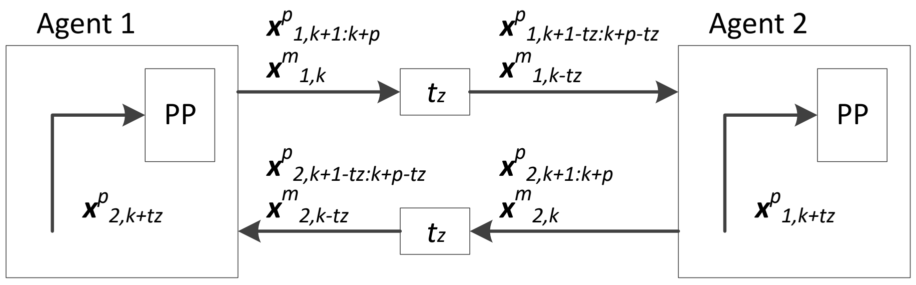


Figure 3.7: Distributed MPC with benefits of sharing predicted information considering a delay in communication lines

3.2.7 Controller Sampling Time

System analysis is necessary to identify at what sampling frequency the control system must run in order to capture necessary dynamics of the system. For MIMO systems, analysing the singular values of the dynamic system frequency response will indicate at which frequency the dynamics are located. By using `sigma(sys,w)` in `MATLAB`, the minimum sampling frequency is found at the cross over frequency, the point where the highest singular value is zero dB. The controller sampling frequencies are found separately for each model. For both altitude model from (3.42) and attitude model from (3.48), the inputs G, τ_ϕ, τ_θ and τ_ψ are replaced by the equivalent electrical PWM inputs from (3.9). This is done prior to the analysis, in order to appropriately scale the models using the motor, lift and drag constants c_m, k and b, and the quadcopter arm length L, so that the frequency response represents the physical inputs. As a result, the linearized models are all analysed around the input equilibrium point where the quadcopter is hovering. The hovering input is found by rearranging (3.16) equal to the gravitational force and solving it with respect to the input $\boldsymbol{u}$ where the angles, velocity and acceleration are all zero. The hovering input is given by

$$g = \frac{k\,c_m}{m}(u_1^2 + u_2^2 + u_3^2 + u_4^2) \quad \rightarrow \quad u_i = 56.7069\% \quad \forall i \tag{3.59}$$

Figure 3.8 presents the frequency response of the three models.

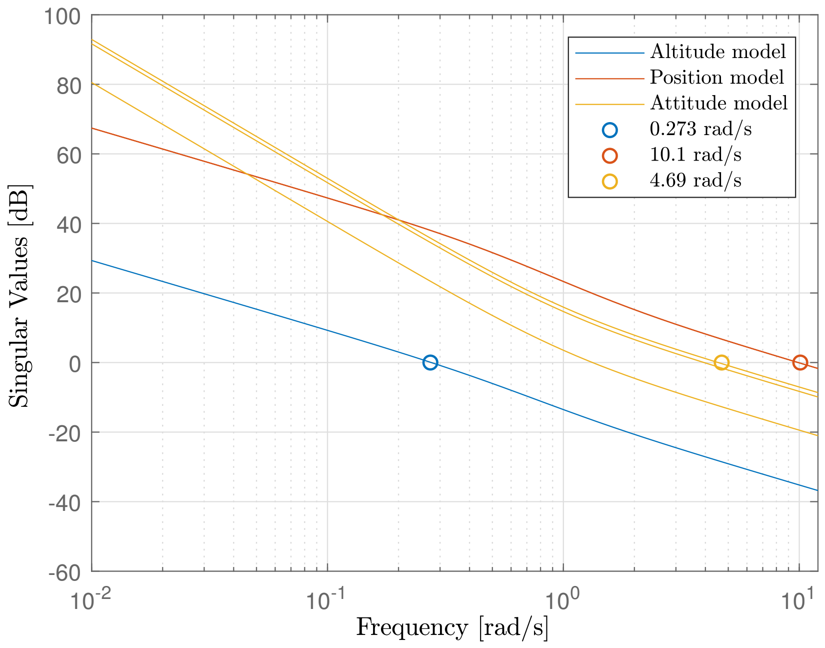


Figure 3.8: Singular values of the three different MPC models.

The collected minimum frequencies $f_{s,min}$ are multiplied by a factor of 10 times in order give the recommended sampling frequency $f_{s,rec}$ an error margin. Table 3.5 presents the frequencies with the associated recommended sampling times.

Table 3.5: Minimum and recommended controller frequencies and sampling times.

	$f_{s,min}$ [rad/s]	$f_{s,rec}$ [Hz]	T_s [s]
Altitude MPC	0.273	0.430	2.356
Position MPC	10.100	16.070	0.062
Attitude MPC	4.690	7.460	0.134

For both simulation and implementation, all controllers run at the equal sampling time. The recommended frequencies did however turn out to be too low for attitude control on the real implementation. As a result of this, the sampling times are all adjusted down to $T_s = 0.025$ s, where the frequency is $f_s = 40$ Hz.

3.2.8 MPC Terminal Cost for Stability

In pursue of designing a stable MPC controller, the terminal costs are used. As the theory suggests [21], terminal costs are derived solving the *Discrete Algebraic Ricatti Equation (DARE)*, which solves the stabilizable weights for a restrictive unconstrained infinite-horizon *Linear Quadratic Regulator (LQR)*. Using this stabilizable weights from LQR approach as terminal costs in MPC approach makes it possible to make finite-horizon MPC equivalent to an infinite-horizon LQR [21].

DARE solves for Q_f in

$$A^\dagger Q_f A - Q_f - A^\dagger Q_f B(R + B^\dagger Q_f B)^{-1} B^\dagger Q_f A + Q = 0 \qquad (3.60)$$

where A, B, Q and R are the motion model, input model, state cost and input cost respectively. Models and penalizing costs for all three controllers, altitude, position and attitude are each used used in DARE and resulting terminal costs have been used in attempt to make the MPC design theoretically stable.

3.3 Path-Planning

This section presents proposed solutions for formation flying and collision avoidance in details.

3.3.1 Formation Flying

A formation flying algorithm is necessary in order to keep desired payload orientation during flight, as varying distance between the quadcopters will cause orientation errors. Formation flying is achieved by following reference points for positions x and y that are dynamically created between the quadcopters based on an intersection point between two imaginary circles formed around each quadcopter. The principle is presented in Figure 3.9. Agent $\boldsymbol{x}_i$ for $i = \{1 ... 3\}$, will follow the closest intersection point between the circles of the two other agents. When all three agents are on reference, an equilateral triangle will be formed with the desired side length of r_{fref}.

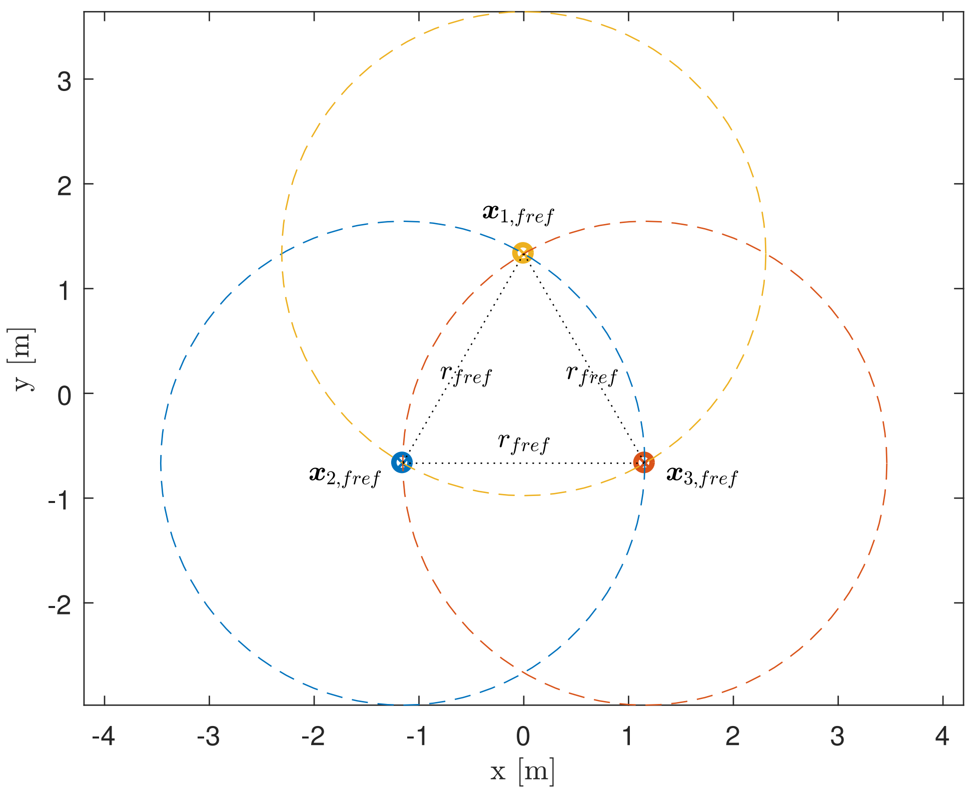

$\boldsymbol{x}_{1,fref}$
$\boldsymbol{x}_{2,fref}$
$\boldsymbol{x}_{3,fref}$
r_{fref}
r_{fref}
r_{fref}
y [m]
x [m]
3
2
1
0
-1
-2
-4
-3
-2
-1
0
1
2
3
4

Figure 3.9: Formation flying concept with three agents.

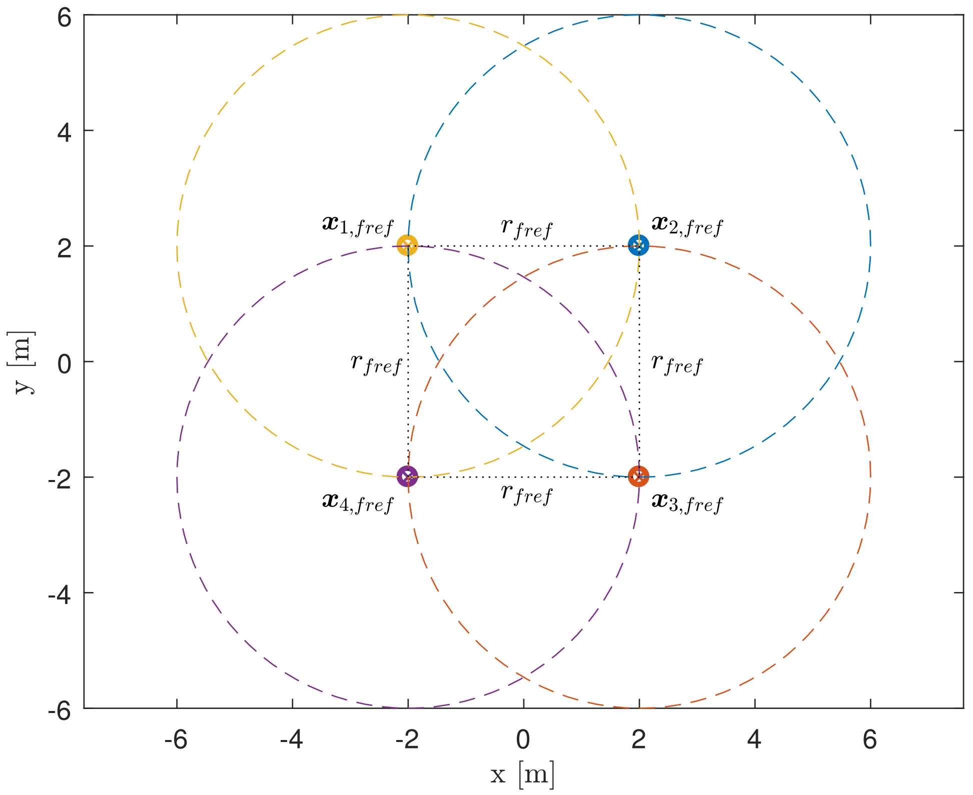
Algorithm Formation flying of three agents is based on the two circles given by

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 & = r_{fref}^2 \\ (x - x_2)^2 + (y - y_2)^2 & = r_{fref}^2 \end{cases} \tag{3.61}$$

where (x_1, y_1) and (x_2, y_2) are the circle centre points. The system of equations are solved for x and y which results in two formation reference candidates for $\boldsymbol{x}_{i,fref}$, denoted by the intersection points $\tilde{\boldsymbol{M}}_{1,c}^{xy} = [x\ y]^T$ and $\tilde{\boldsymbol{M}}_{2,c}^{xy} = [x\ y]^T$ on the edge of the circle radius r_{fref}. The distances d_1 and d_2 between each candidate point and current position of agent $\boldsymbol{x}_i$ are given by

$$d_1 = \left\| \tilde{\boldsymbol{M}}_{1,c}^{xy} - \boldsymbol{x}_i^{xy} \right\| \qquad d_2 = \left\| \tilde{\boldsymbol{M}}_{2,c}^{xy} - \boldsymbol{x}_i^{xy} \right\| \tag{3.62}$$

where the candidate point at the shortest distance becomes the formation reference $\boldsymbol{x}_{i,fref} = \boldsymbol{M}_{2,c}^{xy}$. $\boldsymbol{x}_i^{xy} = [x\ y]^T$ denotes position states x and y of agent i. The circle intersection holds only if the two circles actually intersect, e.g. for agent $\boldsymbol{x}_1$, the two other agents need to intersect given $\|\boldsymbol{x}_3^{xy} - \boldsymbol{x}_2^{xy}\| < 2r_{fref}$. The proposed solution is based on but not limited to three agents. As the number of agents increase, the number of intersections also increase, which emphasises that the formation reference for $\boldsymbol{x}_i$ will always be the closest intersection point when choosing between multiple candidates. Figures 3.10 and 3.11 illustrates the reference formation for four and five agents. The formation shape is changed from an equilateral triangle to a square as $i = \{1...4\}$ and to a pentagon shape as $i = \{1...5\}$. Again, the desired formation distance between each agent is achieved when all agents are on formation reference.



$\boldsymbol{x}_{1,fref}$
r_{fref}
$\boldsymbol{x}_{2,fref}$
r_{fref}
r_{fref}
$\boldsymbol{x}_{4,fref}$
r_{fref}
$\boldsymbol{x}_{3,fref}$
y [m]
x [m]
6
4
2
0
-2
-4
-6
-6
-4
-2
0
2
4
6

Figure 3.10: Formation flying concept with four agents.

The formation flying is implemented in to the position controller by augmenting the

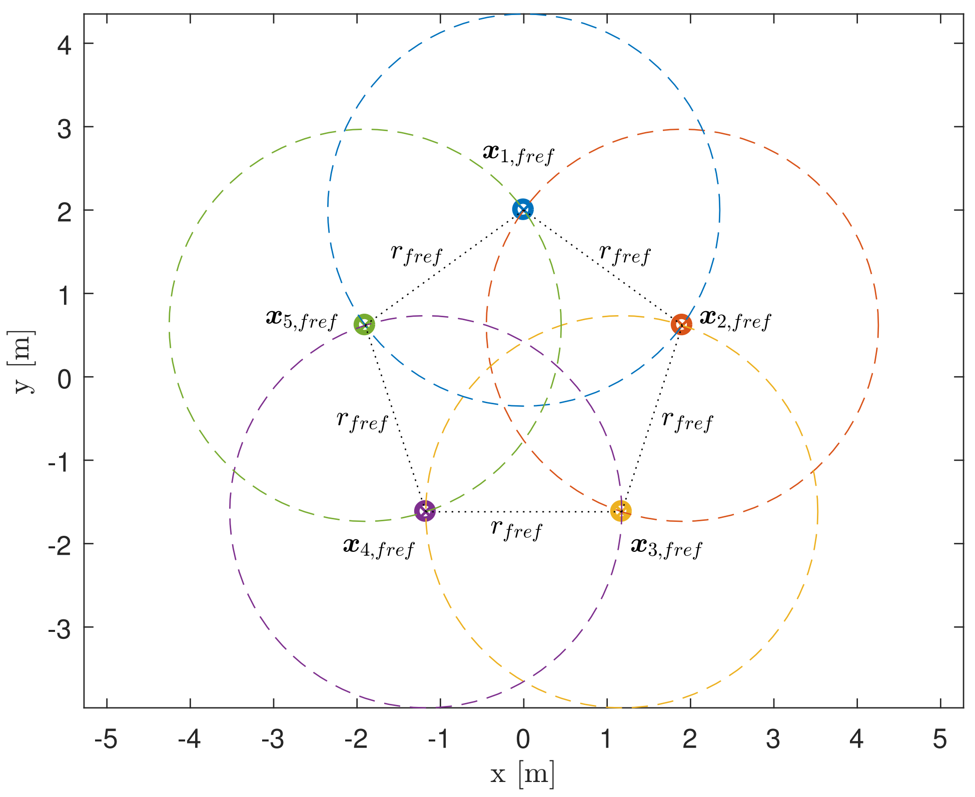


Figure 3.11: Formation flying concept with five agents.

state space model in (3.45) with a copy of the position states x and y given by

$$\begin{bmatrix} \dot{x} \\ \dot{v}_x \\ \dot{y} \\ \dot{v}_y \\ \dot{x}_f \\ \dot{y}_f \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -\frac{k_d}{m} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\frac{k_d}{m} & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ v_x \\ y \\ v_y \\ x_f \\ y_f \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ g & 0 \\ 0 & 0 \\ 0 & -g \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \phi \end{bmatrix} \tag{3.63}$$

where the new position states x_f and y_f are controlled with respect to the formation reference point $\boldsymbol{x}_{i,fref}$. The same augmented state space system in discrete time is given by

$$\begin{bmatrix} x_{k+1} \\ v_{x,k+1} \\ y_{k+1} \\ v_{y,k+1} \\ x_{f,k+1} \\ y_{f,k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1-\Delta t\frac{k_d}{m} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 1-\Delta t\frac{k_d}{m} & 0 & 0 \\ 0 & \Delta t & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \Delta t & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ v_{x,k} \\ y_k \\ v_{y,k} \\ x_{f,k} \\ y_{f,k} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ g\Delta t & 0 \\ 0 & 0 \\ 0 & -g\Delta t \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \theta_k \\ \phi_k \end{bmatrix} \tag{3.64}$$

3.3.2 Collision Avoidance

Collision avoidance is a necessary online path-planning feature when working on autonomous agents. How to approach it can be done in many differenet ways. The proposed solution is based on performing collision avoidance by creating any intermediate temporary reference change for positions x and y in order to steer the agent around an obstacle. In details, when the quadcopter is within a defined vicinity of an obstacle, the final target reference which is beyond the obstacle, will switch to a reference following an imaginary circle trajectory at a defined safety radius from the obstacle. By using the obstacle velocity as a base for setting priorities among multiple agents, the avoidance performance and flow can be increased. There are two scenarios, static Obstacle Collision Avoidance (OCA) and Inter-Vehicle Collision Avoidance (IVCA) between two dynamic agents. The decision making is summarized as a flow chart in Figure 3.12.

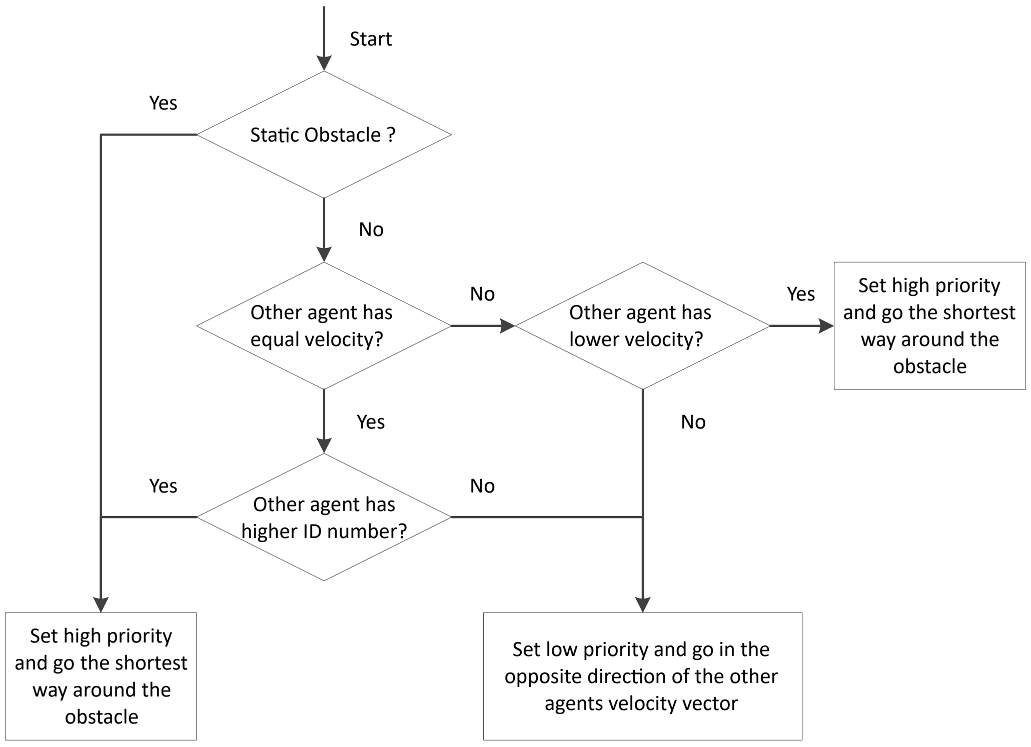

Start
Static Obstacle ?
Yes
No
Other agent has equal velocity?
No
Other agent has lower velocity?
Yes
Set high priority and go the shortest way around the obstacle
Yes
No
Other agent has higher ID number?
Yes
No
Set high priority and go the shortest way around the obstacle
Set low priority and go in the opposite direction of the other agents velocity vector

Figure 3.12: Collision avoidance decision making flow chart.

Static Obstacle Collision Avoidance For OCA, the obstacle velocity is zero, hence the moving agent is the one who needs to perform avoidance around the obstacle and is assigned higher priority. As the agent is within the imaginary circle from the obstacle, the final target reference point will change to go the shortest way around the circle.

Inter-Vehicle Collision Avoidance For IVCA, the agent with higher velocity is assigned higher priority and will perform avoidance by following the imaginary circle in the shortest way around the remote agent, similar to OCA scenario. The low priority, slow flying agent, will perform avoidance by following the imaginary circle in the opposite direction of the velocity vector of the higher priority agent. This selection is crucial in order to avoid a chain reaction which results in both agents going in parallel because the lower priority agent going towards where the higher priority agent is heading instead of going behind it. The slower flying agent has also got less energy to stop when approaching an object, hence it is easier to slow down and travel in the opposite direction of the other agents velocity. If the agents in IVCA have the same velocity up to some defined precision, the lower agent ID number receives the higher priority. Since a small difference in velocity can be due to different numerical rounding, there is a high chance that the algorithm will flip the priority between the agents back and forth.

Algorithm Two imaginary circles are created around the obstacle position $\boldsymbol{x}_{i,obs} = [x\ y]^T$ in the XY plane. The inner circle is given by a radius r_{obs} and forms the temporary collision avoidance trajectory. The outer circle is given by a radius $r_{obs,lim} > r_{obs}$ and forms a limit distance from the inner circle and is used to determine when the collision avoidance trajectory is to be followed. In order to perform collision avoidance, the current position of $\boldsymbol{x}_i$ must be within the outer circle and have the final reference $\boldsymbol{x}_{i,tref}$ beyond the obstacle such that the imaginary line between the current position and the final target reference, intersects the inner circle. When the current position of $\boldsymbol{x}_i$ is within the outer circle, the collision avoidance reference $\boldsymbol{x}_{i,cref} = [x\ y]^T$ is generated based on the following steps:

1. Find the closest intersection point between the inner circle and the imaginary line between $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i,tref}$,
2. Create a normal of the imaginary line at closest intersection point,
3. On the normal, travel α distance in both directions from the intersection point and create vectors towards the obstacle such that two new intersections are created on the inner circle,
4. If the obstacle has zero or lower velocity or the same velocity with bigger agent ID number than $\boldsymbol{x}_i$, then assign the new intersection point which has the shortest distance to $\boldsymbol{x}_{i,tref}$ as the temporary reference $A_{i,cref}$,
5. If the obstacle has higher velocity or have the same velocity with smaller agent ID number than $\boldsymbol{x}_i$, then velocity vector $\boldsymbol{v}_{obs} = [v_x\ v_y]^T$ is normalized with respect to the inner circle r_{obs} and negated. The closest new intersection point to the negated velocity vector $-\hat{\boldsymbol{v}}_{obs}$, is assigned as the temporary reference $\boldsymbol{x}_{i,cref}$,

Figure 3.13 presents two illustrations on the OCA and IVCA selection principles are carried out. The intersection points are created when the agent is within the outer circle. The highlighted point $\boldsymbol{x}_{i,cref}$ is set as the temporary avoidance reference. $\tilde{\boldsymbol{x}}_{i,cref}$ is the alternative candidate which is the other way to go around the obstacle.

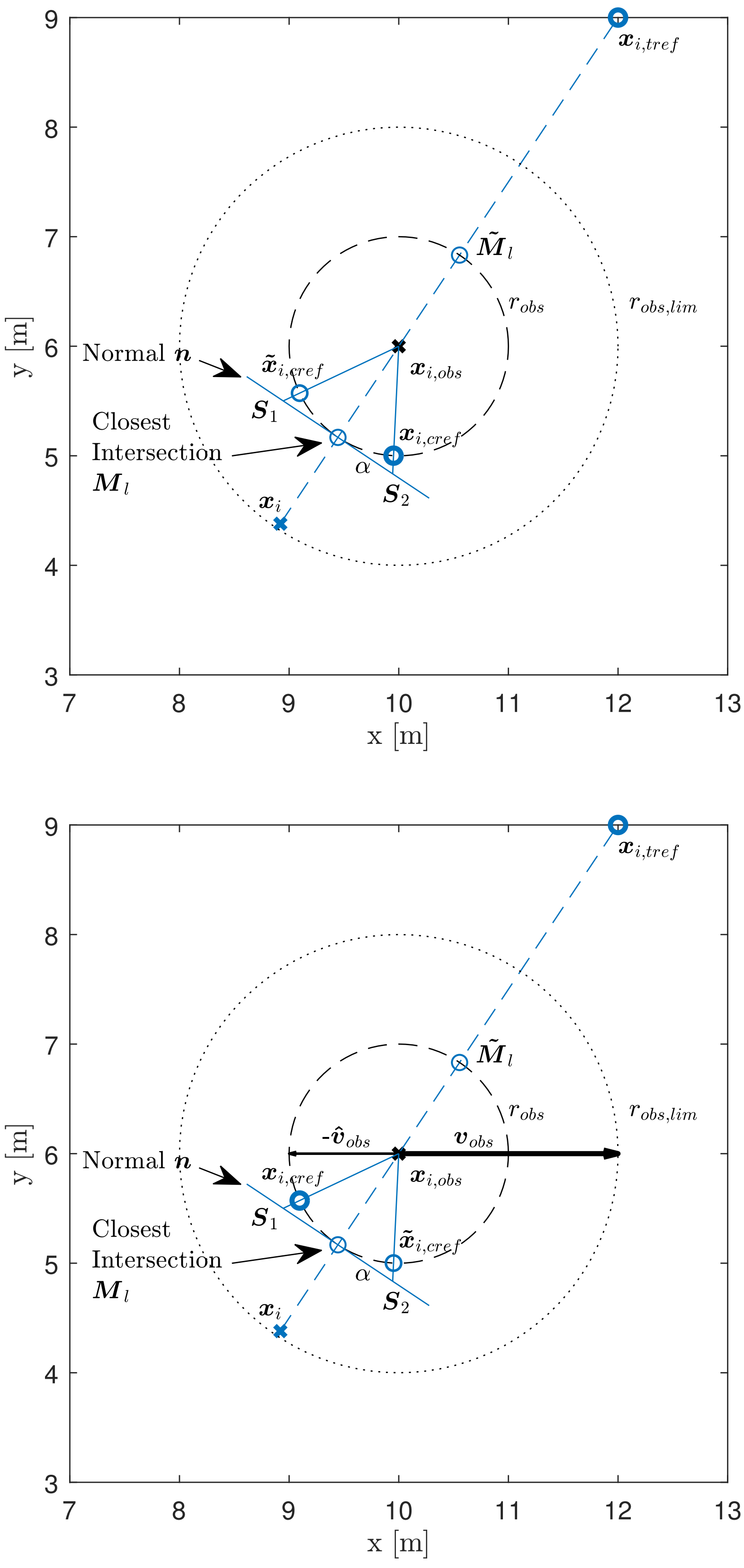

$\boldsymbol{x}_{i,tref}$
$\tilde{\boldsymbol{M}}_l$
r_{obs}
$r_{obs,lim}$
Normal $\boldsymbol{n}$
$\tilde{\boldsymbol{x}}_{i,cref}$
$\boldsymbol{x}_{i,obs}$
$\boldsymbol{S}_1$
Closest Intersection $\boldsymbol{M}_l$
$\boldsymbol{x}_{i,cref}$
α
$\boldsymbol{S}_2$
$\boldsymbol{x}_i$
y [m]
x [m]
$\boldsymbol{x}_{i,tref}$
$\tilde{\boldsymbol{M}}_l$
r_{obs}
$r_{obs,lim}$
-$\hat{\boldsymbol{v}}_{obs}$
$\boldsymbol{v}_{obs}$
Normal $\boldsymbol{n}$
$\boldsymbol{x}_{i,cref}$
$\boldsymbol{x}_{i,obs}$
$\boldsymbol{S}_1$
Closest Intersection $\boldsymbol{M}_l$
$\tilde{\boldsymbol{x}}_{i,cref}$
α
$\boldsymbol{S}_2$
$\boldsymbol{x}_i$
y [m]
x [m]

Figure 3.13: Collision avoidance selection principle. OCA principle in the top figure and IVCA principle in the bottom figure.

The intersection points two $\tilde{\boldsymbol{M}}_{1,l} = [x\ y]^T$ and $\tilde{\boldsymbol{M}}_{2,l} = [x\ y]^T$ between the inner circle and the crossing line is found by solving the quadratic system of equations given by

$$\begin{cases} a_l x + b_l y + c = 0, \\ (x - x_1)^2 + (y - y_1)^2 = r_{obs}^2 \end{cases} \tag{3.65}$$

where a_l, b_l and c_l define the line between $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i,tref}$ and (x_1, y_1) is the position of $\boldsymbol{x}_{i,obs}$. Solving the equations results in two intersection candidate points given by $\tilde{\boldsymbol{M}}_{1,l}$ and $\tilde{\boldsymbol{M}}_{2,l}$. The distances d_1 and d_2 between each intersection point and current position of agent $\boldsymbol{x}_i$ is given by

$$d_1 = \left\| \tilde{\boldsymbol{M}}_{1,l} - \boldsymbol{x}_i^{xy} \right\| \qquad d_2 = \left\| \tilde{\boldsymbol{M}}_{2,l} - \boldsymbol{x}_i^{xy} \right\| \tag{3.66}$$

where the closest intersection point $\boldsymbol{M}_l$ is chosen among the intersection candidate points. Two temporary avoidance reference candidates $\tilde{\boldsymbol{x}}_{i,cref}$ are created on the inner circle by finding the intersection points on the vectors from $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ to the obstacle $\boldsymbol{x}_{i,obs}$. The points $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ are located on the normal $\boldsymbol{n}$ with a distance α from $\boldsymbol{M}_l$. α is a tuning factor which decides the step size of how fast the collision avoidance reference generator moves around the inner circle as $\boldsymbol{x}_i$ approaches it.
The criteria for selecting $\boldsymbol{x}_{i,cref}$ among the two candidates depends on whether the obstacle is static or not. If the obstacle is static, has lower velocity than $\boldsymbol{x}_i$, or the agent velocities are equal but the $\boldsymbol{x}_i$ has lower identify number, the candidate $\tilde{\boldsymbol{x}}_{i,cref}$ with the shortest distance to the final reference $\boldsymbol{x}_{i,tref}$, is assigned to $\boldsymbol{x}_{i,cref}$. If the obstacle has faster velocity $\boldsymbol{v}_{i,obs}$ than $\boldsymbol{x}_i$, the candidate $\tilde{\boldsymbol{x}}_{i,cref}$ which has the shortest distance to $-\hat{\boldsymbol{v}}_{i,obs}$, is assigned to $\boldsymbol{x}_{i,cref}$. $\hat{\boldsymbol{v}}_{i,obs}$ is the obstacle velocity vector, normalized with respect the inner circle radius r_{obs}, given by

$$\hat{\boldsymbol{v}}_{i,obs} = r_{obs} \frac{\boldsymbol{v}_{i,obs}}{\|\boldsymbol{v}_{i,obs}\|} \tag{3.67}$$

In this way, the slower flying quadcopter will avoid the faster flying obstacle in the opposite direction of the obstacle velocity, hence the mentioned chain reaction is avoided.

3.3.3 Reference Ramp

Target references can be given in a ramp fashion ahead of the agent according to

$$\boldsymbol{x}_{tref,ramp} = \begin{cases} \hat{\boldsymbol{x}}^{xy} + \alpha \frac{\boldsymbol{x}_{tref - \hat{\boldsymbol{x}}^{xy}}}{|\boldsymbol{x}_{tref - \hat{\boldsymbol{x}}^{xy}}|}, & if \mid \boldsymbol{x}_{tref - \hat{\boldsymbol{x}}^{xy}} \mid \geqslant \alpha \\ \boldsymbol{x}_{tref}, & \text{otherwise.} \end{cases} \tag{3.68}$$

where the ramp reference is $\boldsymbol{x}_{tref,ramp} = [x\ y]^T$. Ramping the reference has several advantages. Firstly, aggressive agent motion and velocity can be controlled with respect to avoiding big reference changes. Secondly, it makes it easier to tune the controller weights between the target tracking $\boldsymbol{x}_{tref}$ and the formation tracking $\boldsymbol{x}_{fref}$. Regardless of how big the references changes are, ramp function will limit down the step changes and give it to each agent in smaller proportions. Finally, ramping allows more freedom with respect to keeping aggressive weights such that manoeuvrability does not become limited. If the weights are used purely to dampen the velocity when not using the ramp functionality, the quadcopter might be too limited due to actuator constraints, hence the risk of running in to a state where the quadcopter cannot recover from.

3.4 Observer

In general, model based control design relies on full state feedback. For the quadcopter, the number of measurements does not match the number of states, hence a state observer is necessary. This section describes the EKF based state observer and how it is used to provide full state feedback and estimation of unknown disturbances. In addition to the full state observer, a decoupled version is presented in order to reduce the computational effort on hardware. The main argument for using the nonlinear EKF observer is based on that the nonlinear motion model has more information and is more representative to the real system.

3.4.1 State and Disturbance Estimation

The EKF based observer uses an augmented version of the nonlinear quadcopter model, where the disturbance states $\tilde{d}_x$, $\tilde{d}_y$ and $\tilde{d}_z$ are added to the model in terms of the Gaussian random walk model $x_{k+1} = x_k + q_k$, with the normally distributed zero mean white noise $q_k \sim \mathcal{N}(0, Q)$ [20]. The idea is that given a prediction of the states, the disturbance components will converge over time to the true disturbances as the prediction mismatches the measurement, which is the innovation of the filter. The full extended nonlinear observer model in discrete time is given by

$$
\begin{aligned}
\hat{x}_{k+1} &= \hat{x}_k + \Delta t\, \hat{v}_{x,k} \\
\hat{y}_{k+1} &= \hat{y}_k + \Delta t\, \hat{v}_{y,k} \\
\hat{z}_{k+1} &= \hat{z}_k + \Delta t\, \hat{v}_{z,k} \\
\hat{v}_{x,k+1} &= \hat{v}_{x,k} + \Delta t \left(-\frac{k_d}{m}\hat{v}_{x,k} + \frac{1}{m}\left(\sin\hat{\psi}_k \sin\hat{\phi}_k + \cos\hat{\psi}_k \cos\hat{\phi}_k \sin\hat{\theta}_k\right) T_k + \hat{\tilde{d}}_{x,k} \right) \\
\hat{v}_{y,k+1} &= \hat{v}_{y,k} + \Delta t \left(-\frac{k_d}{m}\hat{v}_{y,k} + \frac{1}{m}\left(\cos\hat{\phi}_k \sin\hat{\psi}_k \sin\hat{\theta}_k - \cos\hat{\psi}_k \sin\hat{\phi}_k\right) T_k + \hat{\tilde{d}}_{y,k} \right) \\
\hat{v}_{z,k+1} &= \hat{v}_{z,k} + \Delta t \left(-\frac{k_d}{m}\hat{v}_{z,k} + \frac{1}{m}\left(\cos\hat{\theta}_k \cos\hat{\phi}_k\right) T_k + \hat{\tilde{d}}_{z,k} \right) \\
\hat{\phi}_{k+1} &= \hat{\phi}_k + \Delta t \left(\hat{\omega}_{\phi,k} + \hat{\omega}_{\theta,k} \sin\hat{\phi}_k \tan\hat{\theta}_k + \hat{\omega}_{\psi,k} \cos\hat{\phi}_k \tan\hat{\theta}_k \right) \\
\hat{\theta}_{k+1} &= \hat{\theta}_k + \Delta t \left(\hat{\omega}_{\theta,k} \cos\hat{\phi}_k - \hat{\omega}_{\psi,k} \sin\hat{\phi}_k \right) \\
\hat{\psi}_{k+1} &= \hat{\psi}_k + \Delta t \left(\frac{\sin\hat{\phi}_k}{\cos\hat{\theta}_k}\hat{\omega}_{\theta,k} + \frac{\cos\hat{\phi}_k}{\cos\hat{\theta}_k}\hat{\omega}_{\psi,k} \right) \qquad (3.69) \\
\hat{\omega}_{\phi,k+1} &= \hat{\omega}_{\phi,k} + \Delta t \left(\frac{1}{I_{xx}}\tau_{\phi,k} - \frac{I_{yy} - I_{zz}}{I_{xx}}\hat{\omega}_{\theta,k}\,\hat{\omega}_{\psi,k} \right) \\
\hat{\omega}_{\theta,k+1} &= \hat{\omega}_{\theta,k} + \Delta t \left(\frac{1}{I_{yy}}\tau_{\theta,k} - \frac{I_{zz} - I_{xx}}{I_{yy}}\hat{\omega}_{\phi,k}\,\hat{\omega}_{\psi,k} \right) \\
\hat{\omega}_{\psi,k+1} &= \hat{\omega}_{\psi,k} + \Delta t \left(\frac{1}{I_{zz}}\tau_{\psi,k} - \frac{I_{xx} - I_{yy}}{I_{zz}}\hat{\omega}_{\phi,k}\,\hat{\omega}_{\theta,k} \right) \\
\hat{\tilde{d}}_{x,k+1} &= \hat{\tilde{d}}_{x,k} \\
\hat{\tilde{d}}_{y,k+1} &= \hat{\tilde{d}}_{y,k}
\end{aligned}
$$

$$\hat{\hat{d}}_{z,k+1} = \hat{\hat{d}}_{z,k}$$

where the discrete time disturbance components $\hat{\hat{d}}_{x,k}$, $\hat{\hat{d}}_{y,k}$ and $\hat{\hat{d}}_{z,k}$ are influencing the translational motion states $\hat{v}_{x,k}$, $\hat{v}_{y,k}$ and $\hat{v}_{z,k}$. The associated state vector is given by

$$\hat{\boldsymbol{x}}_k = [\hat{x}_k \;\; \hat{y}_k \;\; \hat{z}_k \;\; \hat{v}_{x,k} \;\; \hat{v}_{y,k} \;\; \hat{v}_{z,k} \;\; \hat{\phi}_k \;\; \hat{\theta}_k \;\; \hat{\psi}_k \;\; \hat{\omega}_{\phi,k} \;\; \hat{\omega}_{\theta,k} \;\; \hat{\omega}_{\psi,k} \;\; \hat{\hat{d}}_{x,k} \;\; \hat{\hat{d}}_{y,k} \;\; \hat{\hat{d}}_{z,k}]^T \tag{3.70}$$

The disturbance estimations are expected to change dynamically due to mismatching model parameters, swinging payload and declining battery power which causes the total propeller thrust to drop as time goes. Introducing it to the observer is expected to improve the overall performance of the system by means of using close to real estimates of the actual true disturbances. The number of measurements $p = 9$ is less than the number of states $n = 15$. The available quadcopter measurements are given by

$$\boldsymbol{y}_k = [x_k \;\; y_k \;\; z_k \;\; \phi_k \;\; \theta_k \;\; \psi_k \;\; \omega_{\phi,k} \;\; \omega_{\theta,k} \;\; \omega_{\psi,k}]^T \tag{3.71}$$

where the translational velocity and disturbance states are absent. From the theory on EKF in Section 2.2.2, the augmented nonlinear motion model from (3.69) represents the nonlinear function $\boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1|k-1})$, where the related Jacobian matrix $\boldsymbol{F}(\hat{\boldsymbol{x}}_{k-1|k-1})$ is not presented here. As there is no nonlinear measurement model, the related measurement model $\boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1})$ and Jacobian matrix $\boldsymbol{H}(\hat{\boldsymbol{x}}_{k|k-1})$, simply selects the associated measurements $\boldsymbol{y}_k$ from the prediction $\hat{\boldsymbol{x}}_{k|k-1}$ and certainty $\boldsymbol{P}_{k|k-1}$. Figure 3.14 summarizes the observer in a block diagram. $\boldsymbol{r}_k$ is the reference vector for the initial 12 nonlinear states from (3.23), $\boldsymbol{u}_k$ is the electrical motor input from (3.57), $u_{T,k} = \boldsymbol{T}$ and $\boldsymbol{u}_{\tau,k}$ are the physical input signals from (3.9), $\boldsymbol{y}_k^T$ is the measurement vector, and finally where $\hat{\boldsymbol{d}}_k$ is the estimated disturbance vector and $\hat{\boldsymbol{x}}_k$ is the estimated full state feedback. The discrete time nonlinear observer model used for prediction in the EKF is given

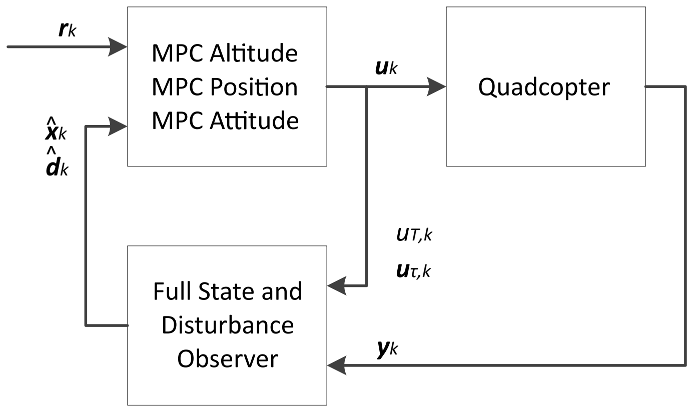


Figure 3.14: Block diagram of the observer with controller.

Due to hardware limitations, the measurement of position is received at a different sampling rate than the controller and observer. Whenever the position measurement at $t = k$ has not been deliverred, the innovation of update step of the EKF from (2.9) is set to zero as of (2.10), such that the observer position output is not updated with the old measurements, hence purely based on the prediction using the nonlinear model.

3.4.2 Decoupled Observer

Analysis of the full state and disturbance observer from (3.69), shows that it is necessary to compute a 9x9 $\boldsymbol{S}_k^{-1}$ matrix inverse during the EKF update step from (2.9). With respect to the implemented solution and the real time performance and computational demand, computing such an inverse matrix online is very expensive. An alternative to computing $\boldsymbol{S}_k^{-1}$ online is to produce a symbolic inverse matrix version offline and replace all the symbolic variables with numerical values online. Due to lack of necessary memory, the symbolic version of the 9x9 $\boldsymbol{S}_k^{-1}$ was not possible to compute. By decoupling the full 15 state observer from (3.69) in to two separate observers; one for the position and disturbance states and one for the attitude states, the 9x9 $\boldsymbol{S}_k^{-1}$ is reduced to two smaller 6x6 and 3x3 matrices. The necessary memory needed in order to compute the symbolic versions of these two matrices is available. The first decoupled observer estimates the attitude motion states $\hat{\boldsymbol{x}}_{att,k} = [\hat{\phi}_k\ \hat{\theta}_k\ \hat{\psi}_k\ \hat{\omega}_{\phi,k}\ \hat{\omega}_{\theta,k}\ \hat{\omega}_{\psi,k}]^T$ from (3.69). These states are not coupled to the remaining position and disturbance states. The attitude observer has the number of states $n_{att} = 6$ with full state measurements $p_{att} = 6$, hence the observer is used to improve the pure measurements with the nonlinear motion model. The decoupled attitude observer model is given by

$$\begin{aligned}
\hat{\phi}_{k+1} &= \hat{\phi}_k + \Delta t\left(\hat{\omega}_{\phi,k} + \hat{\omega}_{\theta,k}\sin\hat{\phi}_k\tan\hat{\theta}_k + \hat{\omega}_{\psi,k}\cos\hat{\phi}_k\tan\hat{\theta}_k\right)\\
\hat{\theta}_{k+1} &= \hat{\theta}_k + \Delta t\left(\hat{\omega}_{\theta,k}\cos\hat{\phi}_k - \hat{\omega}_{\psi,k}\sin\hat{\phi}_k\right)\\
\hat{\psi}_{k+1} &= \hat{\psi}_k + \Delta t\left(\frac{\sin\hat{\phi}_k}{\cos\hat{\theta}_k}\hat{\omega}_{\theta,k} + \frac{\cos\hat{\phi}_k}{\cos\hat{\theta}_k}\hat{\omega}_{\psi,k}\right)\\
\hat{\omega}_{\phi,k+1} &= \hat{\omega}_{\phi,k} + \Delta t\left(\frac{1}{I_{xx}}\tau_{\phi,k} - \frac{I_{yy}-I_{zz}}{I_{xx}}\hat{\omega}_{\theta,k}\hat{\omega}_{\psi,k}\right)\\
\hat{\omega}_{\theta,k+1} &= \hat{\omega}_{\theta,k} + \Delta t\left(\frac{1}{I_{yy}}\tau_{\theta,k} - \frac{I_{zz}-I_{xx}}{I_{yy}}\hat{\omega}_{\phi,k}\hat{\omega}_{\psi,k}\right)\\
\hat{\omega}_{\psi,k+1} &= \hat{\omega}_{\psi,k} + \Delta t\left(\frac{1}{I_{zz}}\tau_{\psi,k} - \frac{I_{xx}-I_{yy}}{I_{zz}}\hat{\omega}_{\phi,k}\hat{\omega}_{\theta,k}\right)
\end{aligned}\tag{3.72}$$

where the associated state and measurement vectors are given by

$$\hat{\boldsymbol{x}}_{att,k} = [\hat{\phi}_k\ \ \hat{\theta}_k\ \ \hat{\psi}_k\ \ \hat{\omega}_{\phi,k}\ \ \hat{\omega}_{\theta,k}\ \ \hat{\omega}_{\psi,k}]^T \qquad \boldsymbol{y}_{att,k} = [\phi_k\ \ \theta_k\ \ \psi_k\ \ \omega_{\phi,k}\ \ \omega_{\theta,k}\ \ \omega_{\psi,k}]^T \tag{3.73}$$

The second decoupled observer, estimates the position states $\hat{\boldsymbol{x}}_{pos} = [\hat{x}\ \hat{y}\ \hat{z}\ \hat{v}_x\ \hat{v}_y\ \hat{v}_z]^T$ and the disturbance states $\hat{\boldsymbol{d}} = [\hat{d}_x\ \hat{d}_y\ \hat{d}_z]^T$. The resulting attitude estimations from the first observer, are passed as parameters $\hat{\boldsymbol{\theta}}_k = [\hat{\phi}_k\ \hat{\theta}_k\ \hat{\psi}_k]^T$ to the second observer, making the motion model time varying. The attitude observer has the number of states $n_{pos} = 9$ with full state measurements $p_{pos} = 3$, hence the observer is used to estimate the non measured states and unknown disturbances. The decoupled position and disturbance observer model is given by

$$\begin{aligned}
\hat{x}_{k+1} &= \hat{x}_k + \Delta t\, \hat{v}_{x,k} \\
\hat{y}_{k+1} &= \hat{y}_k + \Delta t\, \hat{v}_{y,k} \\
\hat{z}_{k+1} &= \hat{z}_k + \Delta t\, \hat{v}_{z,k} \\
\hat{v}_{x,k+1} &= \hat{v}_{x,k} + \Delta t \left(-\frac{k_d}{m}\hat{v}_{x,k} + \frac{1}{m}\left(\sin\hat{\psi}_k \sin\hat{\phi}_k + \cos\hat{\psi}_k \cos\hat{\phi}_k \sin\hat{\theta}_k\right) T_k + \hat{\hat{d}}_{x,k} \right) \\
\hat{v}_{y,k+1} &= \hat{v}_{y,k} + \Delta t \left(-\frac{k_d}{m}\hat{v}_{y,k} + \frac{1}{m}\left(\cos\hat{\phi}_k \sin\hat{\psi}_k \sin\hat{\theta}_k - \cos\hat{\psi}_k \sin\hat{\phi}_k\right) T_k + \hat{\hat{d}}_{y,k} \right) \\
\hat{v}_{z,k+1} &= \hat{v}_{z,k} + \Delta t \left(-\frac{k_d}{m}\hat{v}_{z,k} + \frac{1}{m}\left(\cos\hat{\theta}_k \cos\hat{\phi}_k\right) T_k + \hat{\hat{d}}_{z,k} \right) \\
\hat{\hat{d}}_{x,k+1} &= \hat{\hat{d}}_{x,k} \\
\hat{\hat{d}}_{y,k+1} &= \hat{\hat{d}}_{y,k} \\
\hat{\hat{d}}_{z,k+1} &= \hat{\hat{d}}_{z,k}
\end{aligned} \tag{3.74}$$

where the state and measurement vectors is given by

$$\hat{\boldsymbol{x}}_{pos,k} = [\hat{x}_k \;\; \hat{y}_k \;\; \hat{z}_k \;\; \hat{v}_{x,k} \;\; \hat{v}_{y,k} \;\; \hat{v}_{z,k} \;\; \hat{\hat{d}}_{x,k} \;\; \hat{\hat{d}}_{y,k} \;\; \hat{\hat{d}}_{z,k}]^T \qquad \boldsymbol{y}_{pos,k} = [x_k \; y_k \; z_k]^T \tag{3.75}$$

Figure 3.15 summarizes the decoupled observer in a block diagram. $\boldsymbol{r}$ is the reference vector for the initial 12 nonlinear states from (3.23), $\boldsymbol{u}_k$ is the electrical motor input from (3.57), $u_{T,k} = \boldsymbol{T}$ and $\boldsymbol{u}_{\tau,k}$ are the physical input signals from (3.9), $\boldsymbol{y}_{att,k}$ and $\boldsymbol{y}_{pos,k}$ are the measurement vectors, $\hat{\boldsymbol{\theta}}_k$ are the decoupled attitude state estimates passed as parameters to the second observer, and finally where $\hat{\boldsymbol{d}}_k$ is the estimated disturbance vector and $\hat{\boldsymbol{x}}_k$ is the estimated full state feedback.

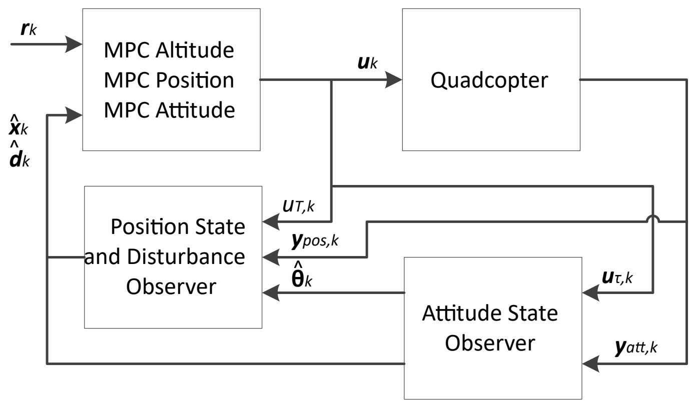


Figure 3.15: Block diagram of the decoupled observer with controller.

3.4.3 Alternative Disturbance Estimation

Ideally the quadcopter would have evenly distributed mass with aligned inertias I_{xx} and I_{yy}, perfectly synchronized motor angular velocities, perfectly specified model parameters such as lift k, motor c_m and drag b constants, and accurate, high sampled attitude and position measurements. This is not the case for the real hardware, where the mass is unevenly distributed, model parameters are not specified by manufacturer but manually identified, attitude measurements are results of estimation from sensor fusion, and where the local positioning system is based on low cost hardware where sampling frequency is not matching the sampling frequency of the online observer. By analysing the complete observer model from (3.69), augmenting the model with inertia estimation could potentially improve the model accuracy online, resulting in better model predictions within the attitude controller. However, the inertias are coupled between each axis, and represented as inverse functions. The risk of observer instability is high as singular solutions can occur when inertia estimations approach zero.

3.4.4 Observer Sampling Time

The system analysis with respect to sampling time for the observer is done similar to the controller sampling time in 3.2.7. The only difference is that since the EKF based observer linearizes the model at every sampling instance, the complete nonlinear observer model from (3.69) is analysed around multiple equilibrium points. By stepping through a range of initial input signals, velocities, angles and angular velocities, a range of sampling frequencies can be found, where the largest result ends up being the selected observer frequency. The range of initial values are selected around the expected operating points, where angles go from -15 to 15 degrees, angular and translational velocities from -0.2 to 0.2 rad/s and -1 to 1 m/s, and PWM input signals from 0-100 %. Figure 3.16 presents the frequency response of the observer model with initial conditions giving largest frequency. This maximum frequency was found around maximum initial condition for PWM input and both minimum and maximum conditions for the angular and translational motion, hence the fast quadcopter moves, the faster the observer need to sample. Table 3.6 presents the frequency with the associated recommended sampling times.

Table 3.6: Minimum and recommended observer frequencies and sampling time.

	$f_{s,min}$ [rad/s]	$f_{s,rec}$ [Hz]	T_s [s]
Observer	8.5600	13.6000	0.0735

Similar to the controller sampling time, the actual sampling time of the implemented observer runs at a considerably faster rate than the analysed recommendation. The implemented observer runs in a real time loop together with orientation estimation algorithm. In order to achieve convergence of the sensor fusion filter and the EKF based observer, the sampling time is set to $T_s = 0.01$ s, where the frequency is $f_s = 100$ Hz.

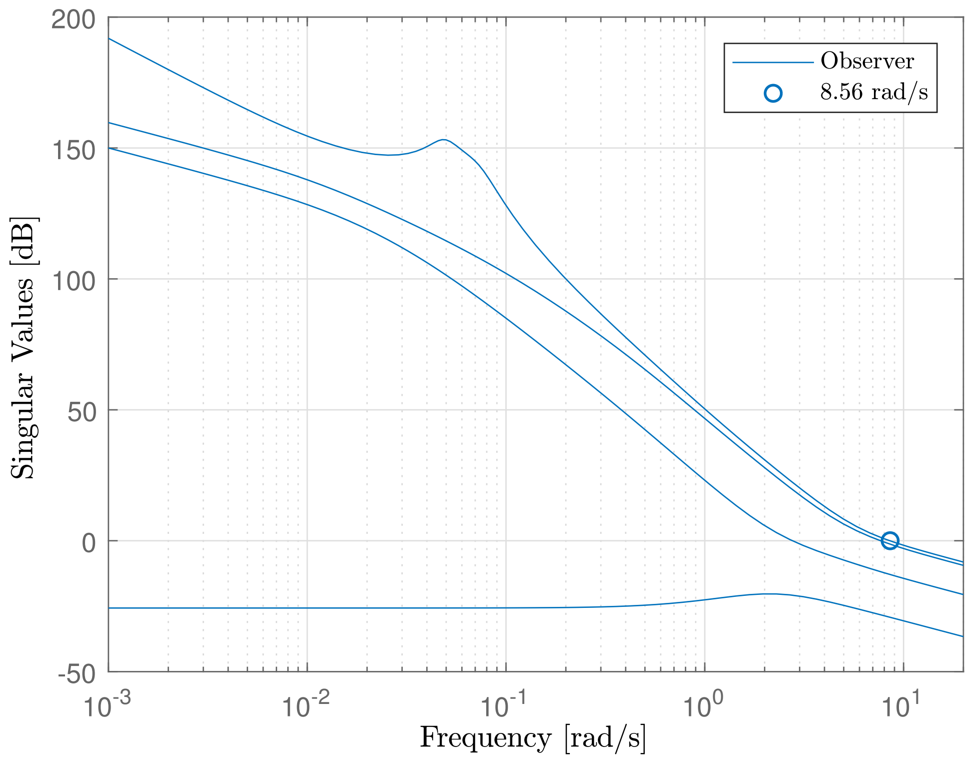


Figure 3.16: Singular values of the observer model.

3.5 Sensor Fusion, Filtering and Orientation Estimation

This section describes the IMU based sensor fusion algorithm used to estimate orientation of the quadcopter. It presents a raw sensor measurement analysis prior to giving results on orientation estimation. It also describes how physical quadcopter frame vibration caused by the spinning motors, makes the orientation estimation drift, and a mechanical dampening solution to solve this issue.

3.5.1 IMU Sensor

Orientation estimation is typically based on combining information from a gyroscope, accelerometer and a magnetometer in order get steady and non-drifting attitude angles ϕ, θ and ψ. The sensor used is the *Inertial Measurement Unit (IMU)* TDK InvenSense MPU9250, which has the three mentioned sensors, including a temperature sensor. The gyroscope provides relative raw angular velocity body frame, the accelerometer provides absolute raw acceleration applied to the sensor in the world frame and the magnetometer measures the surrounding magnetic field. All measurements are related to the x, y and z axes of the sensor (reference) frame. Before proceeding on to the details around the particular orientation filter used, the sensors are analysed with respect to raw measurement noise; mean and variance. Figures 3.17 and 3.18 presents raw measurements from the accelerometer, magnetometer and gyroscope, and Table 3.7 presents the variances and mean values for all axes.

Table 3.7: Mean and variance of raw gyroscope, accelerometer and magnetometer measurements.

	Mean	Variance
g_x	-0.000851	3.722076e-06
g_y	0.000795	8.430107e-06
g_z	0.000706	8.312619e-06
a_x	-0.145991	2.862287e-06
a_y	-0.097737	2.929547e-06
a_z	0.9031941	3.986317e-06
m_x	-3.052830	0.587330
m_y	-8.181940	0.592540
m_z	6.679058	0.551842

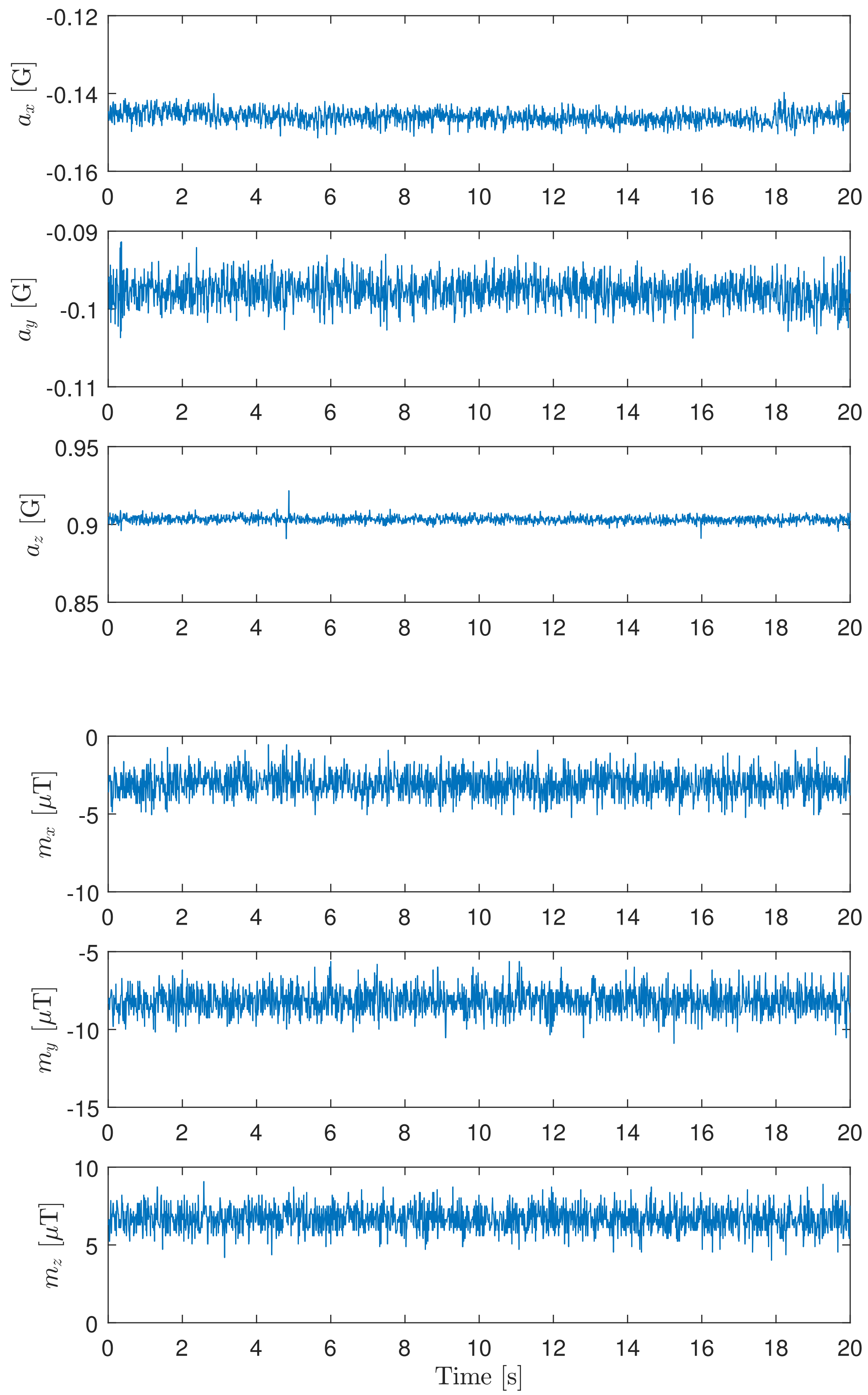


Figure 3.17: Raw accelerometer and magnetometer measurements with gravity unit G and magnetic unit μT.

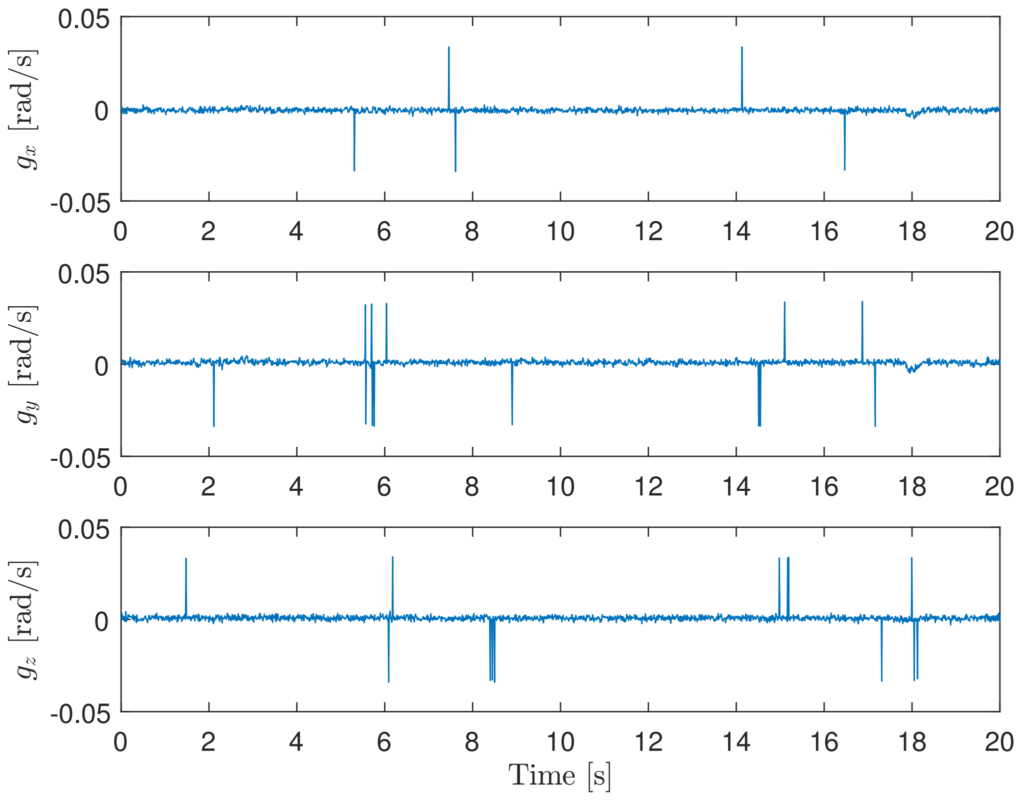


Figure 3.18: Raw gyroscope measurements with unit rad/s.

All of the measurement are simultaneously collected with the sensor board mounted on the quadcopter. The quadcopter is located on a table in a stationary horizontal position. The gyroscope measurement is expected to have a mean ≈ 0. The collected small mean and variance indicates that the gyroscope measurement is of good quality. If heading ψ is only estimated based on integrating the angular velocity around the z axis, a bias of $\bar{g}_z = 0.000706$ rad/s will cause slow accumulating drift in ψ. More on how to compensate for such drift in Section 3.5.2. The accelerometer measurement is expected to have a mean value ≈ 0 in x and y axes and ≈ 1 in z axis (one gravitational force unit). Again, the collected small mean and variance indicates that the accelerometer measurement is of good quality. The small biases indicates that the sensor board is not perfectly horizontally aligned, which results in that a small proportion of the gravitational force unit in the z axis becomes distributed in x and y axes. The magnetometer measurement is expected to have a mean approximately equal to the surrounding field, which is unknown. The collected data has a larger variance which indicates that the measurement is subject to noise. When using magnetometers indoor, the surrounding environment must be taken in to account as it is subject to magnetic fields such as electrical equipment.

3.5.2 Sensor fusion and Orientation Estimation

The implemented orientation estimation is based on using only the gyroscope and accelerometer. The non-filtered sensor measurements are passed directly to an open source, computational inexpensive IMU algorithm, designed by Sebastian Madgwick from [15, 16]. The algorithm derivation is not presented in detail here and can be found in the citations, however the main steps are still covered in order to provide an understanding on how the sensor fusion is used to get desired orientation estimations. Figure 3.19 presents a block diagram of the algorithm flow.

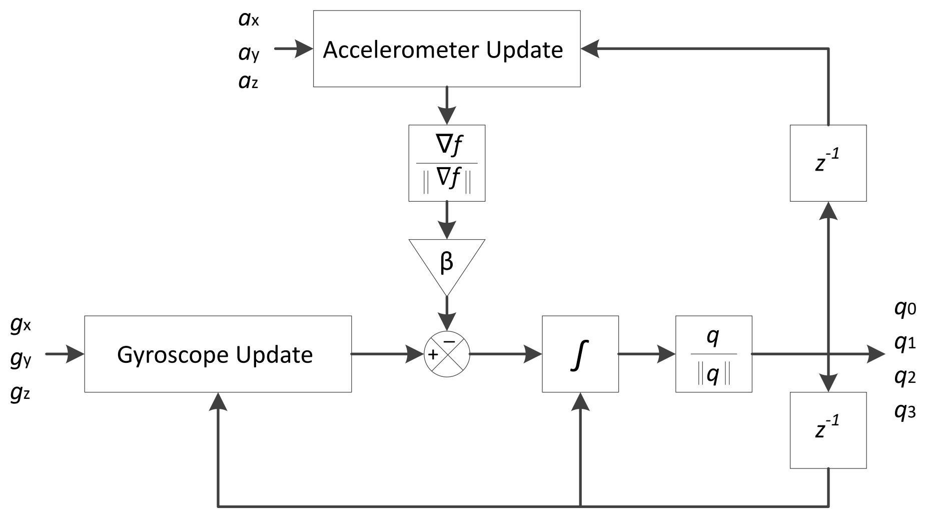


Figure 3.19: Orientation estimation IMU filter block diagram.

The algorithm returns a quaternion representation of orientation given by $\boldsymbol{q} = [q_0 \; q_1 \; q_2 \; q_3]^T$. The three axes gyroscope measurements are passed in to the gyroscope update, where the quaternion representation of the angular rate of change is given by

$$ {}^S_E\dot{\boldsymbol{q}}_{\omega,t} = \frac{1}{2} {}^S_E\hat{\boldsymbol{q}}_{t-1} \otimes {}^S\boldsymbol{\omega}_t \tag{3.76}$$

where ${}^S_E$ denotes earth relative to sensor frame, the gyroscope measurements are arranged in to $\boldsymbol{\omega} = [0 \; g_x \; g_y \; g_z]^T$, $\otimes$ is a quaternion product and $\hat{}$ represents a normalized vector. The orientation is achieved by integrating (3.76) given by the discrete time equation

$$ {}^S_E\boldsymbol{q}_{\omega,t} = {}^S_E\hat{\boldsymbol{q}}_{t-1} + {}^S_E\dot{\boldsymbol{q}}_{\omega,t}\Delta t \tag{3.77}$$

In theory, the orientation is now obtained, however the present gyroscope noise will accumulate over integration and make the attitude drift as time goes. The accelerometer is used to provide an absolute measurement of the earth gravitational force in order to compensate for the drifting angles around x and y axes. The quaternion orientation estimation based on the accelerometer is found by using the computational inexpensive gradient descent method in order to solve an optimization problem defined such that a complete solution for the quaternion representation is found among infinite solutions based on the raw accelerometer measurements. The objective function is based on a simplification assuming that there is only the gravitational force working on the sensor. The resulting quaternion orientation estimation based on the accelerometer is given by

$$ {}^{S}_{E}\boldsymbol{q}_{\nabla,t} = {}^{S}_{E}\hat{\boldsymbol{q}}_{t-1} - \mu_t \frac{\nabla \boldsymbol{f}}{\|\nabla \boldsymbol{f}\|} \tag{3.78} $$

where the $\nabla \boldsymbol{f}$ is the gradient descent objective function based on the accelerometer measurements and μ is the gradient descent step size. In order to achieve the complete orientation estimation, the two quaternion estimations are combined as of Figure 3.19 and given by

$$ {}^{S}_{E}\boldsymbol{q}_{t} = \gamma_t {}^{S}_{E}\boldsymbol{q}_{\nabla,t} + (1-\gamma_t){}^{S}_{E}\boldsymbol{q}_{\omega,t} \tag{3.79} $$

where γ_t is the weight factor between the orientation estimation from the gyroscope and the accelerometer. This factor is expressed as $\gamma_t \approx \frac{\beta \Delta t}{\mu_t}$ which is the optimal value where the divergence of ${}^{S}_{E}\boldsymbol{q}_{\omega}$ is equal to the convergence of ${}^{S}_{E}\boldsymbol{q}_{\nabla}$ with the tuning factor β. By substituting (3.77) and (3.78) in to (3.79), and defining the accelerometer contribution as an estimate of the quaternion error, the complete sensor fusion is simplified and given by

$$ {}^{S}_{E}\boldsymbol{q}_{t} = {}^{S}_{E}\hat{\boldsymbol{q}}_{t-1} + \left({}^{S}_{E}\dot{\boldsymbol{q}}_{\omega,t} - \beta \frac{\nabla \boldsymbol{f}}{\|\nabla \boldsymbol{f}\|} \right) \Delta t \tag{3.80} $$

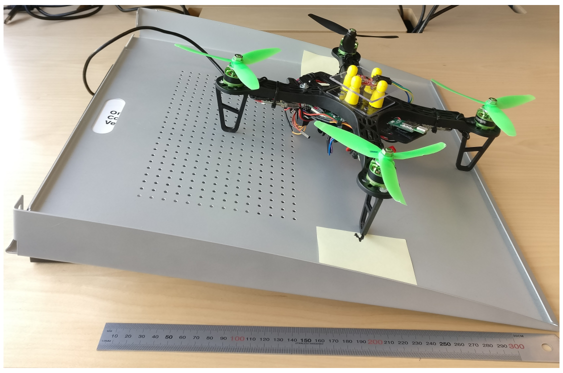
The adjustable factor β is used to tune how much of the gyroscope measurement error is to be removed by the estimated error from the accelerometer. In the practical scenario, β is a trade-off between a fast converging filter that quickly adapts to changes in orientation, or a slower converging filter which has good outlier detection for bad measurement data provided by the gyroscope and the accelerometer. The smaller value of β, the slower the filter is. For the sensor fusion verification presented in Section 3.5.1, the tuning factor of $\beta = 0.05$ turned out to give satisfying results with respect to convergence of true orientation and outlier detection during quadcopter translational motion. The quaternion representation of the orientation of the sensor frame relative to the earth is finally given by the conjugate

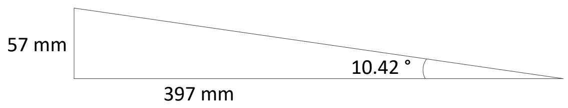
$$ {}^{E}_{S}\hat{\boldsymbol{q}}_{t} = {}^{S}_{E}\hat{\boldsymbol{q}}^{*}_{t} = [q_0 \;\; -q_1 \;\; -q_2 \;\; -q_3]^T \tag{3.81} $$

The accelerometer is only going to compensate for the error in estimation of ϕ and θ. In order to achieve non drifting ψ, a magnetometer is typically used. Sebastian Madgwick does introduce an *Attitude and Heading Reference Systems (AHRS)* filter, where the magnetometer is used as a reference for true heading. However, a combination of changing indoor environment magnetic field around the sensor board and noisy variance given in Table 3.7, the AHRS filter did not give satisfying compensation, hence the IMU filter is part of the implemented solution. In order to minimize the drift, data is collected during initialization of the filter such that the present bias can be removed during run time. An alternative to the AHRS filter, is to use an EKF approach on estimating the orientation. An EKF orientation estimation was implemented and benchmarked on computational effort with respect to the IMU filter. The IMU filter turned out to outperform up to three times faster than the EKF based filter, regardless of the hardware.

3.5.3 Orientation Estimation Verification

The resulting orientation estimation is verified by rotating the quadcopter around each axis to predefined angles. In order to keep consistency between rotation, a fixed rig is preset at an angle of 10.42°. Figure 3.20 illustrates the actual physical test rig with the associated calibration measurements and the test angle.


(a) Actual test rig



(b) Test rig measurements

Figure 3.20: Orientation estimation test rig.

The quadcopter is rotated and placed back and forth between the horizontal plane next to the rig and on to the actual rig such that most angle scenarios are covered. The complete test procedure is given by:

1. Initialize at ϕ, θ and ψ equal to zero degrees,
2. Rotate $\theta = -10.41°$ and back to $\theta = 0°$,
3. Rotate $\psi = 90°$,
4. Rotate $\phi = -10.41°$ and back to $\phi = 0°$,
5. Rotate $\psi = 180°$,
6. Rotate $\theta = 10.41°$ and back to $\theta = 0°$,
7. Rotate back $\psi = 90°$, $\psi = 0°$ and then to $\psi = -90°$,
8. Rotate $\phi = 10.41°$ and back to $\phi = 0°$,
9. Rotate $\psi = -180°$,
10. Rotate $\theta = 10.41°$ and back to $\theta = 0°$,
11. Finally rotate back $\psi = -90°$ and $\psi = 0°$.

Figure 3.21 presents the results from the described test procedure. The test results shows that the desired angle of $\approx 10.41°$ is reached for both positive and negative rotations in ϕ and θ. Heading estimation within the range of $\psi = [-180°, 180°]$ are good. In between 50 to 70 seconds, the sign of ψ is flipped as the orientation goes above the desired angle $\psi = 180°$. This does not affect the tilt angles ϕ and θ. Notice that when $\theta \approx 10.41°$ at 60 and 135 seconds, the heading is $\psi \approx 180°$ (without the flipped sign) and $\psi \approx -180°$. This is expected as the quadcopter frame is rotated in the same direction for both ϕ and θ regardless of whether the heading is rotated $\pm180°$.

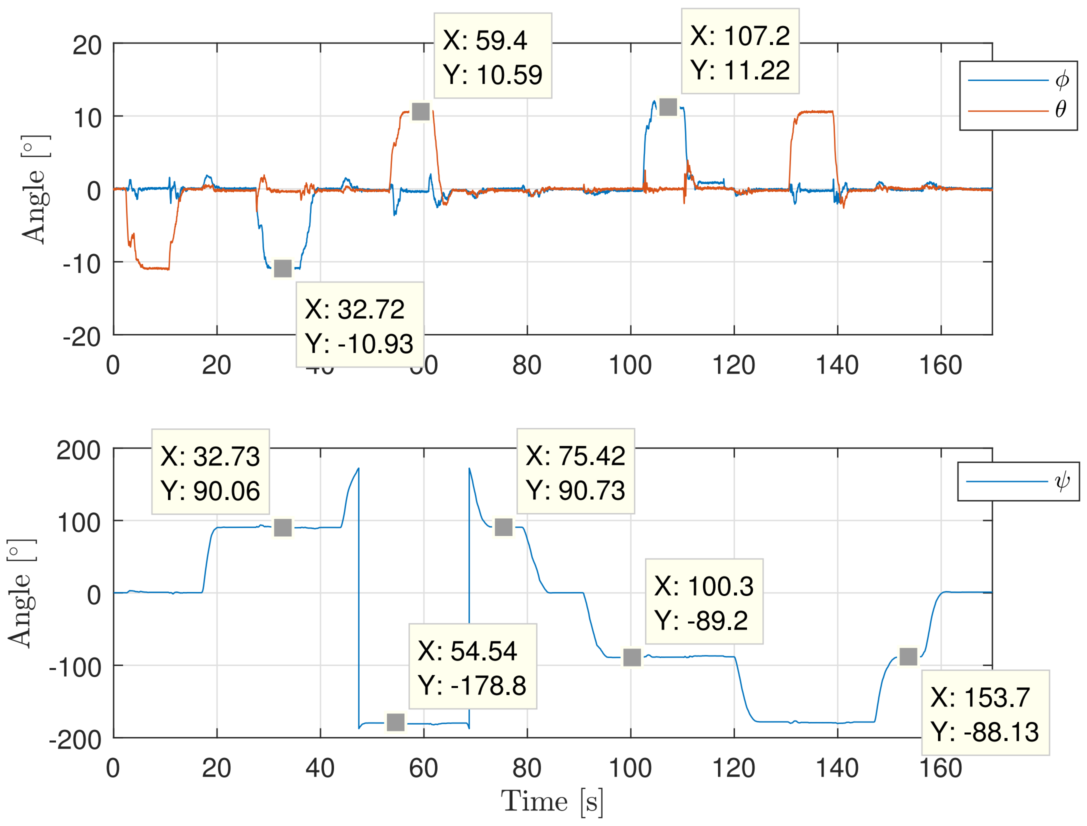


Figure 3.21: Orientation estimation verification results.

The present ψ drift is verified by collecting data for five minutes. Figure 3.22 presents the results, where approximately -0.00385°$/s$ will be the heading drift whenever the quadcopter is stationary. The drift can vary depending on whether the collected gyroscope bias in the z axis actually represents the present angular velocity bias such that raw measurement is compensated correctly prior to being passed in the IMU algorithm. Without the bias compensation, the drift would be considerably larger.

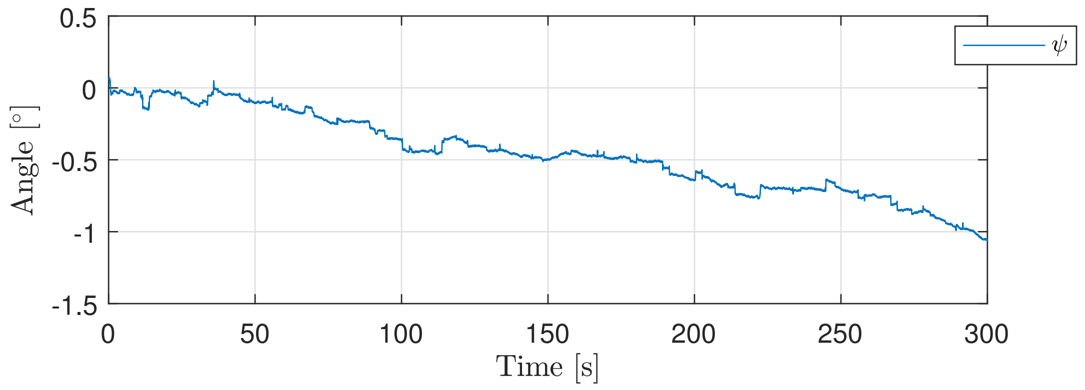


Figure 3.22: Heading ψ drift over time.

3.5.4 Mechanical Damping

UAVs are subject to vibration when the motors are spinning. This has resulted into a random drift in angle estimation from the sensor fusion algorithm and made sensor measurements unworthy. As a practical solution, a mechanical damping is used. It consists of two rigid plastic plates where the only connections between them are four ear plugs. One of the plates is mounted on the quadcopter frame stiffly and the sensor board is mounted on the other plate stiffly and the ear plugs in between are meant to capture the vibrations. Figure 3.23 presents the design.

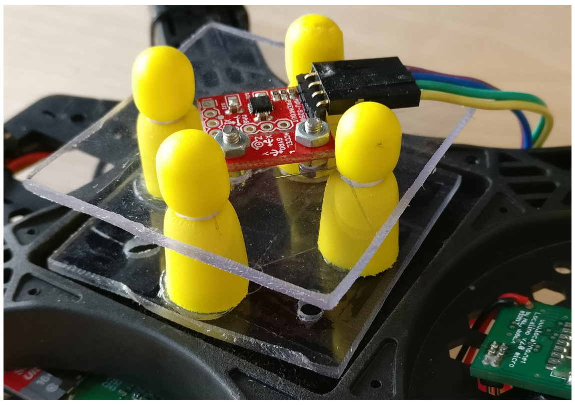


Figure 3.23: Mechanical damping of IMU sensors using earplug foam material.

To evaluate the effect of them, a comparison is done between two scenarios of with and without the ear plugs in both time and frequency domains. Figure 3.24 illustrates two different runs, without and with mechanical damping. With the quadcopter constrained in a horizontal position in a rig with all four motors spinning equally at $u_i = 90\%$, the attitude estimation drifts without dampening and stays immune against the vibrations with dampening.

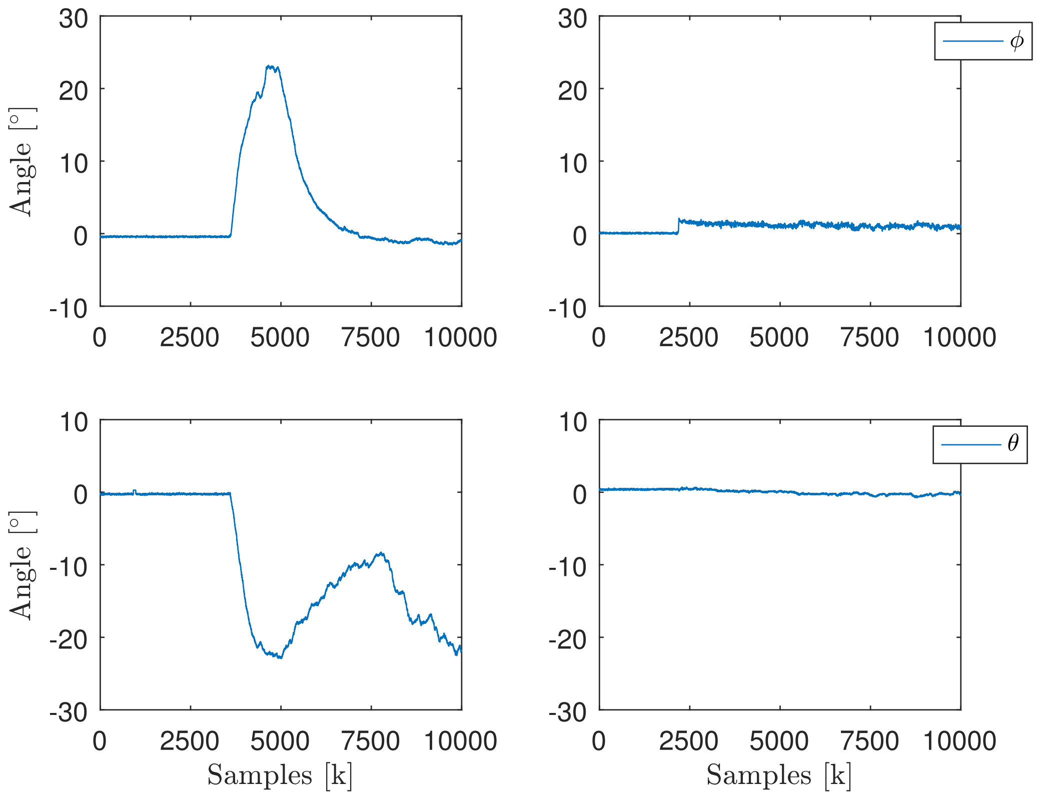


Figure 3.24: Tilt estimation comparison; without mechanical damping (left) and with mechanical damping (right).

Figures 3.25 and 3.26 present raw measurements from the IMU sensor for both cases of without and with mechanical damping for all three axes in the time domain. The mean values are also presented. On the accelerometer plots, a clear shift in the mean values for the x and y axes, occurs as soon as the motors start spinning at around sample $k = 3500$ without using damping. When using damping, the mean values are immune to such shift. This is suspected to be the main reason for the unworthy attitude estimations, since the accelerometer cannot accurately compensate for drift from the gyroscope. Table 3.8 presents the variance of each sensor without and with damping. For the accelerometer, the variances of vibration noises would get 177, 9 and $\frac{1}{3}$ times larger in x, y and z axes respectively, compared to using damping. However for the gyroscope, using damping gives equal mean values as without damping, but bigger noise variances of 6, 2 and 8 times larger. Loosely specking, accelerometer is better, gyroscope is worse, but not as much as accelerometer got better in terms of variances. In terms of mean values, while there is not much change for gyroscope, accelerometer shows much better results. This magnifies the importance of using an accelerometer as an absolute measurement in the world frame for correction of drifting attitude estimation using purely the gyroscope which is a measurement relative to the sensor frame. Note that the two set of data with and without damping are separately collected from the system, since the sensor board has to be taken off and changed with respect to damping, hence the starting time of the motors are slightly different between the two scenarios. Also the accelerometer sensor has a saturation of $\pm 2G$.

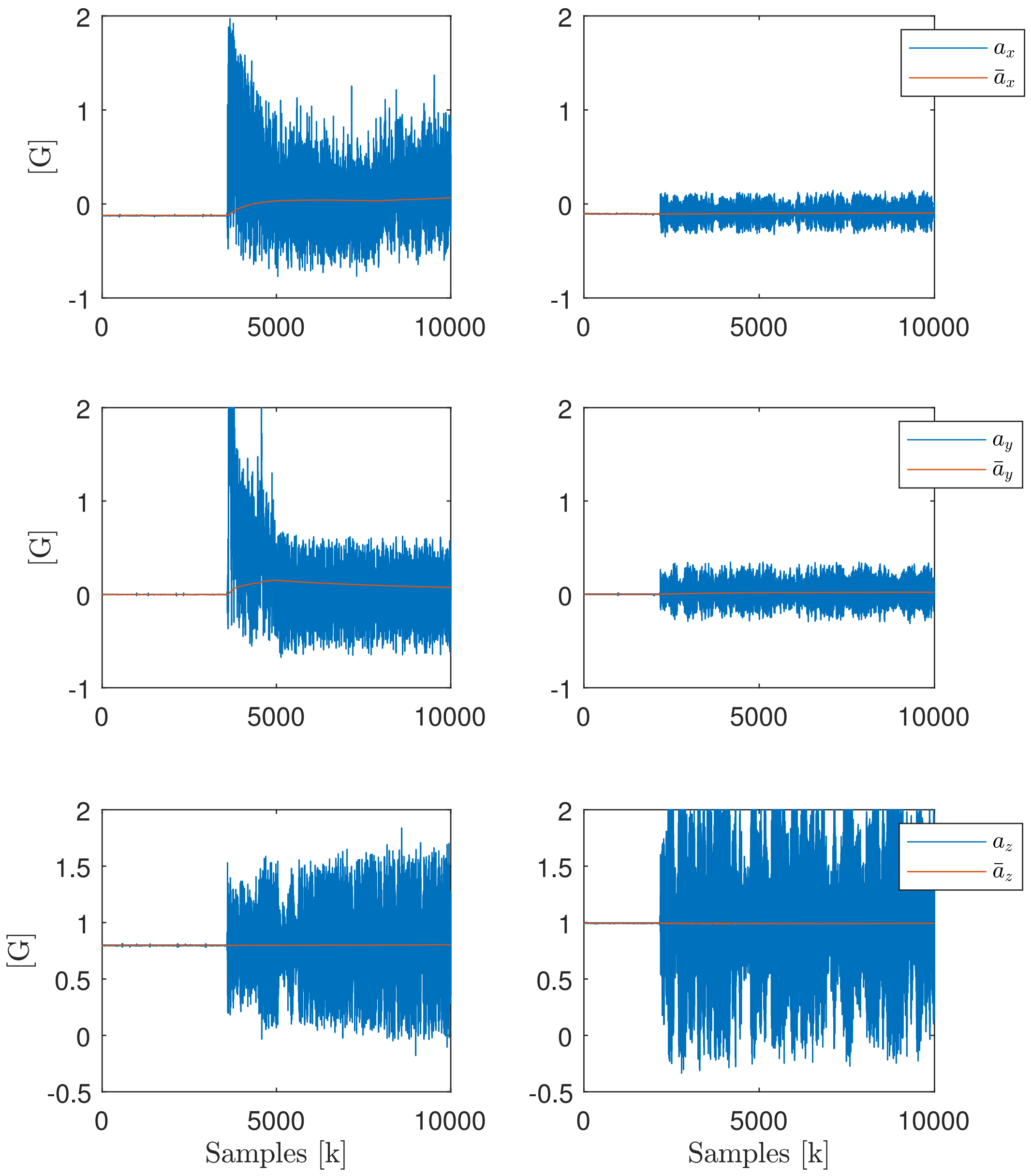


Figure 3.25: Raw accelerometer comparison; without mechanical damping (left) and with mechanical damping (right).

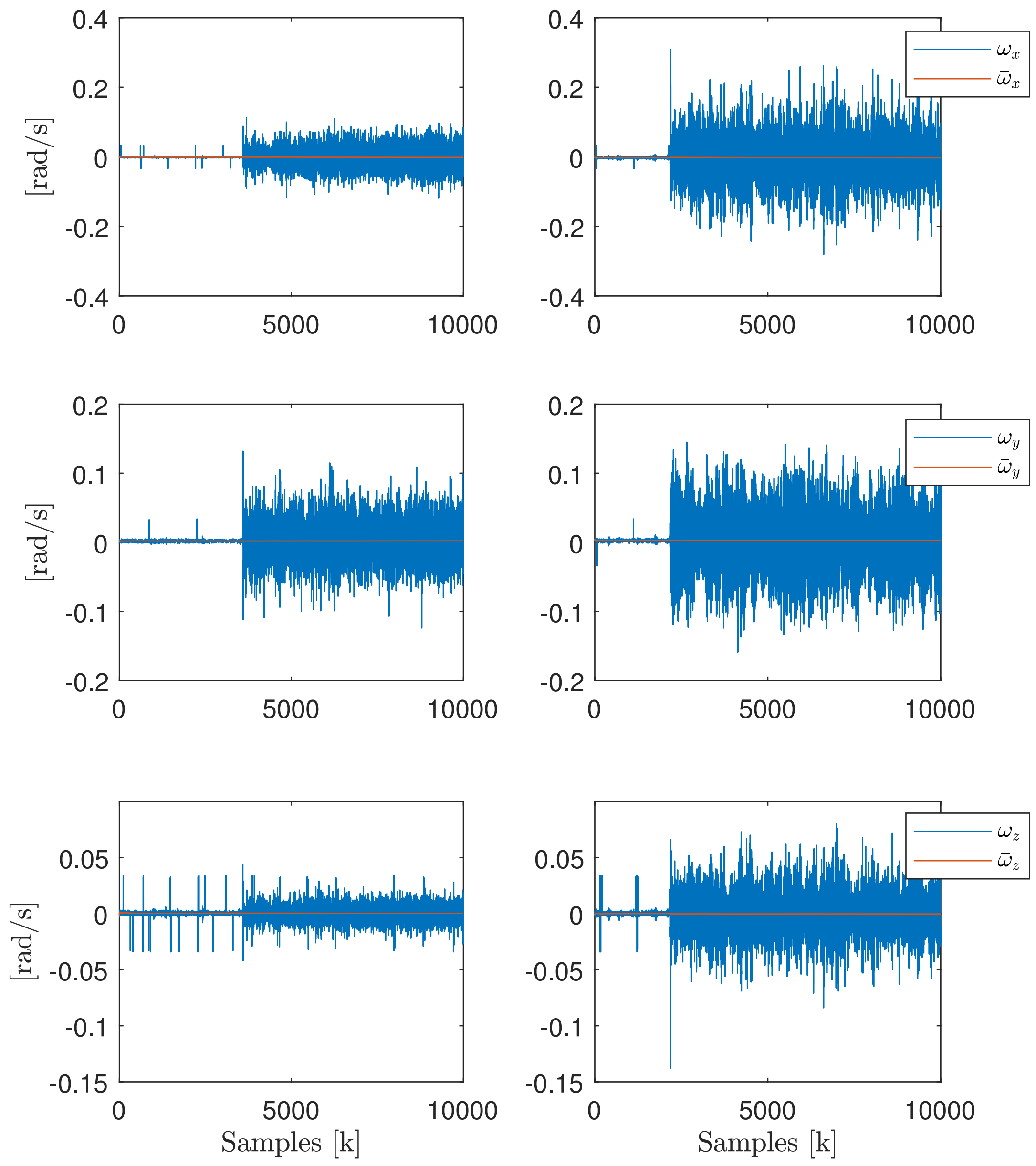


Figure 3.26: Raw gyroscope comparison; without mechanical damping (left) and with mechanical damping (right).

Table 3.8: Variances of raw accelerometer and gyroscope measurements with and without damping.

	Without damping			With damping		
Sensor	σ_x^2	σ_y^2	σ_z^2	σ_x^2	σ_y^2	σ_z^2
Accelerometer [G]	0.12300	0.11500	0.07900	0.00700	0.01300	0.24400
Gyroscope [rad/s]	0.00070	0.00060	0.00004	0.00420	0.00160	0.00033

Figures 3.27 and 3.28 presents the frequency plots of the same raw measurements from the accelerometer and gyroscope. It does not show much improvement for gyroscope using the damping. However for accelerometer, frequencies below 30 dB are considerably filtered using the damping. The filtered frequencies are suspected to be the vibration frequency components causing the problem, as there is no real motion of quadcopter in these tests. Note again that the two sets of data with and without damping are separately collected, hence the ticking frequencies of the data sets are slightly different and that is why one cuts shorter in frequency compared to the other one.

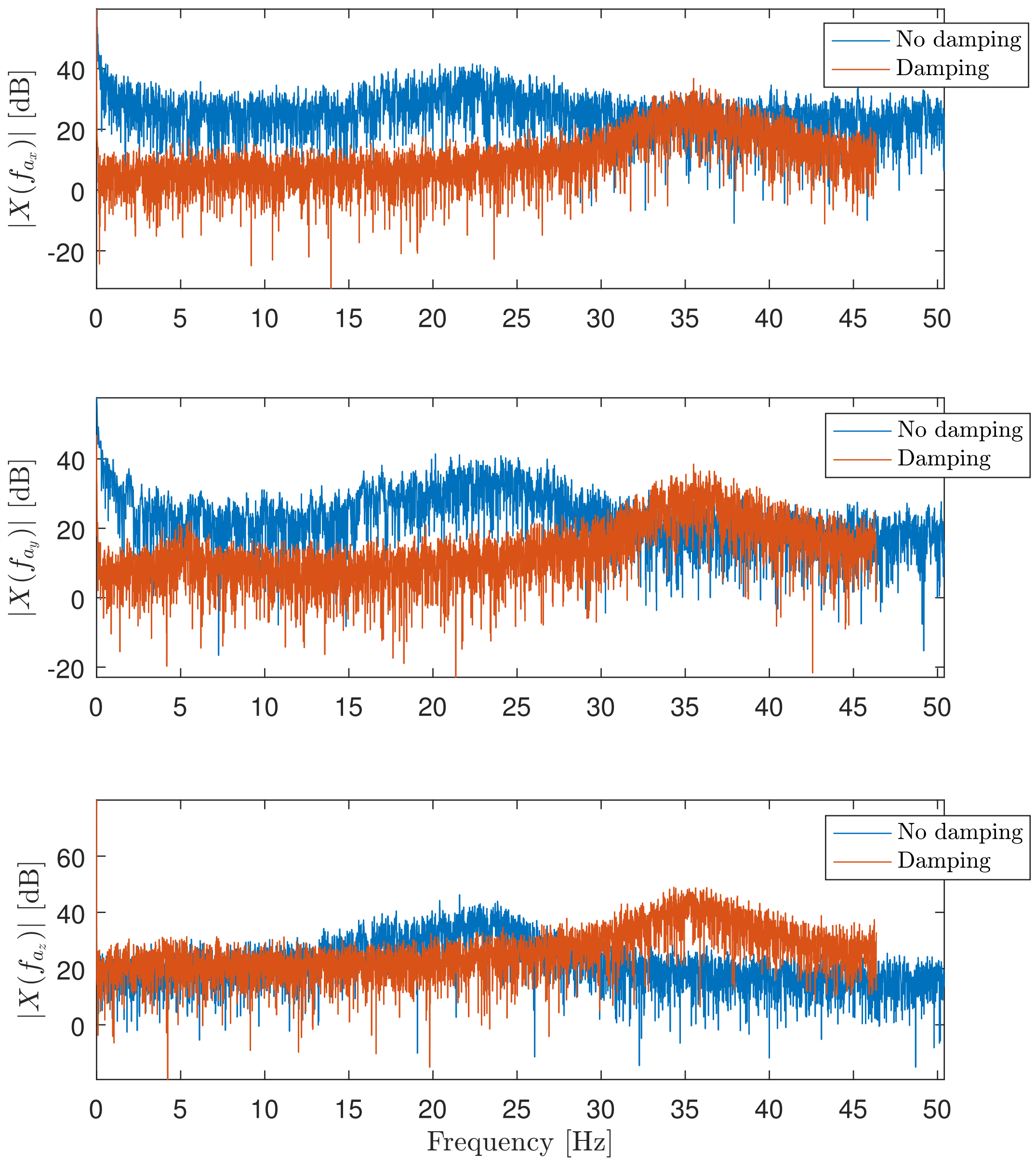


Figure 3.27: Raw accelerometer comparison in frequency domain

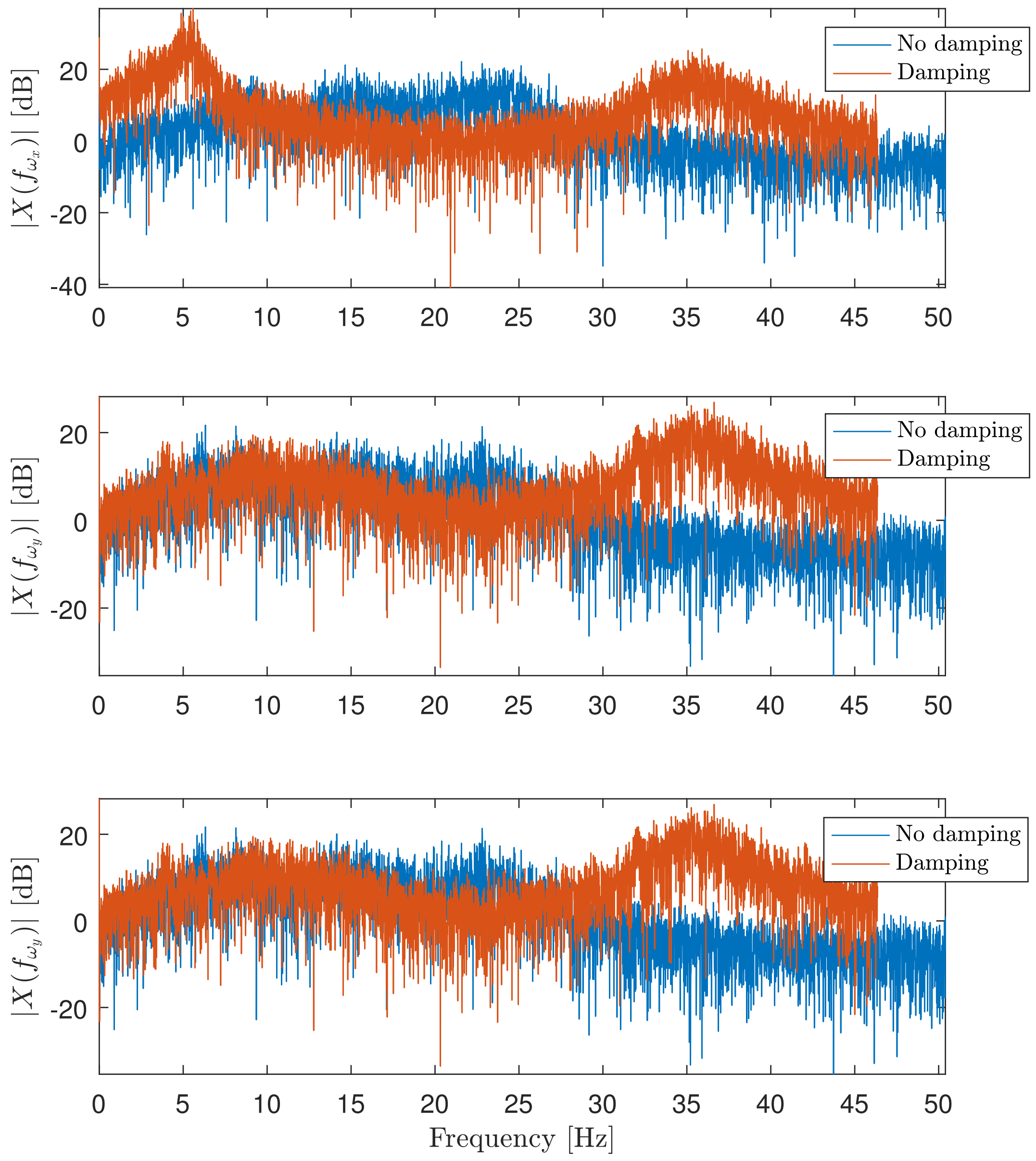


Figure 3.28: Raw gyroscope comparison in frequency domain

3.6 Parameter Identification

This section presents parameter identification of the quadcopter, where each individual parameter given in Section 3.1 are estimated based on raw calculations and collected data.

3.6.1 Mass

The quadcopter mass is the sum of all sub parts. The total mass is weighted using a scale and found to be $m = 0.4234$ kg. Ideally quadcopter would have the mass equally distributed along the structure. This is not case here due to battery and controller placement. However, the centre of mass is placed as low as possible, creating a pendulum effect which forces the quadcopter to always face the propellers upward.

3.6.2 Motor and Lift Constant

The motor constant is necessary in order to accurately translate the electrical motor input PWM u_i to physical motor speed ω_i, for $i = \{1...4\}$. The motor constant is initially identified by collecting data from a tachometer which is connected directly to the motor shaft such that the motor speed is measured and related to the electrical motor input. The physical connection between the tachometer and motor shaft was vulnerable at high motor speed due to vibrations and a slipping effect, hence only motor speeds up to approximately 50 % are investigated. Battery voltage level is crucial as the resulting motor speed will decrease as voltage decreases. As the current increases together with the increases of motor speed, the voltage drops. For the collected data, multiple fixed voltage levels are investigated, where the minimum is 9.0 V and the maximum is 12.6 V. The nominal voltage is considered to be 11.1 V. Figure 3.29 presents the collected motor speed in *Round Per Minute (RPM)* for u_i = 20, 30, 40 and 50 %, together with a linear approximation of the complete range u_i = 0-100 %.
The lift constant is necessary in order to go from motor speed to thrust. The lift constant is identified by placing the quadcopter upside down on top of a high resolution scale. Further, the total mass is set zero such that when the motors are spinning, force will push upwards, creating measurable mass. The total mass M measured is a sum of all four motors, and is mapped to the force f_i created by each individual motor $i = \{1...4\}$ by $f_i = \frac{Mg}{4}$ using the gravitational force g. Figure 3.30 presents the collected total mass for u_i = 20, 30, 40 and 50 %, together with a linear approximation of the complete range u_i = 0-100 %.
The combined relation between the motor and the lift constant, translates the electrical input PWM to motor force as from (3.7). Table 3.9 summarizes the resulting parameters based on the collected data. Ideally, the lift constant is independent of the voltage level as it depends on the the physical propeller design, which does not change. For initial quadcopter flight tests, c_m = 23.0907 and k = 1.0107e-05 at the nominal voltage of 11.1 V was used. The initial estimation of c_m and k showed instability in attitude control with increasing oscillation around the reference point at

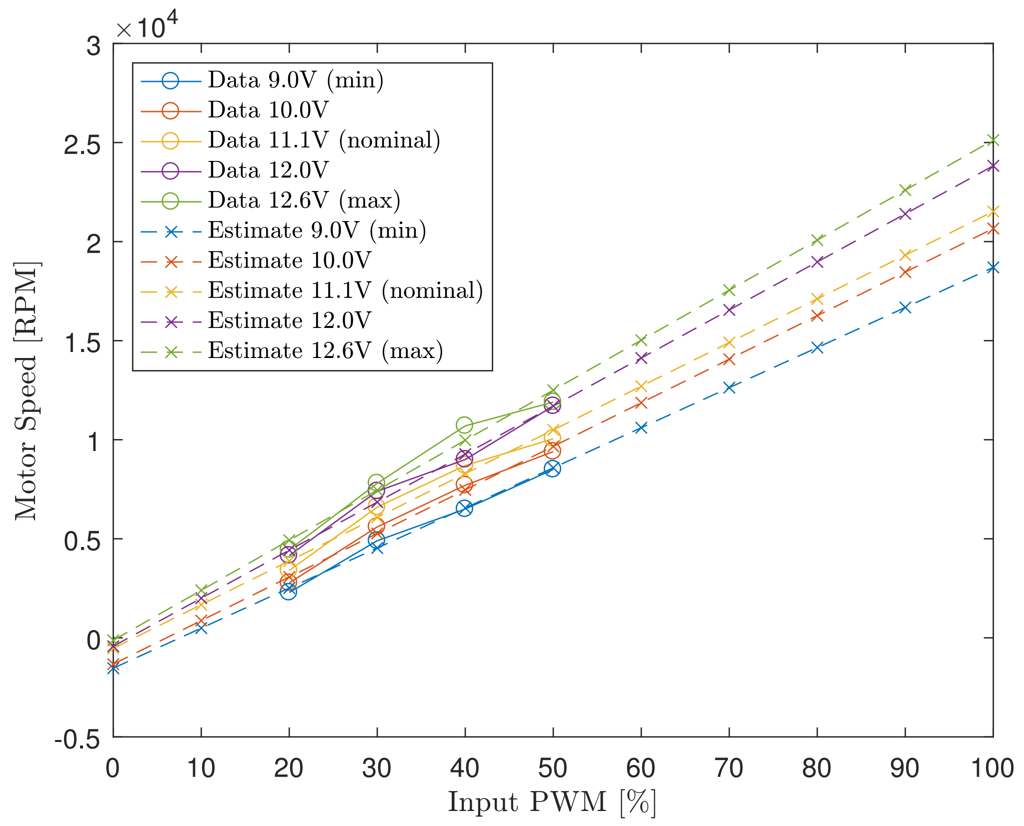


Figure 3.29: Relation between input PWM and resulting motor speed in RPM.

Table 3.9: Estimated motor and lift constants for different voltage levels.

Voltage [V]	c_m [PWM to rad/s]	k [rad/s to N]
9.0	21.1848	7.4528e-06
10.0	23.0279	8.4282e-06
11.1	23.0907	1.0107e-05
12.0	25.3945	1.0464e-05
12.6	26.4417	1.0607e-05

zero degrees. The increasing oscillation indicates that either c_m or k, or both of the parameters are wrong. Based on the vulnerable initial estimation method of c_m, this parameter is approximated again. By using a *Proportional-Integral (PI)* controller for attitude, thrust is applied manually controlled such that the quadcopter stays at a hovering altitude position where the gravitational acceleration is cancelled, where c_m can be estimated based on which electrical input the motors actually need to keep hovering. For this experiment, the unstable motor constant $c_m = 23.0907$, is increased to $c_m = 30$ in order for the attitude PI controller to fly stable around zero degrees in ϕ and θ. Regardless of which c_m or k parameter used, the electrical input signal u_i will always stay the same. From the test $u_i = 56.7062$ % is collected. Further, the new motor constant estimation is found by solving the motion equation

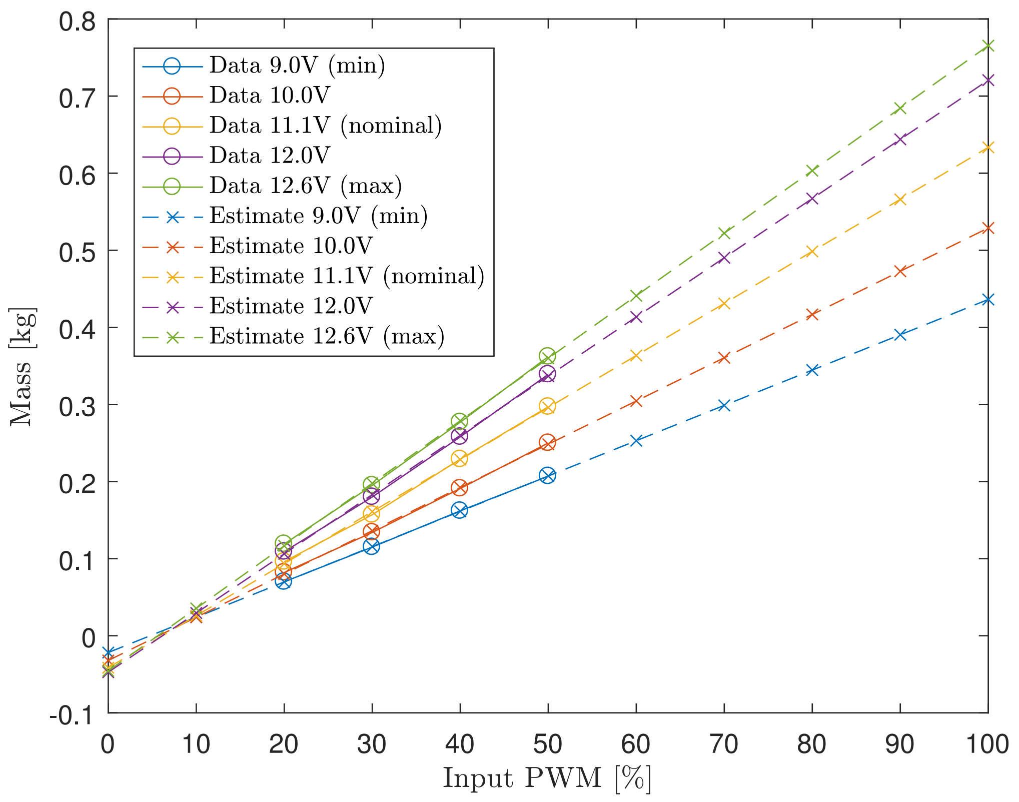


Figure 3.30: Relation between input PWM and resulting total mass produced by spinning motors.

for $\dot{v}_z$ from (3.16) with respect to c_m at the hovering state, given by

$$c_m = \frac{mg}{k\left(u_1^2 + u_2^2 + u_3^2 + u_4^2\right)} \tag{3.82}$$

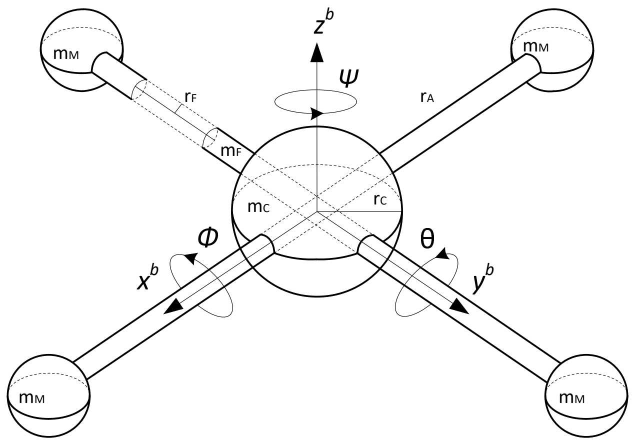
where k = 1.0107e-05 from the initial guess is used. Since k and c_m are in all equations except (3.22) coupled as $k\,c_m$, offset of k compared to the real value would also map onto c_m and the coupled term would be correct. The resulting motor constant is $c_m = 31.9513$. Compared to the initial estimation $c_m = 23.0907$, the new motor constant has an approximately 40 % increase, hence the motors become more dampened, and the reason for why the initial flight was unstable with increasing oscillation.

3.6.3 Drag Constant

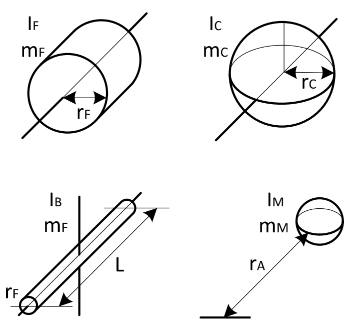
The drag constant b has not been estimated. A value of b = 3.3691e-07 is used during flight. The value is a *rough guess* based on the relationship of approximately $b = \frac{1}{30}k$ between the lift and drag constant from [6].

3.6.4 Inertia

Initial estimations of inertias I_{xx}, I_{yy} and I_{zz} are carried out using basic formulas for moment of inertia on physical systems, with respect to the quadcopter [22]. The quadcopter structure is simplified in to several sub parts. Figure 3.31 illustrates how the initial body frame in Figure 3.1a has reduced to spheres and cylinder shaped rods together with the moment of inertia concept applied.



(a) Simplified quadcopter frame



(b) Inertia concepts

Figure 3.31: Simplified quadcopter frame illustrating momen of inertia estimation.

Figures 3.31a and 3.31b introduce some new notations, where m_F and r_F is the mass and radius of the frame arm, m_C and r_C is the mass and radius of the centre, and m_M and r_A is the motor mass and distance from frame centre to motor. L_f represent the total length between the diagonal motors. The quadcopter inertia I_{xx}, I_{yy} and I_{zz} are then estimated by summarizing the individual frame, centre, body and motor inertia I_F, I_C, I_B and I_M, resulting in

$$I_{xx} = I_{yy} = \underbrace{\frac{1}{2}m_F r_F^2}_{I_F} + \underbrace{\frac{1}{4}m_F r_F^2 + \frac{1}{12}m_F L^2}_{I_B} + \underbrace{\frac{2}{5}m_C r_C^2}_{I_C} + \underbrace{2(m_M r_A^2)}_{I_M} \tag{3.83}$$

$$I_{zz} = \underbrace{2\left(\frac{1}{4}m_F r_F^2 + \frac{1}{12}m_F L^2\right)}_{I_B} + \underbrace{\frac{2}{5}m_C r_C^2}_{I_C} + \underbrace{4(m_M r_A^2)}_{I_M} \tag{3.84}$$

where the different inertia parameters are given by Table 3.10.

Table 3.10: Model parameters.

m_F [kg]	m_C [kg]	m_M [kg]	r_F [m]	r_C [m]	r_A [m]	L_f [m]
0.055	0.269	0.025	0.010	0.035	0.125	0.250

The resulting estimated inertia is given by Table 3.11.
Test results of attitude control with reference point at the origin for both ϕ and θ, shows that the delivered control input to the quadcopter is aggressive causing

Table 3.11: Initial inertia estimations.

I_{xx} [kg/m^2]	I_{yy} [kg/m^2]	I_{zz} [kg/m^2]
0.0012	0.0012	0.0023

unstable oscillation. The MPC has little or no room for tuning using state and input weights, hence the prediction model does not represent the real plant parameters. A second inertia estimation is therefor carried out for ϕ and θ by applying a sine wave shaped input torque signal for τ_ϕ and τ_θ to the quadcopter in open loop. White noise is added on top of the sine wave in order to excite more frequencies of the system. Raw measurement data is collected directly from the gyroscope where I_{xx} and I_{yy} are estimated using *Linear Least Squares (LLS)* . The linearized discrete time versions of Equations (3.20) and (3.21) given by

$$\omega_{\phi,k+1} = \omega_{\phi,k} + \frac{1}{I_{xx}}\Delta t\, \tau_{\phi,k} \tag{3.85}$$

$$\omega_{\theta,k+1} = \omega_{\theta,k} + \frac{1}{I_{yy}}\Delta t\, \tau_{\theta,k} \tag{3.86}$$

are rewritten to match the regressor model used for Linear Least Squares estimation and given by

$$\underbrace{\omega_{\phi,k+1} - \omega_{\phi,k}}_{\boldsymbol{y_\phi}} = \underbrace{\frac{1}{I_{xx}}}_{\boldsymbol{\beta_{xx}}} \underbrace{\Delta t\, \tau_{\phi,k}}_{\boldsymbol{X_\phi}} \tag{3.87}$$

$$\underbrace{\omega_{\theta,k+1} - \omega_{\theta,k}}_{\boldsymbol{y_\theta}} = \underbrace{\frac{1}{I_{xx}}}_{\boldsymbol{\beta_{yy}}} \underbrace{\Delta t\, \tau_{\theta,k}}_{\boldsymbol{X_\theta}} \tag{3.88}$$

where the output and input vectors $\boldsymbol{y}$ and $\boldsymbol{X}$ contain measurements $k = \{1, ..., N - 1\}$. N is total collected samples. The resulting estimation of $\boldsymbol{\beta_{xx}}$ and $\boldsymbol{\beta_{yy}}$ is found by solving

$$\hat{\beta}_{xx} = (\boldsymbol{X}_\phi^T \boldsymbol{X}_\phi)^{-1} \boldsymbol{X}_\phi^T \boldsymbol{y}_\phi \tag{3.89}$$

$$\hat{\beta}_{yy} = (\boldsymbol{X}_\theta^T \boldsymbol{X}_\theta)^{-1} \boldsymbol{X}_\theta^T \boldsymbol{y}_\theta \tag{3.90}$$

$\hat{\beta}_{xx}$ and $\hat{\beta}_{yy}$ returns the inertia inverse which is inverted back. Figure 3.32 presents the collected data for ω_ϕ and ω_θ. The input signal is based on the discrete sine wave $y(n) = Asin\left(\frac{2\pi n}{L} f\right) + \omega$, where the amplitude $A = 0.004$, white noise $\omega \sim N(0, \sigma^2)$ with variance σ^2 where $3\sigma = \frac{A}{3}$ and frequency $\frac{f}{L} = \frac{40}{0.8}$Hz. The resulting estimated and verified inertia is given by Table 3.12. Verification is based on a second set of collected data from equal input signal.

Table 3.12: Inertia estimations based on LLS.

	I_{xx} [kg/m^2]	I_{yy} [kg/m^2]
Estimate	0.001692037	0.001463176
Verification	0.001799774	0.001496044

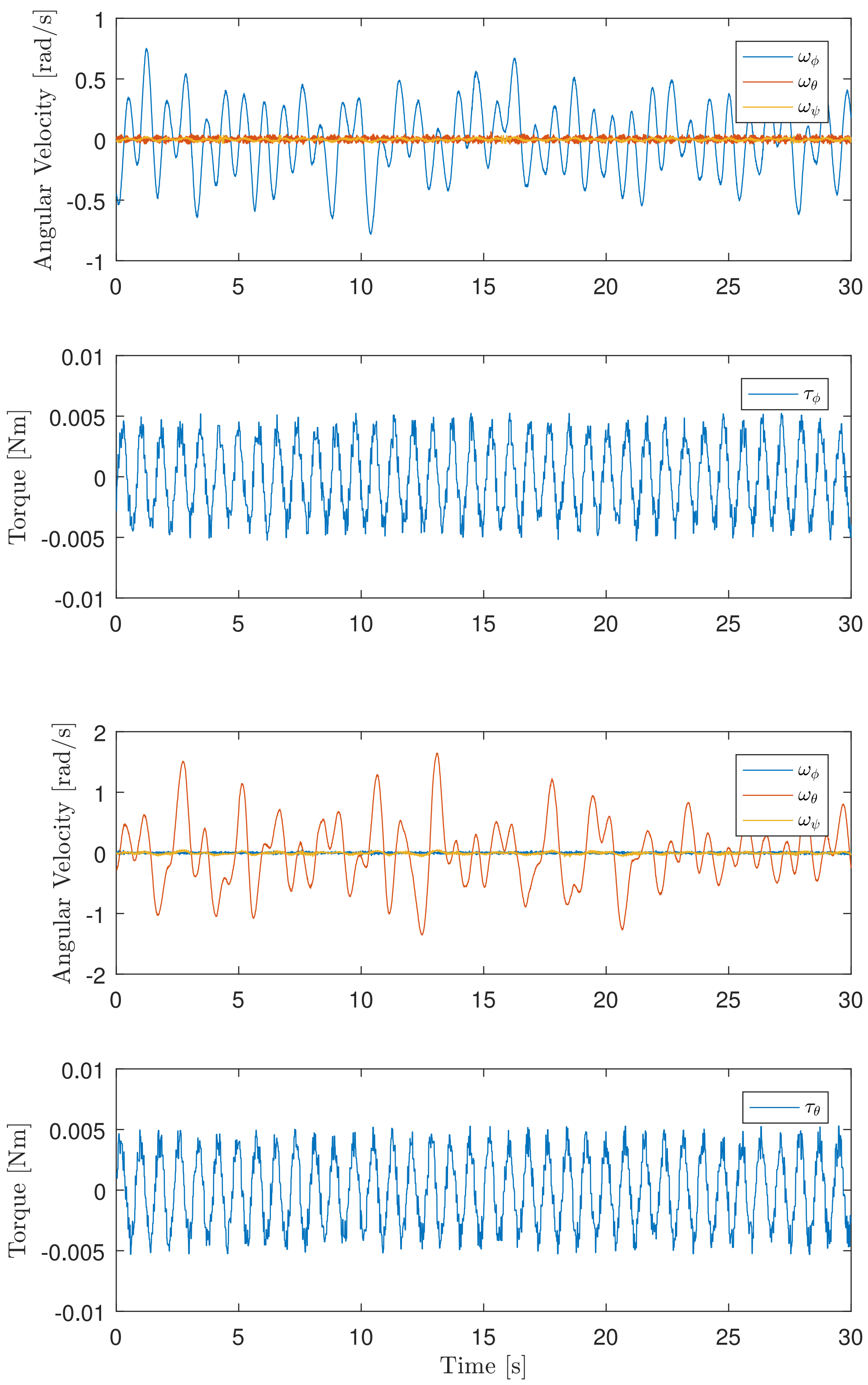


Figure 3.32: Open loop gyroscope response on input sine waves. Two first plots are ω_ϕ and τ_ϕ. Two last plots are ω_θ and τ_θ.

3.7 Implementation

This section presents the implemented solution of the base functionality of a single quadcopter. A hardware overview is presented together with how the overall system is configured and setup. A software overview is presented describing the architecture and functionality of the designed platform, including a list of the main functions.

3.7.1 Hardware

The assembled quadcopter developed for real time experiments is presented in Figure 3.33. The hardware setup contains a list of different components. The core of the flight control system is the *Raspberry Pi (RPi)* computer. The RPi receives the accelerometer and the gyroscope measurements from the IMU sensor board, performs sensor fusion and state estimation, calculates the control action and passes it to the motors. An external positioning system calculates the quadcopter position indoor. The positioning system consists of a mobile tag mounted on to the quadcopter and four static anchor points which are mounted around the indoor environment. Position estimation is carried out on a stand alone computer. All communication between anchors, server and quadcopter is over Wifi. Figure 3.34 presents an hardware overview. Table 3.13 lists all the hardware in details.

Table 3.13: Hardware overview.

Type	Manufacturer	Description
FPV250 frame kit	HobbyKing	Quadcopter frame
1704 Motors	Multistar	Brushless motors
3s LiPoly 1000mAh	Turnigy	Battery
5x3x3in Propellers	Hobbyking	Propellers (CW/CCW)
V3 12A ESC	Afro (Simionk)	Motor Speed Controller
16-Channel PWM Hat	Adafruit	Motor Controller
Raspberry Pi 3 Model B	Raspberry Pi	Flight Controller
Localino v2.0 Kit	Heuel & Loeher	Positioning System

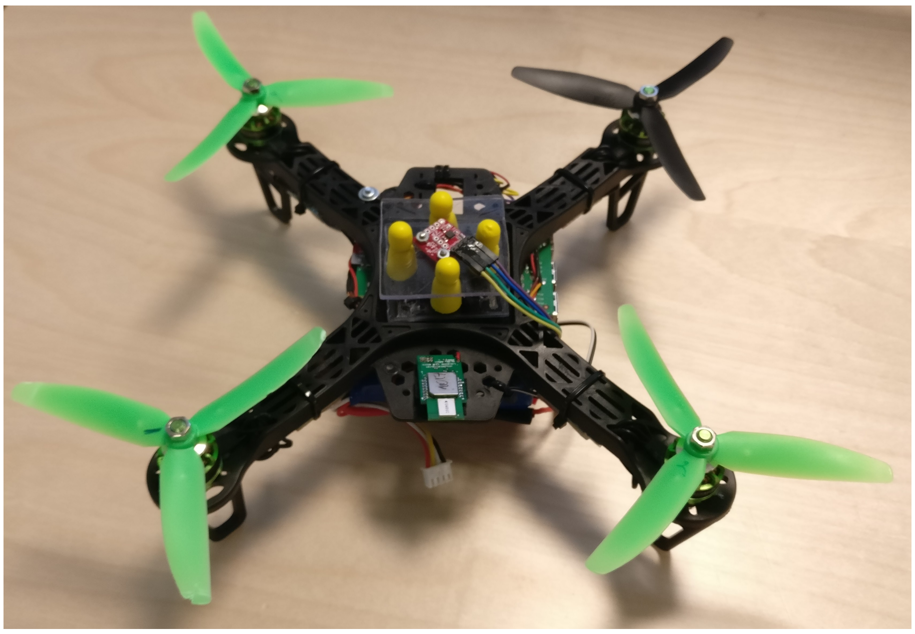


Figure 3.33: The quadcopter developed for real time experiments.

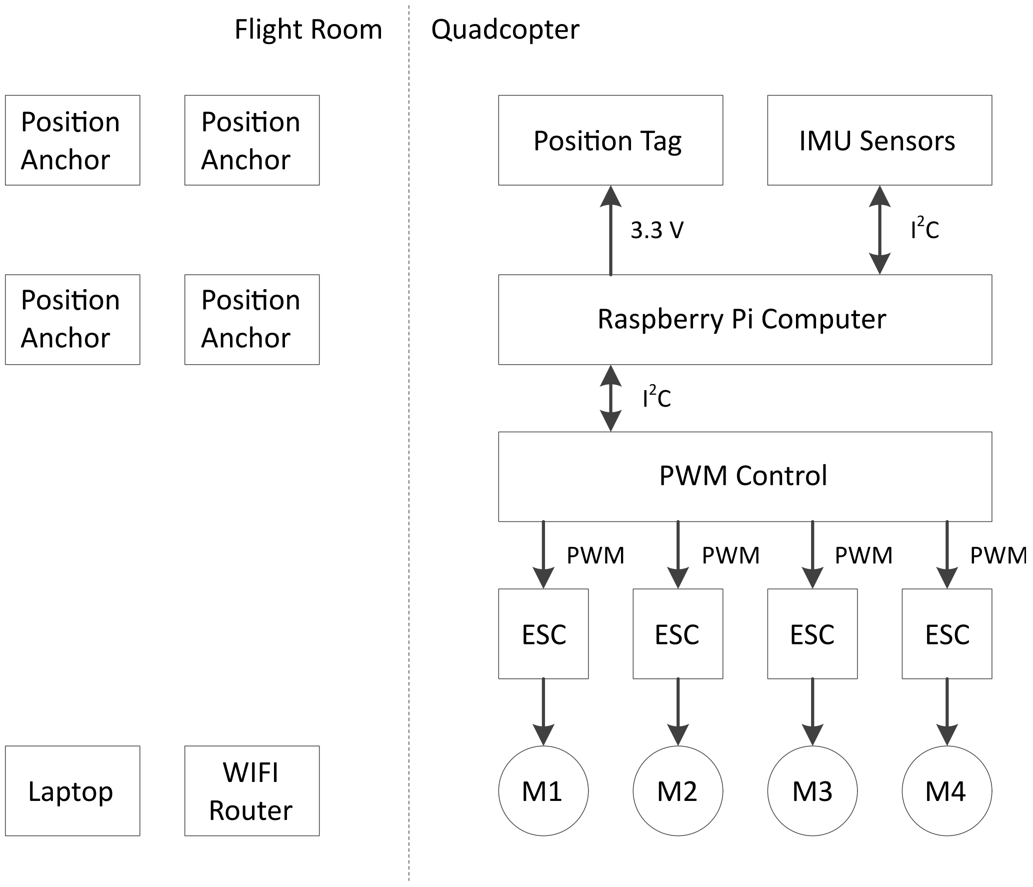


Figure 3.34: Hardware overview.

3.7.2 Software

The flight controller is developed from scratch, where all the presented algorithms from 3.2 are implemented in C. Systems based on MPC are typically computational demanding. The RPi has a four core CPU. By exploiting this hardware architecture, the computational load is expanded over as much as possible of the available recourses. Figure 3.35 presents the developed multi-core software architecture in a simplified overview. The system is initialized by a startup procedure in

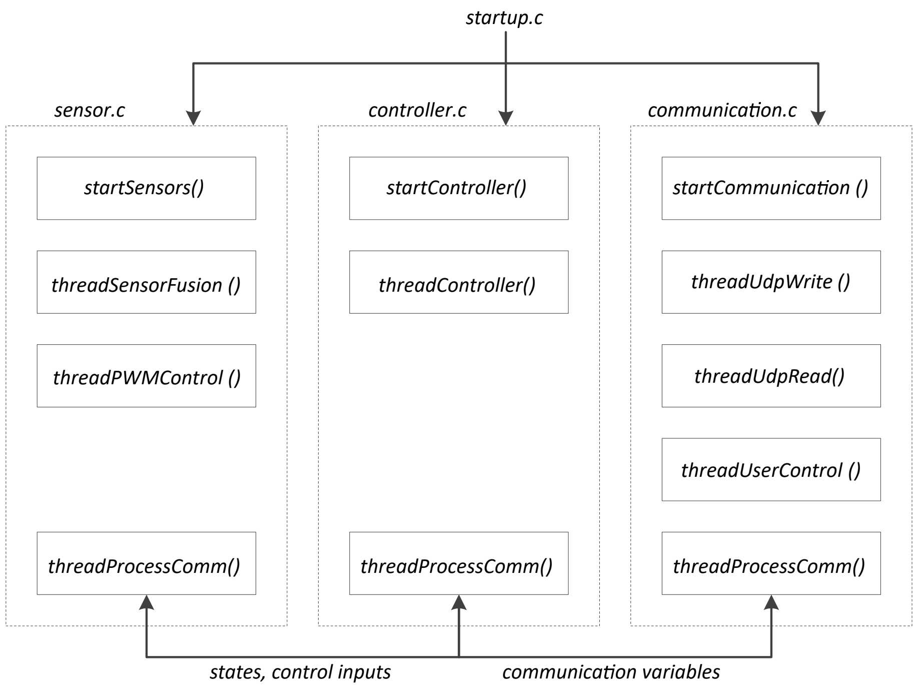


Figure 3.35: Simplified software overview.

`startup.c` which calls and activates three processes: `sensor.c` , `controller.c` and `communication.c`. The first process is reserved for tasks related to the sensors, including measurements, sensor fusion, state observer and motor control. The second process is reserved for tasks related to the FMPC, including position, attitude and altitude control. The last process is reserved for tasks related to communication. Between each process there is a two way pipe line with communication of variables such as estimated states, control inputs, constraints, communication inputs from the stand alone computer and variables to be communicated out from quadcopter. Within each process, the main tasks are split in to parallel running threads. For `sensor.c`, the following main functions are implemented:

- `threadSensorFusion()`, IMU sensor fusion, state observer and disturbance estimation:
  - `enableMPU9250()`, enable the IMU sensor board, including hardware sensor calibration,
  - `readAllSensorData()`, read the gyroscope and accelerometer,
  - `MadgwickAHRSupdateIMU()`, IMU filter orientation estimation,
  - `q2euler_zyx()`, quaternions to euler angles,
  - `ekfCalibration()`, calibrate EKF,

  - `EKF()`, state observer and disturbance estimation.
- `threadPWMControl()`, PWM motor control:
  - `setPWM()`, set the PWM signal to each individual ESC.

For `controller.c`, the following main functions are implemented:

- `threadControll()`, FMPC algorithms for position, attidue and altitude:
  - `controllerPos()`, position controller,
  - `controllerAtt()`, attitude controller,
  - `controllerAlt()`, altitude controller,
  - `refGen_formation()`, formation reference generator,
  - `getAltitudeInputConstraints()`, dynamic altitude control input constraints based on current attitude control input,
  - `controllerPID()`, PID controller for test scenarios,
  - `posFmpc()`, position FMPC algorithm called by the controller,
  - `attFmpc()`, attitude FMPC algorithm called by the controller,
  - `altFmpc()`, altitude FMPC algorithm called by the controller,
  - `fmpcsolve()`, FMPC solver called by each FMPC algorithm,

For `communication.c`, the following main functions are implemented:

- `threadUdpWrite()`, broadcasts states and inputs over UDP.
- `threadUdpRead()`, receives position measurements:
  - `messageDecode()`, decodes received message.
- `threadKeyReading()` (`threadUserControl()`) , user input from keyboard:
  - `keyReading()`, decodes the keyboard input.

The RPi runs a Linux kernel with the `PREEMPT_RT` real time patch [23]. Each individual thread is given a specific priority from [0...99], where 99 is the highest priority. In addition, certain threads which requires specific sampling times, are given this too. Table 3.14 presents an overview over the specific real time system priorities and sampling times. Notice that the highest priority is assigned the controller, as without a delivered control signal, the resulting flight control can be devastating for the quadcopter. Also, the specific sampling times for `threadController()` and `threadSensorFusion()` are assigned based on the calculated controller and observer sampling times from Sections 3.2.7 and 3.4.4.

Table 3.14: Real time priorities and sampling times.

Thread	Priority	Sampling Time [s]
`threadController()`	60	0.025
`threadSensorFusion()`	50	0.01
`threadPWMControl()`	40	`Interrupted`
`threadProcessComm()`	35	`Interrupted`
`threadUdpWrite()`	33	0.01
`threadUdpRead()`	32	`Interrupted`
`threadKeyReading()`	31	`Interrupted`

4 Results and Discussion

This chapter presents results from simulation and implementation using all the proposed methods from Chapter 3. The effect of different communication delays in a coordinated multi-agent system is benchmarked between the distributed and decentralized control strategies. Next, the formation flying algorithm is evaluated, followed by collision avoidance is verified for static obstacle and inter-vehicle collision avoidance, both with and without formation flying. Finally, the implemented solution on a single quadcopter is evaluated by mean of performance with respect to low cost hardware. In addition to a number of figures presented together with each simulation scenario, a selection of animations illustrating the main simulation scenarios can be found in Appendix A.

4.1 Simulation

This section presents results from simulation. To evaluate the performance of the distributed control strategy, the system is benchmarked against the decentralized control strategy. The simulation scenarios include mobile and stationary formation flying tracking, target tracking, static Obstacle Collision Avoidance and Inter-Vehicle Collision Avoidance. Final target reference ramping is used during all scenarios. In addition, a single scenario is also simulated without using target reference ramping. For simulation, all three quadcopters are implemented identical. Communication is set up between each agent with adjustable time delay. All simulations are implemented in `MATLAB` and `Simulink`. It is based on a combination of scripts written only for simulation and implemented C code from the real time system as a Software-In-Loop test. The SIL test has covered most of the functionality implemented in `sensor.c` and `controller.c`. Results given in this section are all based on different combinations of the methods described in Chapter 3 and the reader is referred to that chapter for detailed description of the methods.

4.1.1 Simulation Setup

Controller weights are taken the same for all scenarios to make it easier to compare. Table 4.1 presents the setup. The first two rows, $\boldsymbol{Q}$ and $\boldsymbol{R}$, show the state and input weights for the objective functions according to (2.1), where $\boldsymbol{I}_i$ is an identity matrix of dimension $i \times i$. The weights are all defined as diagonal matrices for each controller, with zero weights for cross values and based on the controller dimensions according to (3.63), (3.48) and (3.42). The third row shows the prediction horizon

T taken for each individual MPC. Finally, the last two rows are the tuning factors κ and K^{max} for FMPC as described in Section 2.1.2. Furthermore, the scenarios are run with full-state feedback without the observer, noise-free measurements and zero disturbances.

Table 4.1: Controller setup throughout simulation results.

	Position controller	Attitude controller	Altitude controller
$\boldsymbol{Q}$	$\begin{bmatrix}10000\\1\\10000\\1\\10000\\10000\end{bmatrix}\boldsymbol{I}_6$	$\begin{bmatrix}1000\\1\\1000\\1\\1000\\1\end{bmatrix}\boldsymbol{I}_6$	$\begin{bmatrix}100000\\100\end{bmatrix}\boldsymbol{I}_2$
$\boldsymbol{R}$	$\begin{bmatrix}100\\100\end{bmatrix}\boldsymbol{I}_2$	$\begin{bmatrix}1000\\1000\\1000\end{bmatrix}\boldsymbol{I}_3$	0.1
T	20	20	20
κ	0.1	0.001	0.001
K^{max}	5	5	5

4.1.2 Distributed Control against Decentralized Control on Formation Flying and Target Tracking

Different control strategies are exploited given the requirements and limitations of a multi-agent system as described in Section 2.3. A benchmark study between the decentralized and the distributed strategy, is carried out in simulation in terms of performance. In general, controller performance can be described in different terms such as reference tracking error, *Time Of Arrival (TOA)*, oscillating behaviour and chain reaction behaviours. Evaluating performance factors is a trade-off, e.g. an attempt in making on performance criteria better might weaken another one. The performance terms analyzed in this benchmark are listed as the following

1. Formation flying with respect to tracking error,
2. TOA at the final target reference,
3. The effect of communication delays on the shared information between agents on terms 1. and 2.

The scenario tested is the same for all the different cases. That is manoeuvring in XY plane between a set of target references while keeping formation between all agents. After starting from initial position and reaching the *current* set of target reference points within a vicinity given by $||\hat{\boldsymbol{x}}_i^{xy} - \boldsymbol{x}_{i,tref}|| \leqslant 0.02; \forall i = \{1...3\}$, a new set of target reference points are given for all three agents to track. Figure 4.1 presents the scenario in XY plane, where the formation moves between two sets of target reference points, $\boldsymbol{x}_{i,tref_1}$ and $\boldsymbol{x}_{i,tref_2}$. An animation is presented in Figure A.1 in Appendix A.

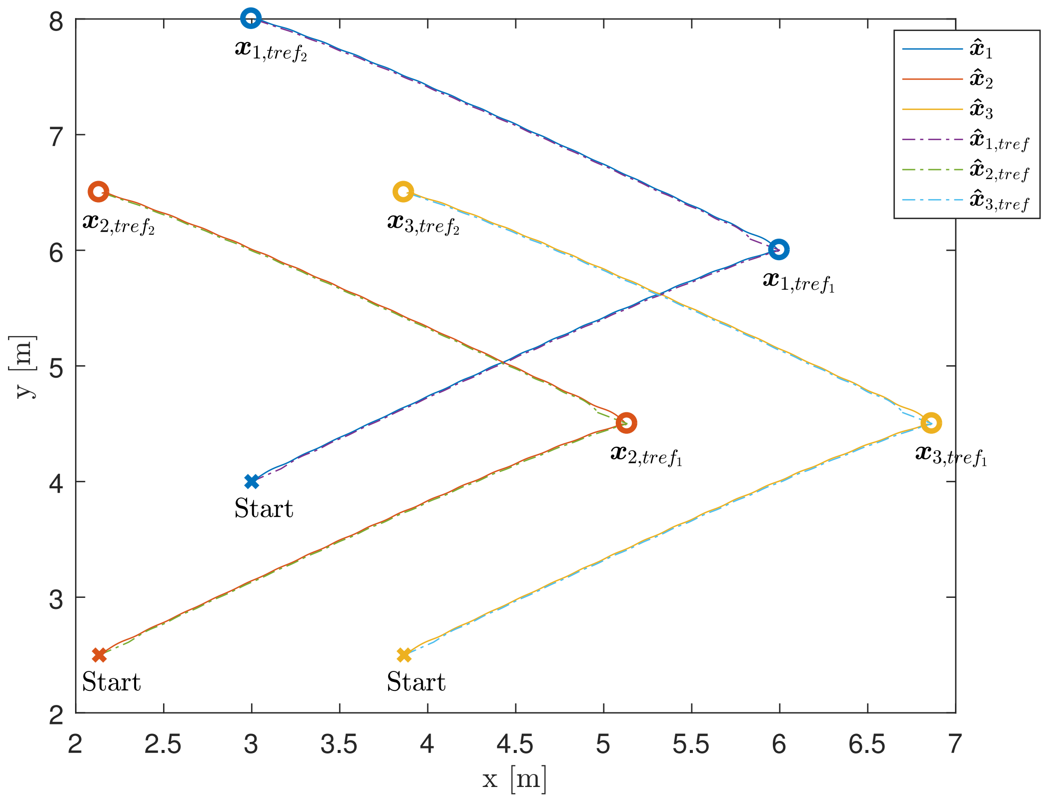


Figure 4.1: Simulation scenario in XY plane.

Decentralized Control Figure 4.2 presents the resulting performance for the decentralized control scheme in an environment with delay in communication as described in 3.2.6. Agent $\boldsymbol{x}_{i,k}$ at current time k, receives information of the remote agents $\hat{\boldsymbol{x}}_{j,k-5}; \forall j \neq i$ with $t_z = 5$ delay steps of sampling time. Given the sampling frequency of $f_s = 40$ Hz, this results in $t = 0.125$ seconds of delay for information between the agents. Target references are ramped as described in Section 3.3.3. The set of target references are changed from $\boldsymbol{x}_{i,tref_1}$ to $\boldsymbol{x}_{i,tref_2}$ at time $t \approx 12.5$ s. The plots show that the agents are leading their formation references $\boldsymbol{x}_{i,fref}$. Since position information from the other agents is used in the formation flying algorithm, this leading behaviour on formation reference tracking is expected due to the delayed information. The algorithm is based on delayed information for formation flying path-planning, hence the formation references lags the agent.

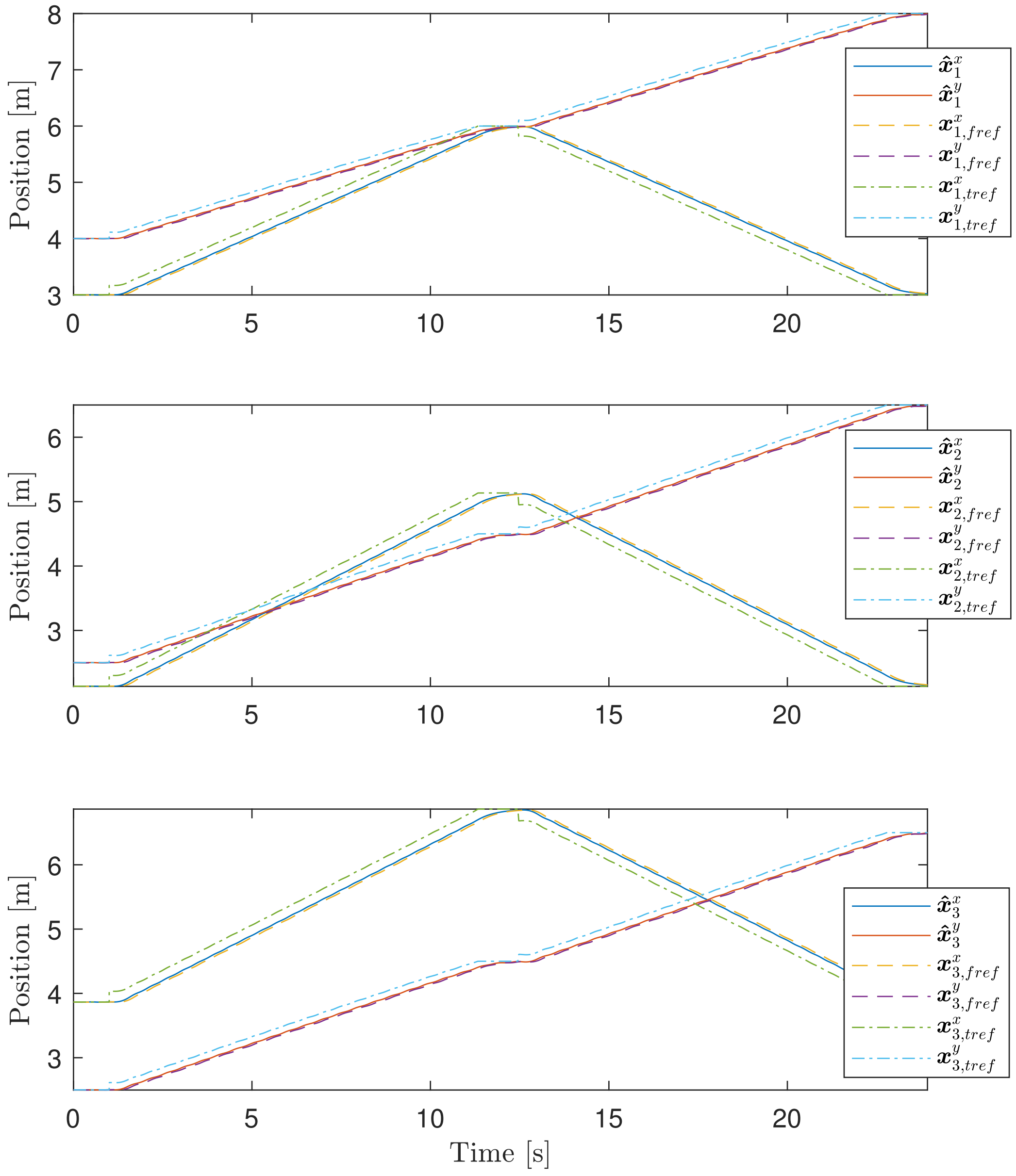


Figure 4.2: Performances of decentralized control scheme with $t_z = 5$

Distributed Control The formation tracking performance can be improved by using the distributed control strategy. Figure 4.3 presents the same delay $t_z = 5$ scenario using distributed control. Here agent $\boldsymbol{x}_{i,k}$ at current time k, receives information of the remote agents $\hat{\boldsymbol{x}}^m_{j,k-5}; \forall j \neq i$ as well as the predicted information $\hat{\boldsymbol{x}}^p_{j,k-5}; \forall j \neq i$. By using the predicted states of each remote agent, it can be compensated for the communication delay as suggested in Section 3.2.6. For the delay step of $t_z = 5$, the predicted state information of $p = 5$ is used from the remote agents, $\hat{\boldsymbol{x}}^p_{j,k-5|k}; \forall j \neq i$ for agent $\boldsymbol{x}_{i,k}$.

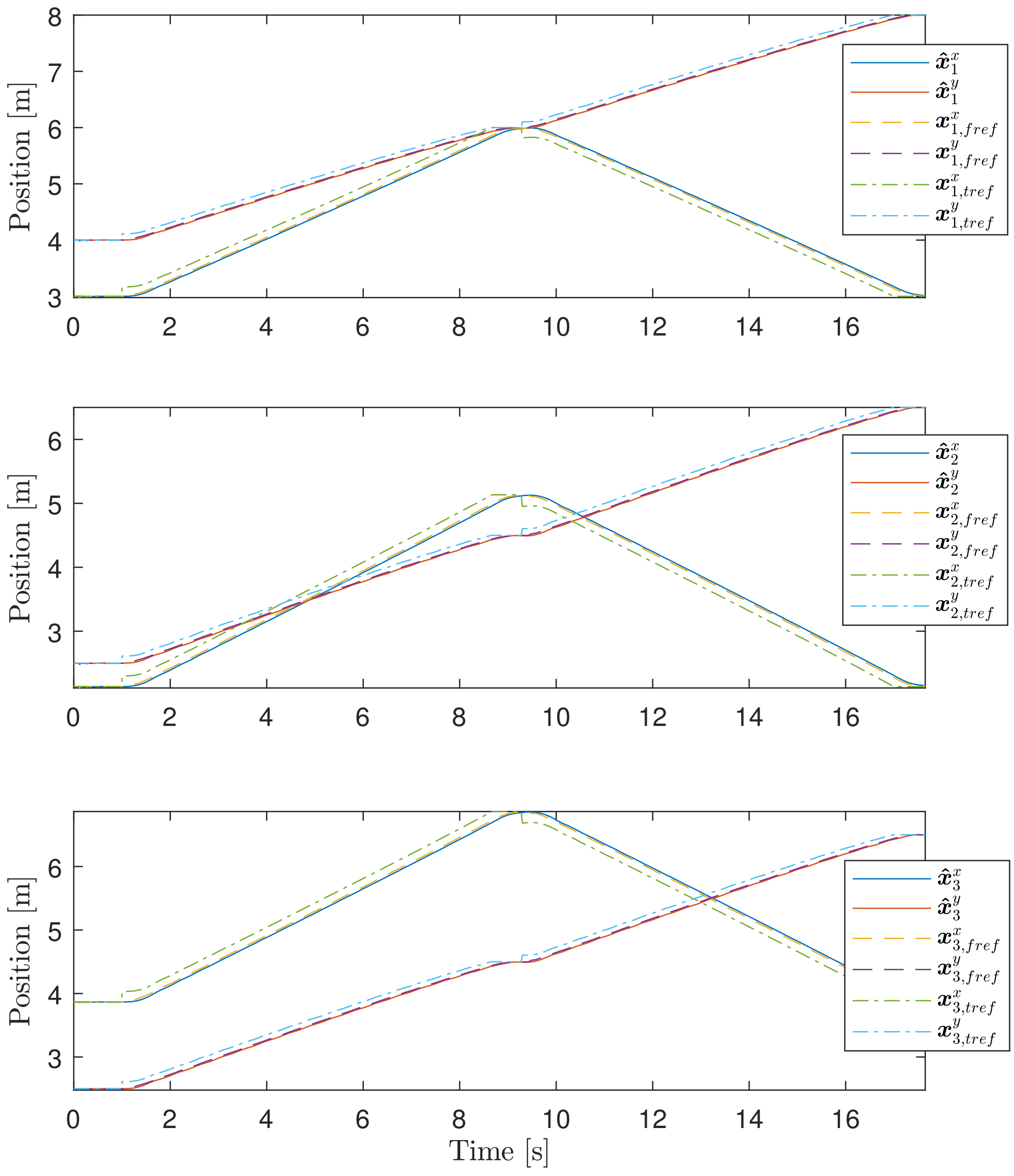


Figure 4.3: Performances of distributed control scheme with $t_z = 5$

Complete Benchmark of Different Time Delays Generally, as delay in communication increases, one can expect performances of both the decentralized and distributed strategies to get worse in terms of formation tracking. Figure 4.4 illustrates how the distributed scheme for a delay of $t_z = 15$ has an increase of formation flying tracking error compared to $t_z = 5$. In addition, even though the distributed control strategy with predicted information has been used, it still cannot compensate for the larger communication delays such as $t_z = 15$, hence there is a similar behaviour to the decentralized strategy, where the agent is leading of formation references. Accuracy of the prediction model highly affects such performances. Also, as the system is correlated in terms of formation flying where each agent compensates for each others error, regardless whether the prediction model is strong, constantly changes of control action on each individual agent makes long predictions difficult.

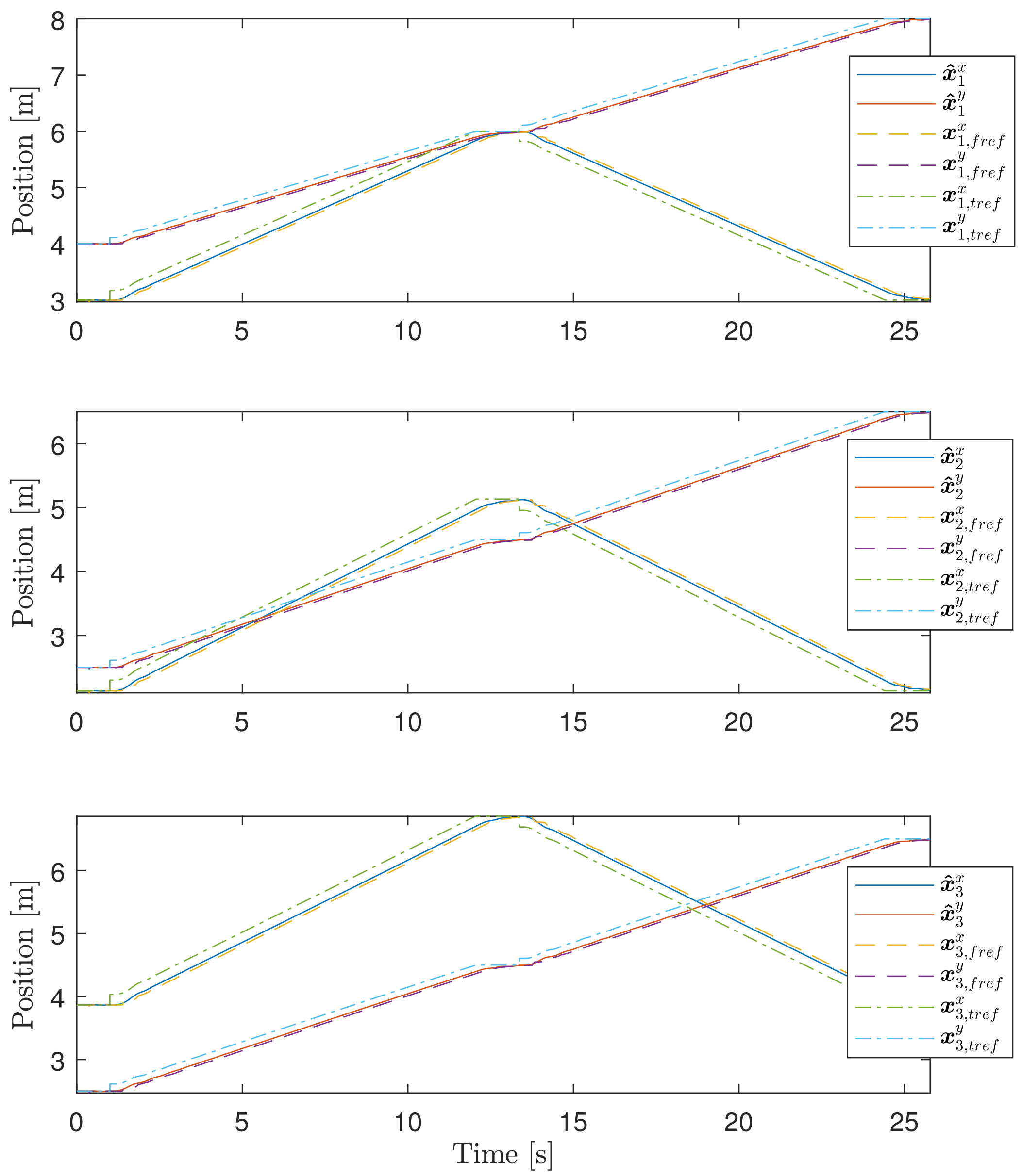


Figure 4.4: Performances of distributed control scheme with $t_z = 15$

A complete comparison between the decentralized and distributed control strategy is presented on Figures 4.5 and 4.6 for a set of delays $t_z = \{1...15\}$. Figure 4.5 presents the average error which represents the performances of formation flying and is given by

$$e_f = \frac{1}{N}\sum_{k=1}^{N}(\hat{\boldsymbol{x}}_{1,k}^{xy} - \boldsymbol{x}_{1,fref,k}) + (\hat{\boldsymbol{x}}_{2,k}^{xy} - \boldsymbol{x}_{2,fref,k}) + (\hat{\boldsymbol{x}}_{3,k}^{xy} - \boldsymbol{x}_{3,fref,k}) \tag{4.1}$$

where $\boldsymbol{x}_{i,fref,k}$ for $i = \{1, 2, 3\}$ is the formation references for i-th agent at time k as described in Section 3.3.1. Figure 4.6 resents the TOA evaluated with respect to all the agents arriving into the vicinity of their final target reference points, e.i. $||\hat{\boldsymbol{x}}^{xy}_{i,k} - \boldsymbol{x}_{i,tref,final}|| \leqslant 0.02; \forall i = \{1, 2, 3\}$. This is generally inversely proportional to formation tracking performances, as when agents are more off-tracking from formation, they are more liberated to reach the target references and consequently reach the destination faster. Figure 4.5 presents two sets of errors for distributed scheme: first, the error described by (4.1), when the agents share predicted horizon information equivalent to their delays, i.e. $p = t_z$, as described in Section 3.2.6. Second, it additionally presents the same error as (4.1), but when the agents share predicted horizon information different to their communication delays, i.e. $p = t_z - 1$.

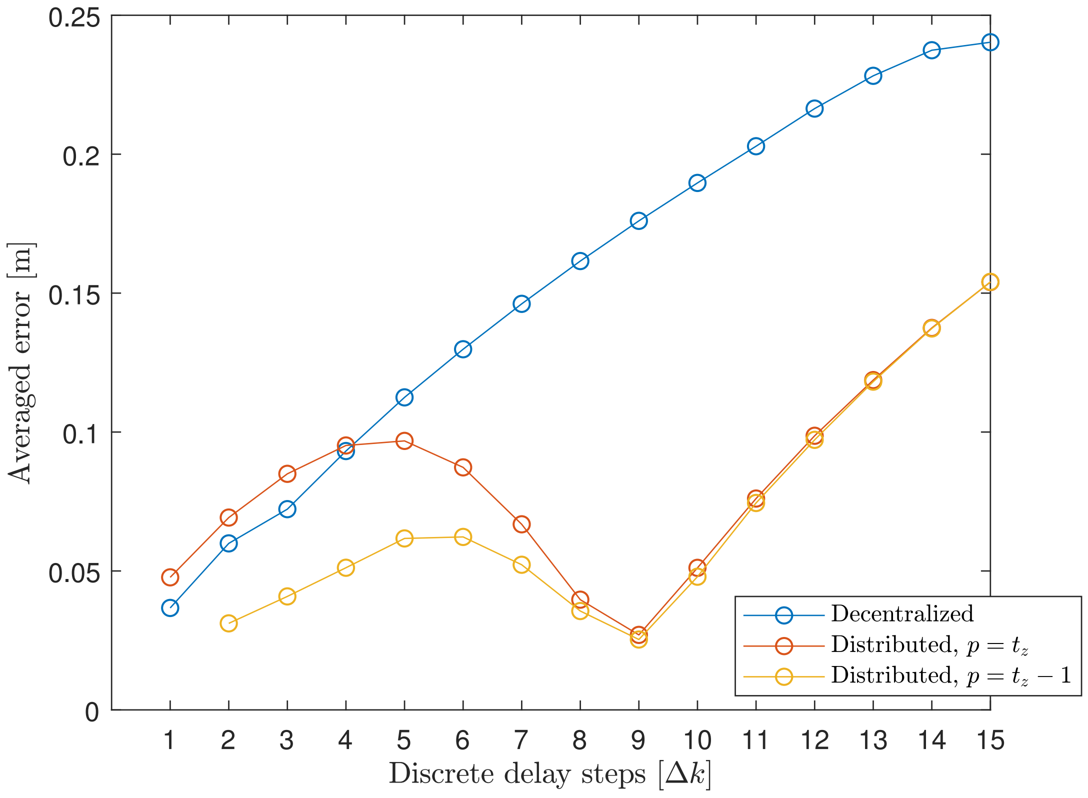


Figure 4.5: Formation flying error for different communication delays.

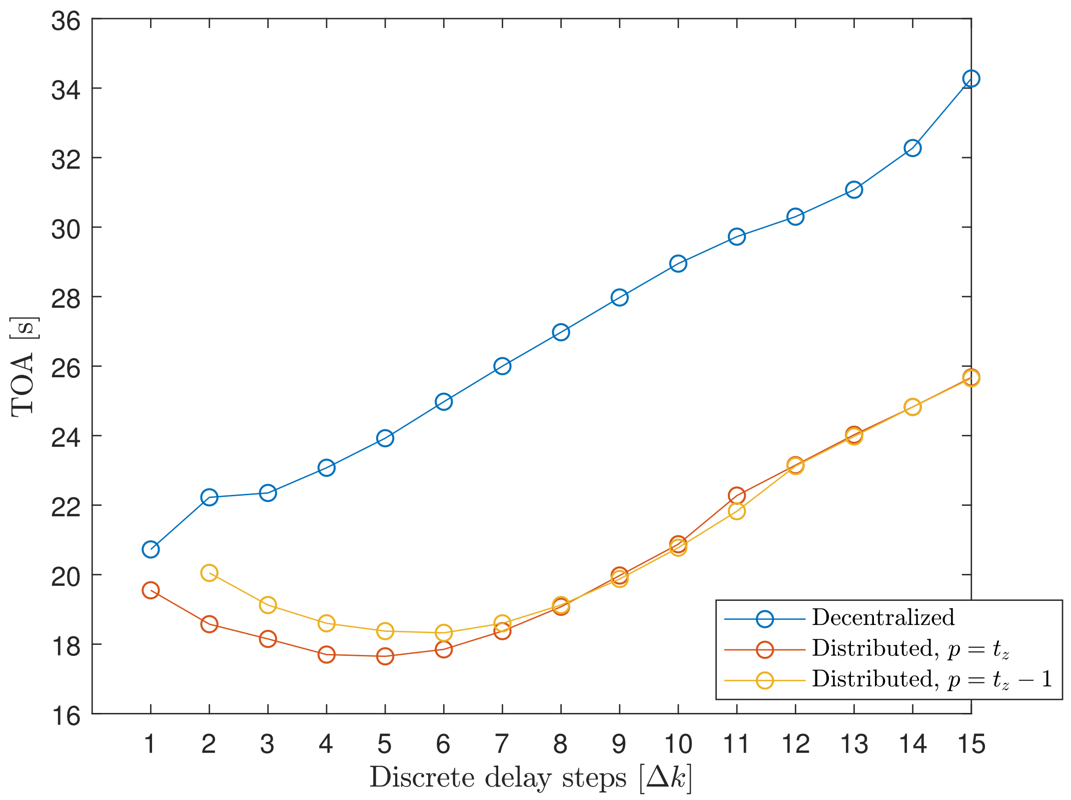


Figure 4.6: TOA for different communication delays.

Distributed control strategy shows better formation tracking performances for $t_z \geq 5$ for when sharing horizon equivalent to the communication delay ($p = t_z$) and better formation tracking performances for all delays when sharing horizon different to the communication delays ($p = t_z - 1$). Distributed control strategy also shows better TOA performances for all delays, regardless of which horizon is shared between the agents.
An interesting observation when using the distributed control strategy with $p = t_z$ is that the error is increasing together with the delay in the range of $t_z = \{1...5\}$ as expected, however for the delays in the range of $t_z = \{6...9\}$ the error is decreasing as the delay increases, before the error starts increasing again in parallel to the increase of delay from $t_z \geq 10$. The reason for this is that the formation reference is leading the agent up $t_z = 5$ where the trend turns around going over to a lagging reference. The result is that as the reference is going from leading to lagging, it will surpass the current position of the agent, hence the error decreases with decreases amount of leading, before the error increases again with increases amount of lagging. This trend behaviour is suspected to be due to the lack of prediction quality as the delay increases because of the non-linearity in the plant model compared to the linearized MPC model for position. One can argue that prediction with linear model upholds good approximations for predictions of less than $p = 6$ into the horizon and as delays get larger they are not as good and make the references lag the agents. The case is similar when using distributed control strategy with shared horizon different to the communication delay ($p = t_z - 1$).

Another interesting observation is that distributed scheme using shared horizon different to the communication delay ($p = t_z - 1$) shows better formation tracking performances than distributed scheme using shared horizon equivalent to the communication delay ($p = t_z$). This demonstrates that not necessarily using the shared horizon equivalent to the communication delay would result in the optimum solution in terms of errors. Using shared horizon different than communication delays with different shifts, e.g. $p = t_z - 2$ or $p = t_z - 3$, have resulted in arbitrary better or worse performances than $p = t_z - 1$, but always better performances than decentralized scheme.
An interesting area of future research would be to improve the prediction model to perform larger delays scenarios where the strength of using the predicted states in a distributed fashion becomes even more obvious. Also as more approximated models compared to reality are used in control designs and shared prediction are based on models used in MPC control design, another interesting area to be investigated further is that which prediction horizon step in regards to communication delay should be shared between the agents to optimize the error.

Without target reference ramping The distributed and decentralized control strategy is also benchmarked without ramping the target reference. The error will no longer be limited down by the ramp function, hence the performance is expected to become more aggressive with a considerable shorter TOA. The simulation scenario is equal the scenario presented in Figure 4.1. Figures 4.7 and 4.8 present the resulting performance using the decentralized and distributed control strategy with delay $t_z = 5$ respectively.

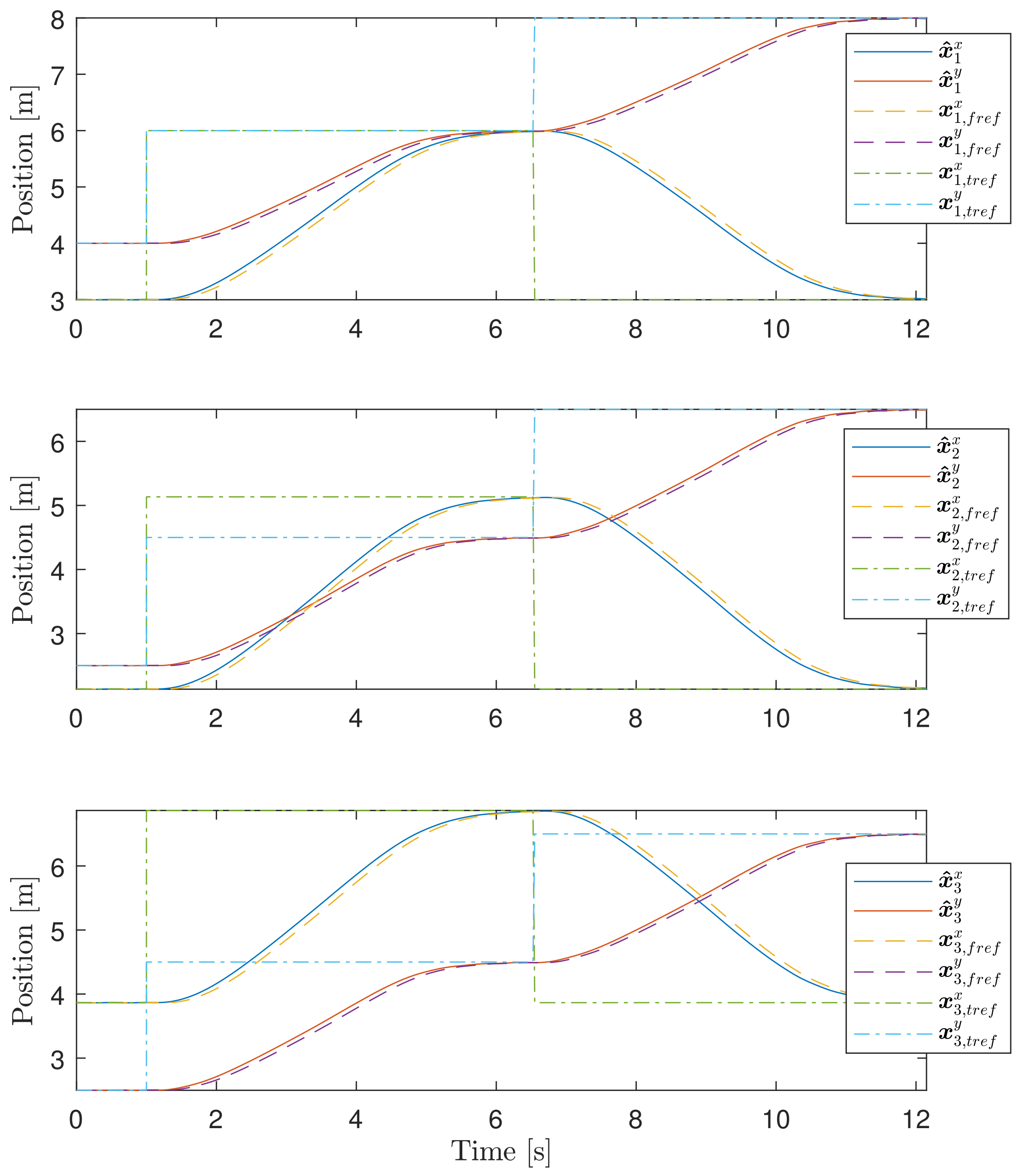


Figure 4.7: Performances of decentralized control scheme with $t_z = 5$ without target ramping.

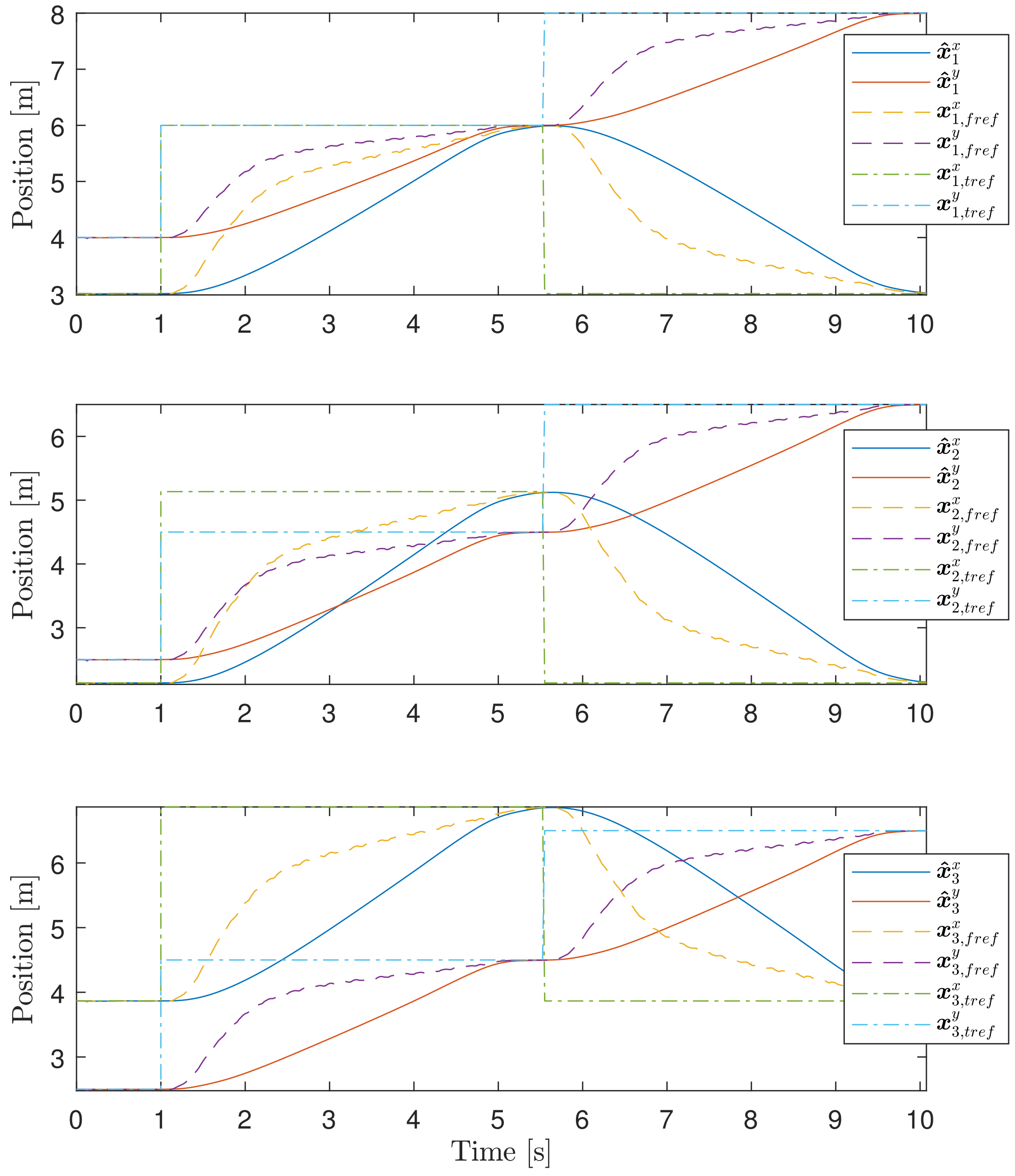


Figure 4.8: Performances of distributed control scheme with $t_z = 5$ without target ramping.

The resulting performance has changed between the two strategies in favour of the decentralized control scheme. Formation flying reference is lagging the agent position for the decentralized controller; the same as with using the reference ramping. This is expected as the larger error in target reference will be minimized the most. As the agents approach the target references at $t \approx 6$s and $t \approx 13$s it is clear that the formation flying error becomes smaller as the formation flying and target references become equally small. The distributed control scheme performance is considerably worse. TOA is faster than the decentralized control strategy, however due to large error in formation flying, the faster TOA is not considered as an improvement. The larger formation flying error is suspected to be a result of the prediction states.

When the target tracking error is large, the resulting predicted velocity will be large, hence the larger error in formation state prediction and tracking. The velocity is considered to be increasing out from the step change at $t \approx 1.5 - 3$s in target reference and decreasing as the error decreases and the agent approaches the target reference at $t \approx 3 - 5$s. As a result of this, the formation flying error is increasing and decreasing in a similar fashion.

Altitude control The altitude control during all simulation scenarios are based on an initial altitude and reference of $z = 2$m. Figure 4.9 presents the resulting altitude tracking performance of agent $\boldsymbol{x}_1$ during the main simulation scenario presented in Figure 4.1. The altitude tracking is stable with a small constant offset of ≈ 0.015m. Small dips in altitude are visible at the times ≈ 1.5s and ≈ 9s. These are caused by step changes in target references for position x and y, which forces the quadcopter in to necessary angles in order to reach the new position references. As a result, the vertical thrust vector is shifted from pointing directly upwards to having components in both x and y directions. The altitude performances are more or less similar for all agents during all simulation scenarios, hence only this particular result is presented.

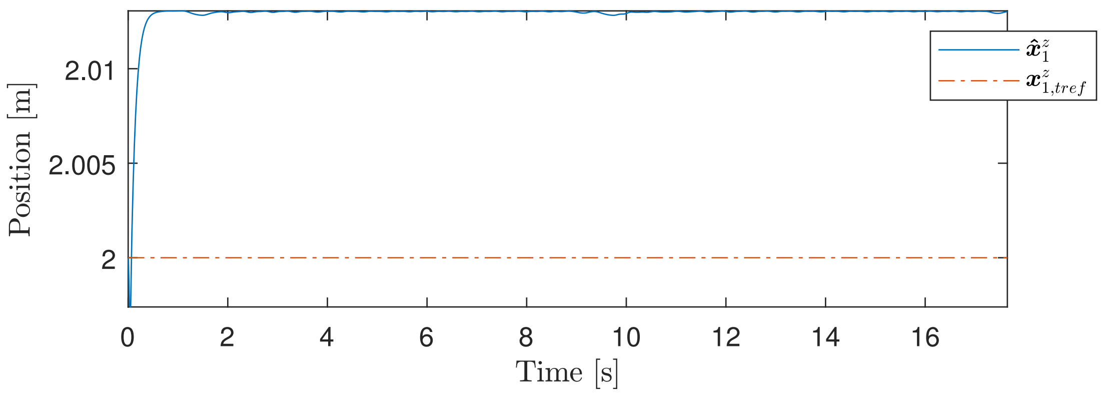


Figure 4.9: Performances of altitude control agent $\boldsymbol{x}_1$ during distributed control scheme with $t_z = 5$.

4.1.3 Stationary Formation Flying

Stationary formation flying in this context is described as when the agents are on the target references and still actively keeping the formation reference. Since the formation flying is based on circle intersections which are depending on each of the agents position, small errors in formation tracking causes a chain reaction between the agents as they try to compensate their own error, while at the same time move the formation references of the other agents. Figure 4.10 presents the decentralized control strategy where the chain reaction causes large oscillating behaviour around the target reference. The communication delay between the agents is $t_z = 5$. As soon as one of the three agents compensates for its error, the error of the two other agents increases or decreases. Notice that the oscillation is not increasing in to unstable states. As both target and formation states are equally weighted in the controller, an increase of error in target reference compared to formation reference would *pull* the agent back towards the target reference.

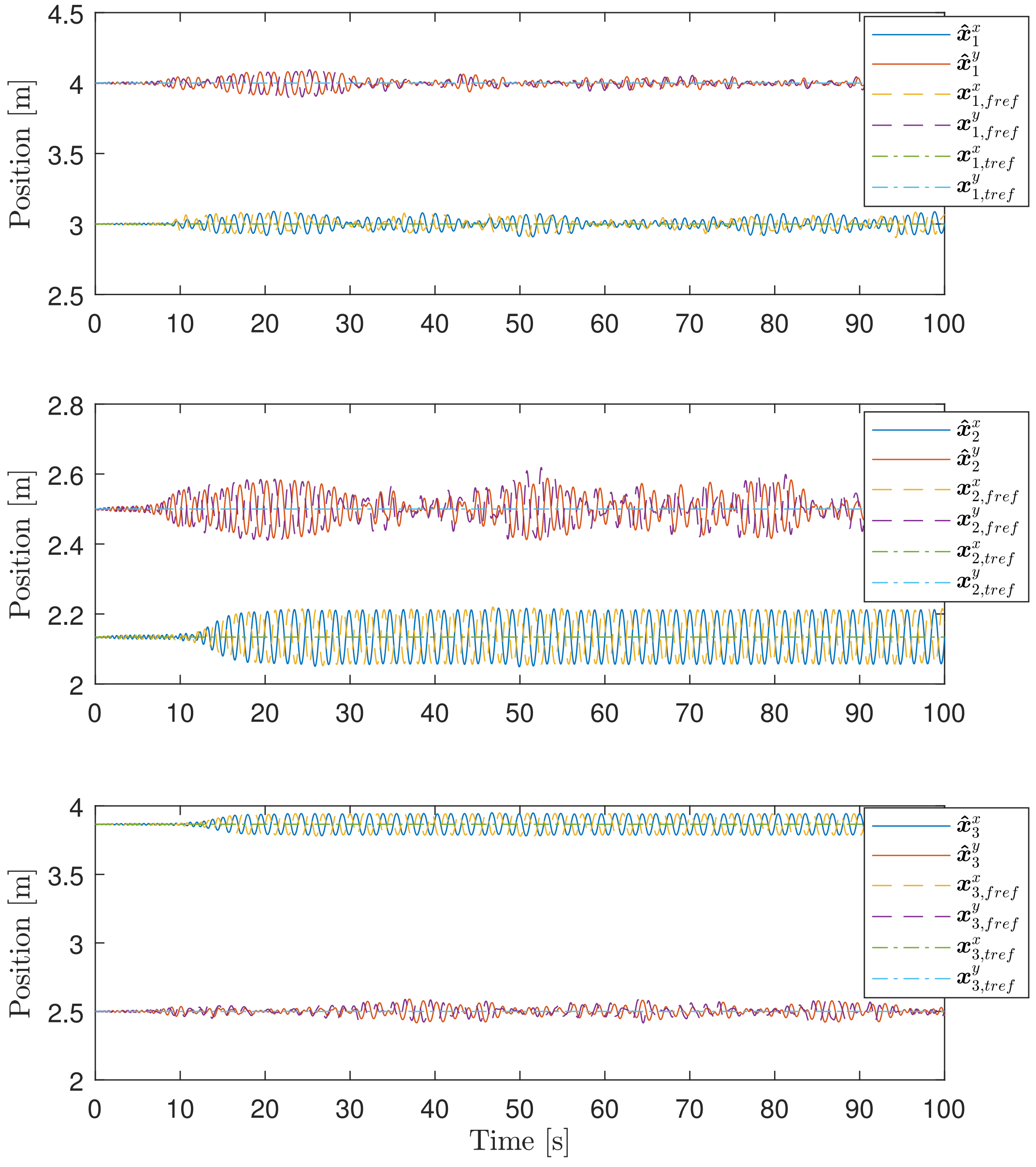


Figure 4.10: Performance of decentralized control scheme with $t_z = 5$ for stationary formation flying.

The stationary chain reaction can be improved by using the distributed control strategy, with the predicted state information of $p = 5$. Figure 4.11 presents the resulting performance, where the oscillating behaviour is still present, though significantly dampened.

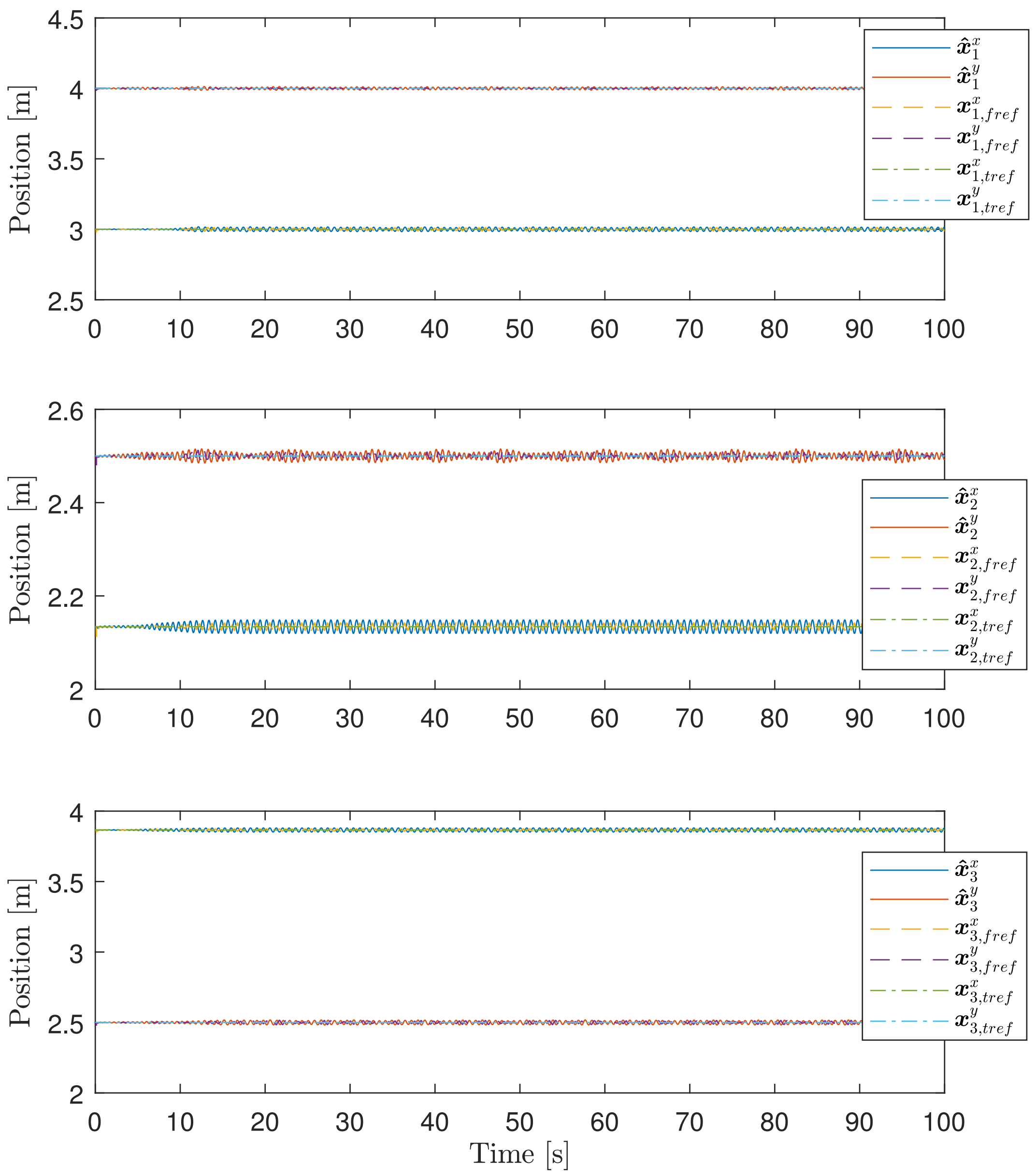


Figure 4.11: Performance of distributed control scheme with $t_z = 5$ for stationary formation flying.

Table 4.2 presents the variance of the formation flying positions of each agent and emphasizes how the predicted state information has contributed by dampening the variance with a minimum of 8.8 times and up to 31.8 times when using the distributed control scheme compared to the decentralized.

Table 4.2: Variances of formation flying chain reaction when using the distributed and decentralized control scheme.

	Decentralized		Distributed	
Agent	σ_x^2 [m]	σ_y^2 [m]	σ_x^2 [m]	σ_y^2 [m]
$\boldsymbol{x}_1$	0.00142	0.00077	0.00010	0.00004
$\boldsymbol{x}_2$	0.00267	0.00159	0.00010	0.00005
$\boldsymbol{x}_3$	0.00267	0.00088	0.00009	0.00010

4.1.4 Obstacle Collision Avoidance

The OCA algorithm described in 3.3.2 is simulated over two different scenarios. First, a single agent performs avoidance of a static obstacle. Second, the formation flying scenario from Section 4.1.2 is introduced to a static obstacle. Figure 4.12 presents the resulting tracking performance of the single agent scenario in XY plane together with the true distance between the position of agent $\hat{\boldsymbol{x}}_1^{xy}$ and the obstacle $\boldsymbol{x}_{obs}^{xy}$ over time, given by $\|\hat{\boldsymbol{x}}_1^{xy} - \boldsymbol{x}_{obs}^{xy}\|$. An animation is presented in Figure A.2 in Appendix A. As the agent moves within $r_{obs,lim}$, the target reference point switches to the temporary avoidance reference around r_{obs}. As soon as the distance between the current position of $\hat{\boldsymbol{x}}_1^{xy}$ and the target reference $\boldsymbol{x}_{1,tref}^{xy}$ is shorter than the distance between $\boldsymbol{x}_{obs}^{xy}$ and $\boldsymbol{x}_{1,tref}^{xy}$, the temporary avoidance reference is completed and the agent continues towards $\boldsymbol{x}_{1,tref}^{xy}$. Note that due to the ramp functionality, the agent distance decreases and increases linearly before and after passing $r_{obs,lim}$ at $t \approx 7$s and $t \approx 17$s. The figure illustrates how the size of $r_{obs,lim}$ can be used to tune the collision avoidance entrance curve. If $\hat{\boldsymbol{x}}_1^{xy}$ has a high velocity, a small $r_{obs,lim}$ would cause the agent to perform aggressive slow down manoeuvre with a resulting sharp turn around $\boldsymbol{x}_{obs}^{xy}$. By increasing $r_{obs,lim}$, $\hat{\boldsymbol{x}}_1^{xy}$ has more room to slow down and start avoidance in a less aggressive fashion. Note that the collision avoidance algorithm is based on reference tracking. How well the avoidance performance is depends on the selected controller weights. For a single agent without any other obligations such as formation tracking, the weights on target states x and y could be larger in order to avoid $\boldsymbol{x}_1^{xy}$ violating r_{obs} a $t \approx 10 - 15$s. However, the weights are kept as given in Section 4.1.1 for consistency throughout the results chapter.

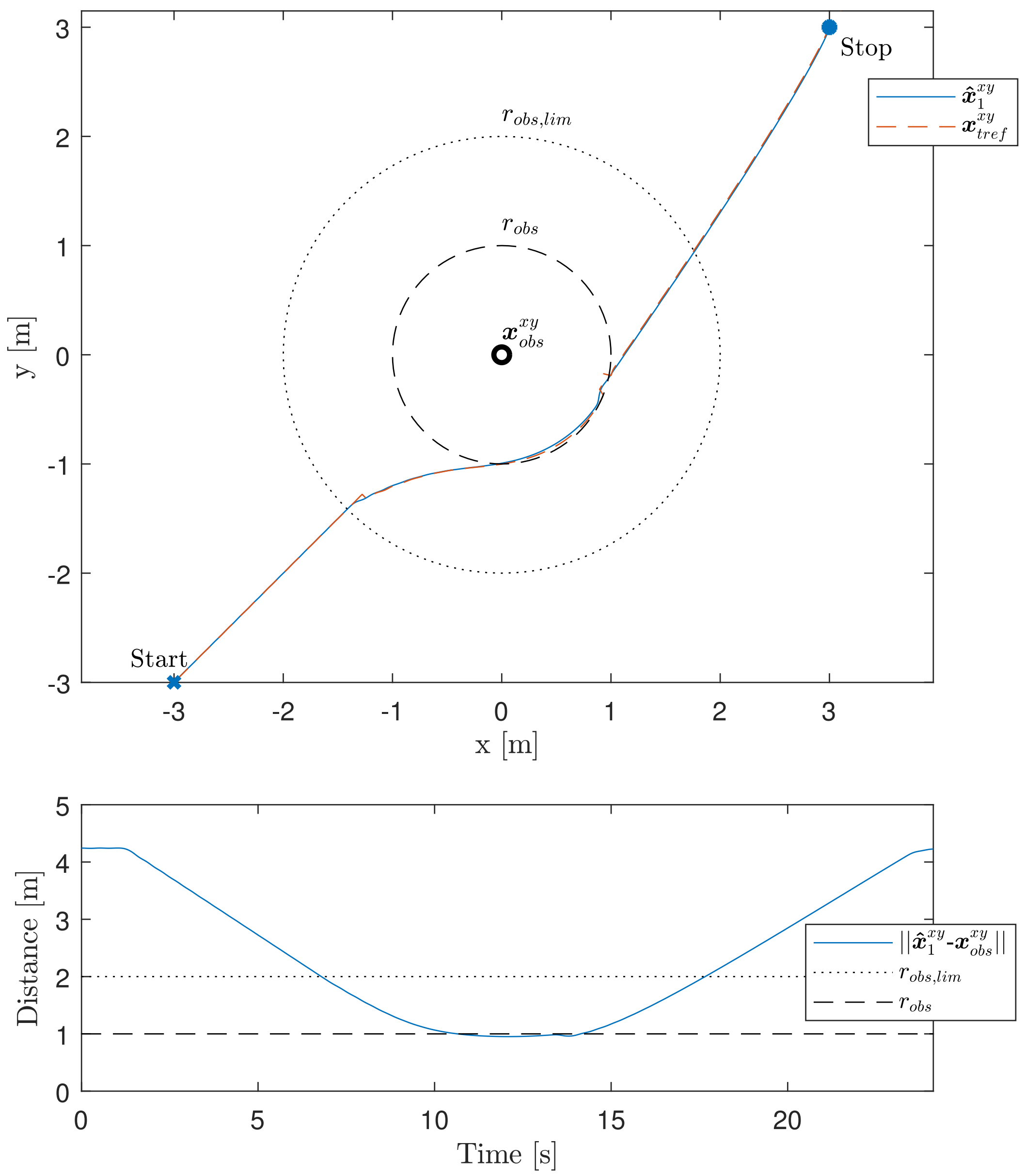


Figure 4.12: Performances of static collision avoidance with a single agent.

Figure 4.13 presents the obstacle avoidance performance with three agents and formation tracking. An animation is presented in Figure A.3 in Appendix A. Each agent has now two references to track, target reference $\boldsymbol{x}_{i,tref}^{xy}$ and formation reference $\boldsymbol{x}_{i,fref}^{xy}$. Initially from start, it is only agent $\boldsymbol{x}_3$ who has the obstacle $\boldsymbol{x}_{obs}^{xy}$ on its path to the first target $\boldsymbol{x}_{3,tref_1}^{xy}$. As $\boldsymbol{x}_3$ performs avoidance within $r_{obs,lim}$, the formation is kept between the three agents. As a result of this, $\boldsymbol{x}_2$ is forced to perform avoidance as $\boldsymbol{x}_{obs}^{xy}$ comes in to its path to $\boldsymbol{x}_{2,tref_1}^{xy}$. As both $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ perform avoidance, $\boldsymbol{x}_1$ is left tracking formation without needing to take the obstacle in to account. Since the controller weights on target states x and y and formation states x_f and y_f are all equal, the position of agent $\hat{\boldsymbol{x}}_i^{xy}$ lays in between the references $\boldsymbol{x}_{i,tref}^{xy}$ and $\boldsymbol{x}_{i,fref}^{xy}$. For better performance, an alternative is that target tracking gets substantially higher weights compared to formation tracking as the obstacle avoidance mode is activated; in case formation tracking would suggest a point inside the critical region r_{obs}. This is also mentioned in the Future Work.

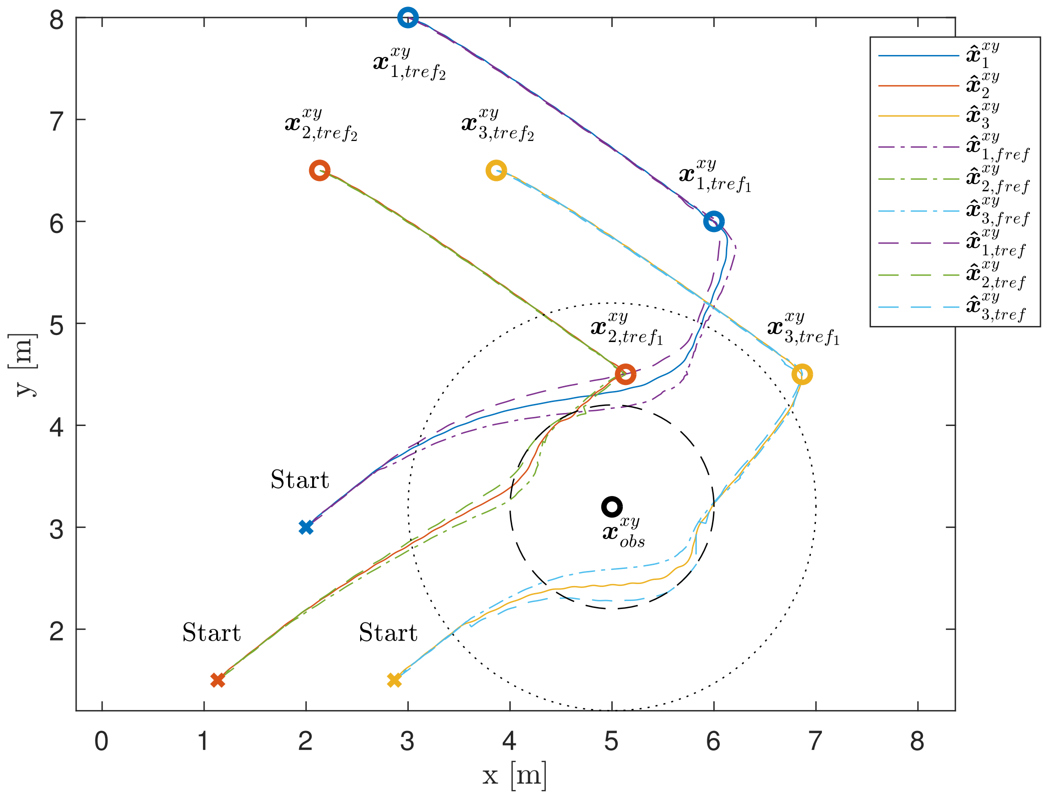


Figure 4.13: Performance of static collision avoidance together with formation flying in XY plane.

Figure 4.14 presents the resulting distance between each agent and the obstacle given by $\|\hat{\boldsymbol{x}}^{xy}_i - \boldsymbol{x}^{xy}_{obs}\|$, and the formation and target tracking error in the XY plane given by

$$e_{i,f} = \|\hat{\boldsymbol{x}}^{xy}_i - \boldsymbol{x}_{i,fref}\| \qquad e_{i,t} = \|\hat{\boldsymbol{x}}^{xy}_i - \boldsymbol{x}_{i,tref}\| \tag{4.2}$$

The target reference errors $e_{i,t}$ have the constant offset of 0.2m which is equal to the step size of the ramp functionality. The formation errors $e_{i,f}$ are all equal to each other up to $t \approx 3$s where the obstacle avoidance for $\boldsymbol{x}_3$ is activated. When all errors $e_{i,f}$ are equal to each other, the absolute position between each agent will be equal, hence the true formation will be on track. As soon as collision avoidance is activated for $\boldsymbol{x}_3$ at $t \approx 3$s, the formation tracking suffers. The target error $e_{3,t}$ has temporarily become the avoidance reference and diverges slightly from the two other agents. The formation errors are increasing as agent $\boldsymbol{x}_2$ starts performing avoidance at $t \approx 5$s. The target errors $e_{2,t}$ and $e_{3,t}$ increase at $t \approx 8$s where the formation errors are largest. This is a result of the equal weighted position states in the controller. As the agents pass the obstacle, $e_{i,f}$ and $e_{i,t}$ go back to the initial values prior to avoidance at $t \approx 2$s. The switching point between temporary obstacle avoidance and target tracking can be seen at $t \approx 14$s.

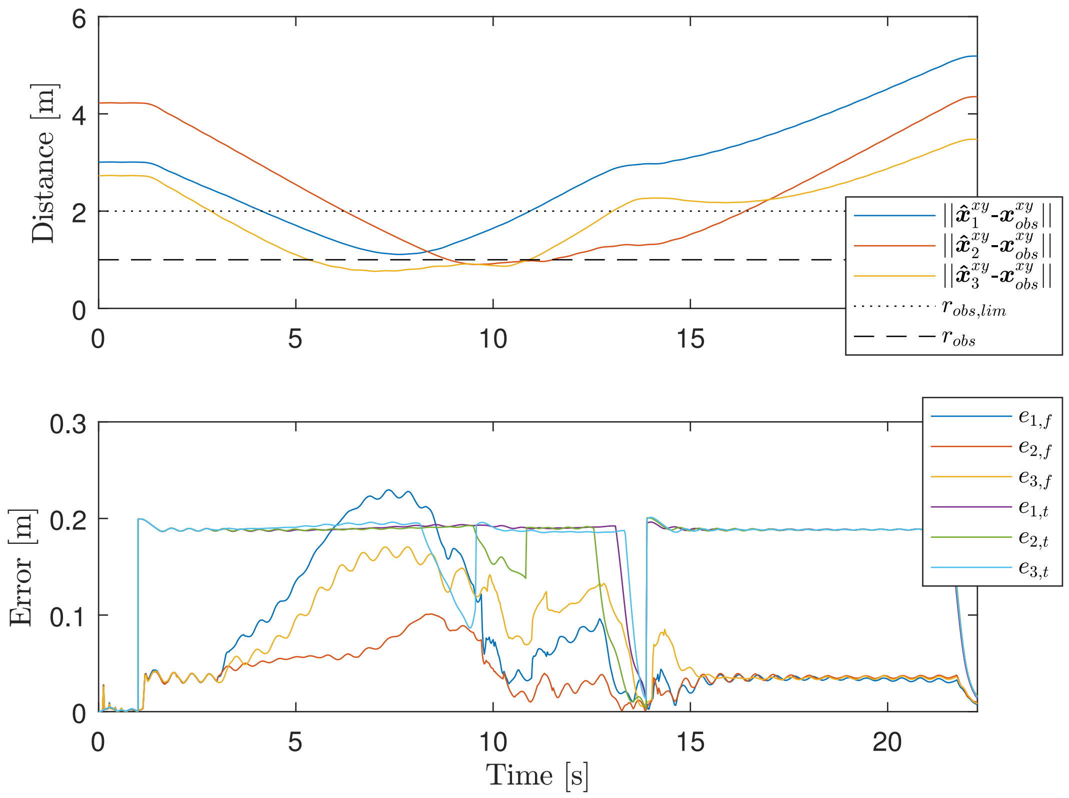


Figure 4.14: Performance of static collision avoidance together with formation flying in time for agent to obstacle distance and tracking errors.

4.1.5 Inter-Vehicle Collision Avoidance

IVCA is the collision avoidance between two dynamic agents as described in Section 3.3.2. The scenario given to evaluate the algorithm is two agents starting in two adjacent corners of a square, travelling in diagonal of the square to reach the corners $\boldsymbol{x}_{1,tref}$ and $\boldsymbol{x}_{2,tref}$, as presented in the top plot in Figure 4.15. An animation is presented in Figure A.4 in Appendix A. As agents have the same target reference ramp functionality that pushes the target reference ahead with the same magnitude, they would meet each other at the origin. Using the IVCA algorithm, the performance of decentralized control strategy is evaluated in Figure 4.15, where in the top plot one can see the performance in XY plane from above and on the bottom one, distance between them in 2D as time goes. Looking into the top plot, one can see that as they arrive into each others vicinity, agent $\boldsymbol{x}_2$ is the one which is going to slow down. Although agents have the same velocity up to some certain precision, the reason for which $\boldsymbol{x}_2$ is the one that reacts is that it has lower priority given its agent number; as the algorithm suggests. Should $\boldsymbol{x}_2$ have higher velocity than $\boldsymbol{x}_1$,$\boldsymbol{x}_1$ would be the one that reacts instead, regardless to their priority suggested by their agent numbering. Also looking into the which path $\boldsymbol{x}_1$ has taken, one can see a slight change in target reference. This is due to that target references are pushed outside the critical region r_{obs} of the obstacle, so even though $\boldsymbol{x}_2$ is the one reacting, $\boldsymbol{x}_1$ target reference would take a small detour as well to not land in the critical region. Looking into the bottom plot, one can see that the agents never come closer to each other than the $r_{obs,lim}$ which is the ideal behaviour for the algorithm.

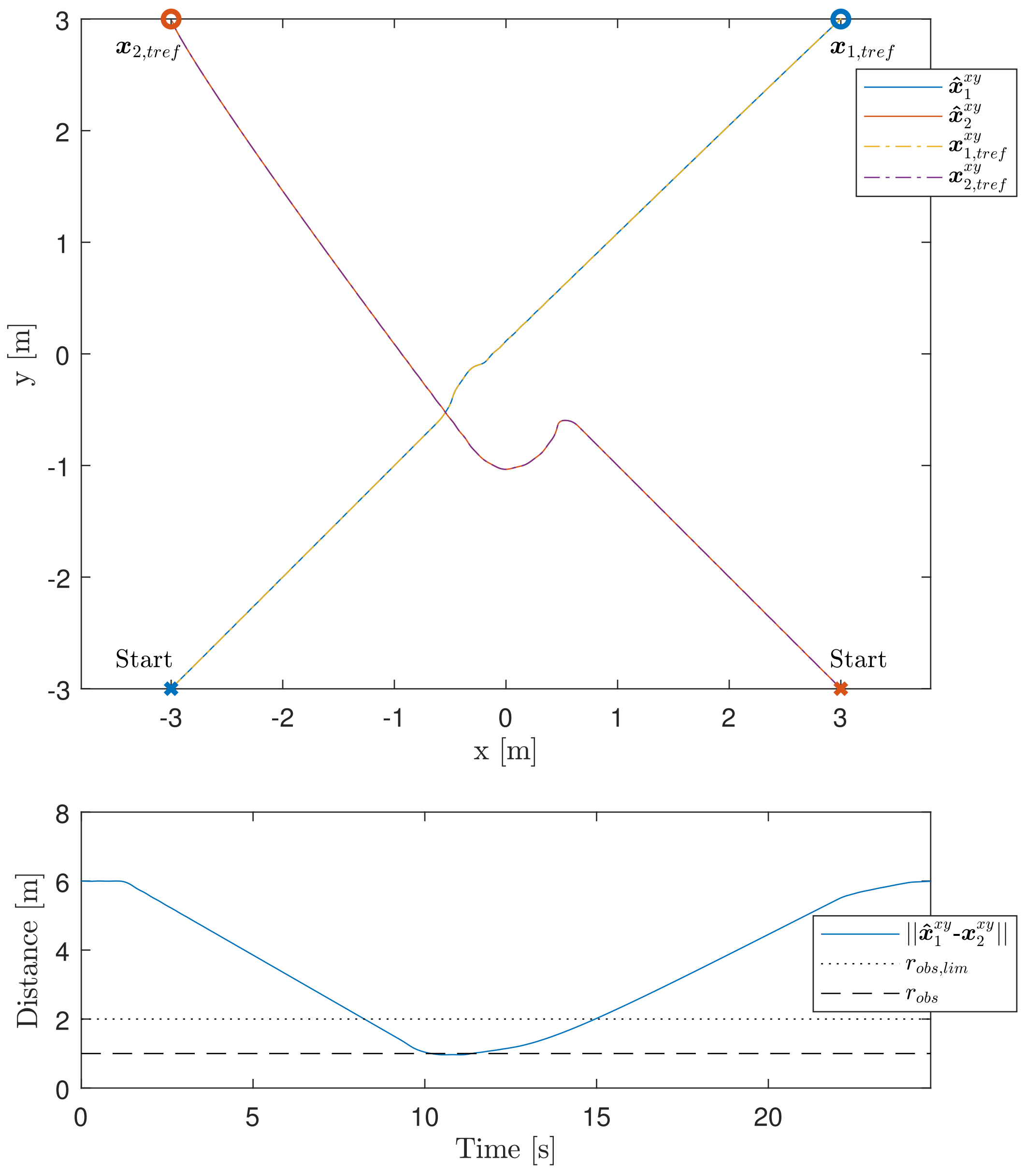


Figure 4.15: Performances of static collision avoidance with a single agent.

The algorithm is also tested for distributed control strategy and it does not show any better performance than the decentralized control strategy. As the references use the ramp functionality, the prediction of the agents at each step would suggest that they would slow down as they are arriving to this references in close future. However that is not the case as the references are moving further next step continuously because of the nature of reference ramp. Therefore they are not slowing down in reality. This phenomenon would make the target states prediction unworthy and unreliable. This is a drawback with using the reference ramp functionality in that it would make the communicated predictions unworthy for distributed fashion. As an extended study, one can investigate having two sets of formation flying agents that two agent, each from one of these sets, are closing in to a collision. In that case they have the opportunity to communicate the formation states predictions which do not suffer from the phenomenon with reference ramping that makes the prediction states unworthy. This is also mentioned in the Future Work.

4.2 Implementation

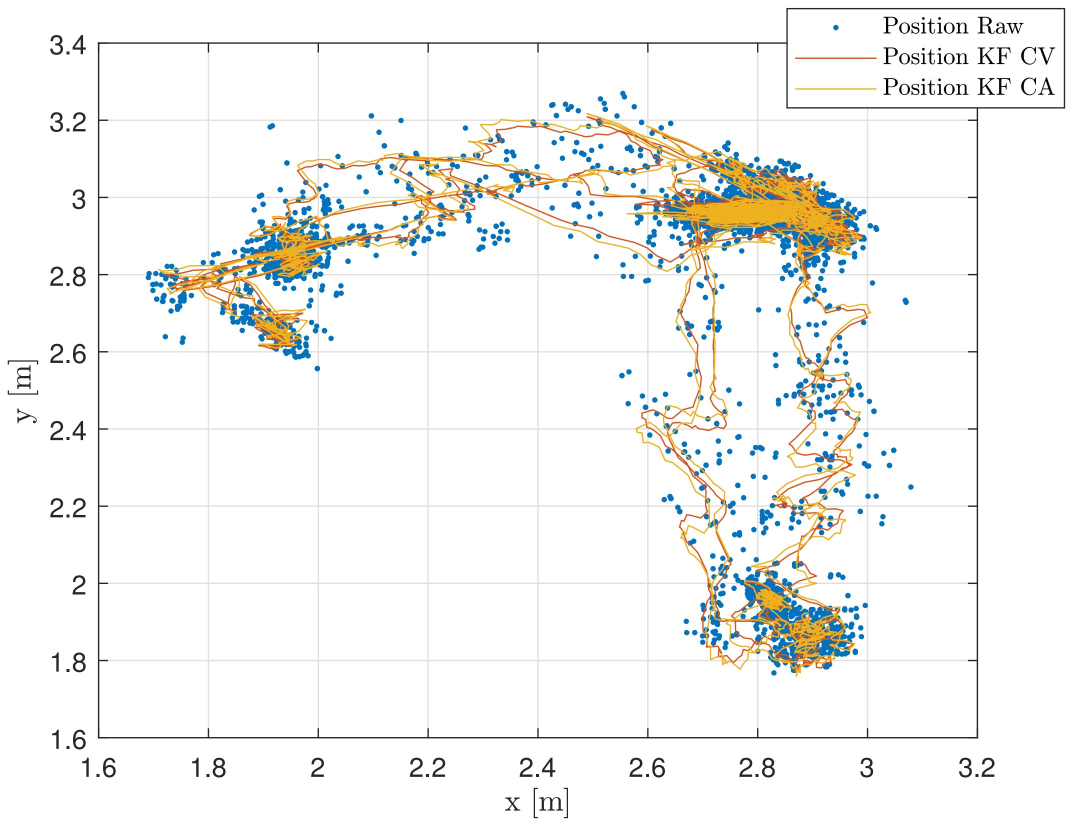
This section covers the resulting performance of the real time implementation on a single quadcopter. The section starts off by introducing and evaluating the indoor positioning system with respect to absolute and relative accuracy. Further, the test scenario is given. This includes an overview of the overall system and actual flight results with evaluation of initial reference and step response tracking. Finally, suggestions on necessary improvements are given.

4.2.1 Positioning System

The positioning system is based on the Localino v2.0 kit presented in Section 3.7.1, consisting of four static anchor points and one mobile tag mounted on the quadcopter. Each anchor measures the distance between itself and the tag. The resulting four measurements are passed in to a recursive trilateration algorithm which then returns the coordinates of the tag [17]. The estimated position is further filtered using the KF from Section 2.2.1. Two different motion models are used for prediction; the *Constant Velocity (CV)* and *Constant Acceleration (CA)* models. To verify the absolute and relative accuracy of the positioning system, the quadcopter is manually moved on a straight line in the XY plane between the starting point $(3, 3)$ and the two end points $(2, 3)$ and $(3, 2)$. The resulting measurements in both the XY plane and in time are presented in Figure 4.16. From the XY plane, results shows that the absolute accuracy with respect to the measured reference points have an error of $\approx (\pm 0.2, \pm 0.2)$. During translation, similar variations of $\approx (\pm 0.2, \pm 0.2)$ are visible. The absolute accuracy is related to overall the accuracy of anchor positions in the global frame. Any slightest offset from true position can result in errors in the estimated absolute position. The relative accuracy is good, where the total translations of $\approx \pm 1$m in both X and Y directions are achieved. From the time plot, the differences between just using the raw position estimation and the KF becomes visible. Table 4.3 presents the variances during static position measurements at time 90-120s. The filtered variance is considerably more dampened compared to the raw positions. In particularly the x position. The differences between the two motion models CV and CA are not visible in the plots. However, during actual flight, the CV introduces a slight time delay in filtered positions compared to actual true position. This gives the quadcopter lagging information which accumulates in to larger error in position reference tracking. Hence, the CA motion model is preferred for the KF.

Table 4.3: Raw and filtered position measurement variances at time 90-120s.

Measurement Type	σ_x^2 [m]	σ_y^2 [m]
Raw	0.0105	0.0066
KF CV	0.0071	0.0062
KF CA	0.0073	0.0063



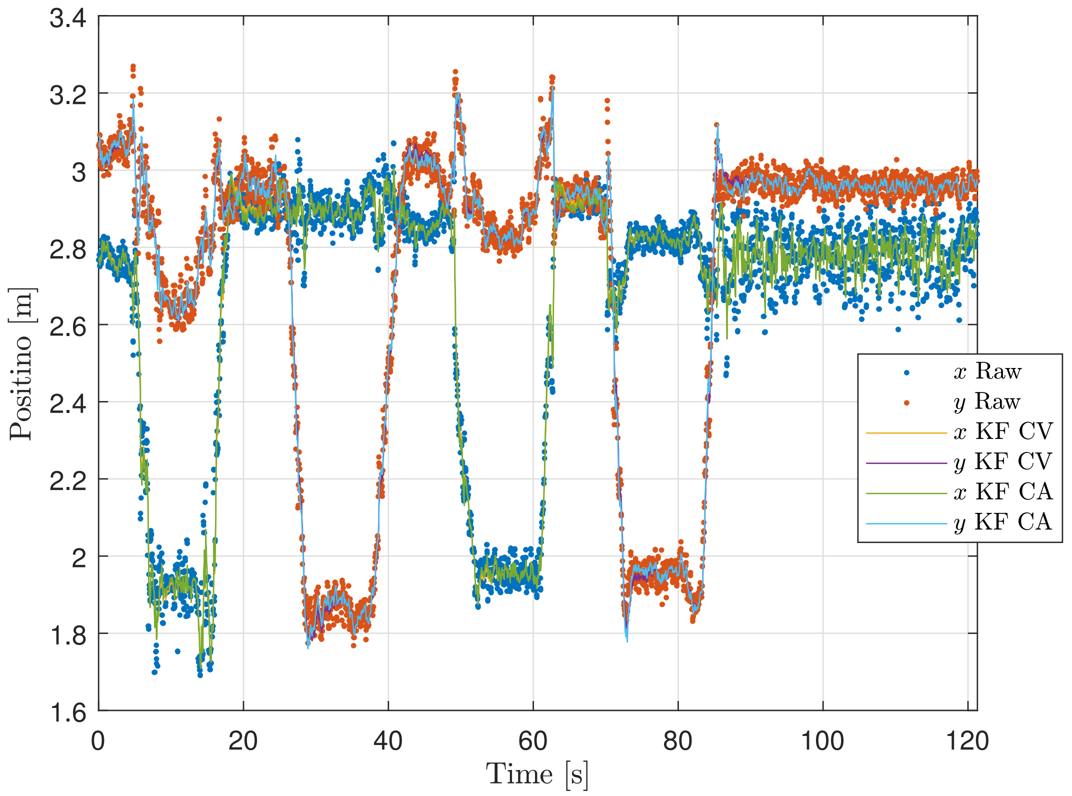


Figure 4.16: Performance of positioning system in both XY plane and time.

From [17] the trilateration algorithm also provides a altitude estimation for the Z position. However, the estimation turned out to be highly non reliable and not suitable for reference tracking. For the coming section on flight results, the altitude is manually controlled via the total thrust produced by all four motors.

4.2.2 Real Time Stationary Flying and Step Response

The real system implementation presented in Section 3.7 is tested on a single quadcopter. The goal of the test is to evaluate the performance of the MPC based scheme on initial reference point and step response tracking. A standalone computer handles the trilateration algorithm for position estimations, which are then wirelessly passed to the quadcopter. For the flight test, the quadcopter is initialized at position $(3, 2)$ in the XY plane. As a results of not having appropriate altitude measurement, manual thrust is given to the quadcopter in order to lift of. At time $\approx$ 2.6s, the reference is stepped to position $(3, 3)$. At time $\approx$ 6.1s the reference is stepped back to the position $(3, 2)$. The controller weights are set according to Table 4.4. As the altitude controller is not used, associated parameters are not relevant. In order to keep real time system performance of the overall implemented solution, the prediction horizon is $T = 20$ for position controller and $T = 10$ for attitude controller.

Table 4.4: Controller setup throughout real time flight.

	Position controller	Attitude controller	Altitude controller
$\boldsymbol{Q}$	$\begin{bmatrix}750\\500\\750\\500\\1\\1\end{bmatrix}\boldsymbol{I}_6$	$\begin{bmatrix}1500\\100\\1500\\100\\100\\300\end{bmatrix}\boldsymbol{I}_6$	-
$\boldsymbol{R}$	$\begin{bmatrix}2000\\2000\end{bmatrix}\boldsymbol{I}_2$	$\begin{bmatrix}1\\1\\100\end{bmatrix}\boldsymbol{I}_3$	-
T	20	10	-
κ	0.001	0.001	-
K^{max}	5	5	-

Figure 4.17 presents the resulting position tracking performance in time. Both x and y position tracking have fluctuating errors of $\pm$0.4m. The position response to the step change in reference is sharp. The quadcopter travels $\pm$1m in y position in $\approx$ 0.3s.

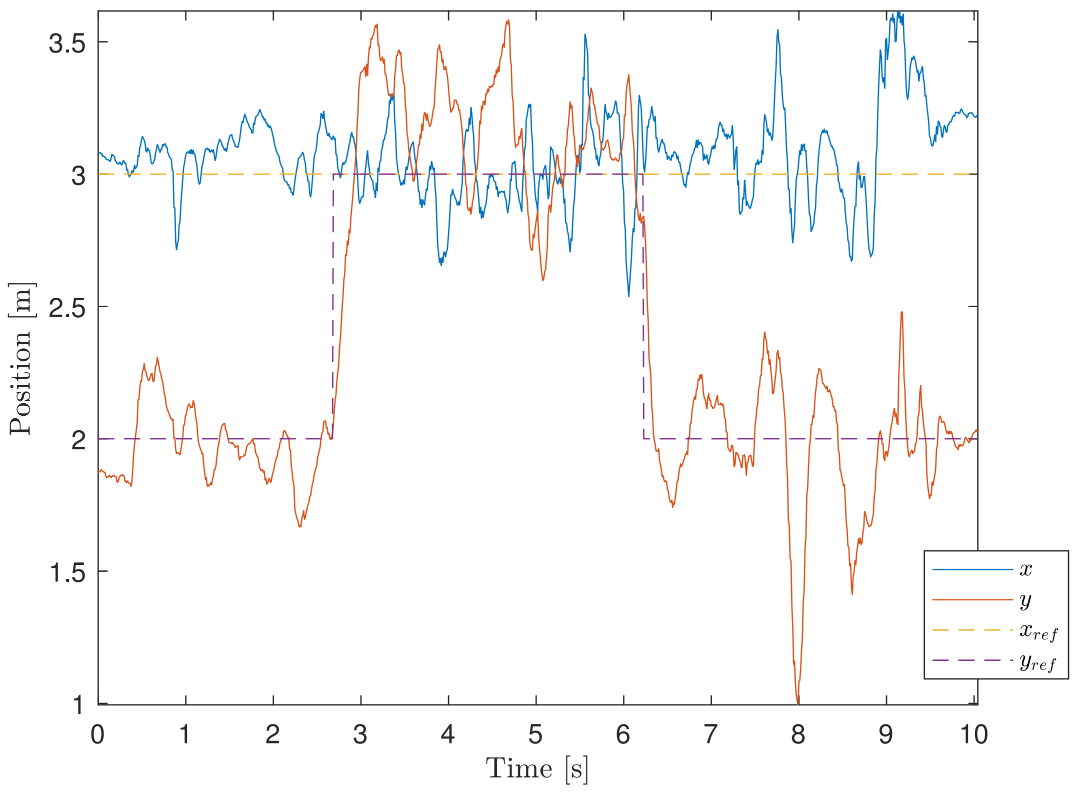


Figure 4.17: Performance of position reference tracking on step response.

The accuracy of the positioning system has a major impact on the overall tracking performance and needs to be taken in to account when evaluating the performance. The underlying position estimation error of ±0.2m in XY plane will create fluctuation in reference tracking. Another important factor is the manual thrust given to the system in order to keep altitude. The MPC position model from (3.45) is based on a hovering quadcopter where thrust is equal to the gravitational force. In reality, the manual thrust creates variance in altitude, hence the position predictions within the MPC algorithm can be misleading. The last factor to consider is accuracy of the motor constant c_m. The parameter estimation in Section 3.6.2 is highly related to the battery voltage level. As the voltage drops over time during flight, the quadcopter response to actuator changes is weakened. Figure 4.18 shows how the manual thrust is increasing during flight time when in reality, the altitude is not increasing. The increase in position error over time can therefor be connected to the dropping battery voltage.

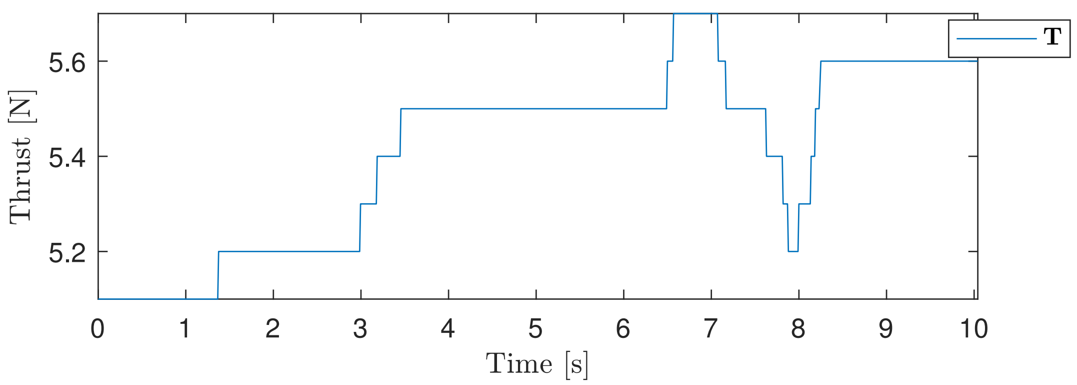


Figure 4.18: Manual thrust $\boldsymbol{T}$ for altitude control.

Further, the accuracy of attitude tracking is directly related to position tracking performance since the position controller passes the necessary angles to the attitude controller in order to reach any given position reference. Figure 4.19 presents resulting attitude control performance with associated torques. Both roll and pitch references $\bar{\phi}_{ref}$ and $\bar{\theta}_{ref}$ are heading compensated according to (3.49). The overall attitude tracking is considered good. A slight time delay between reference and state is present for both ϕ and θ. Notice how the reference step change in position y is visible at time ≈ 2.6s. $\bar{\phi}_{ref}$ is mainly affected as it is the driving angle in achieving change is position y. Also $\bar{\theta}_{ref}$ has an increase in activity with respect to reference tracking. This is suspected to be due to nonlinearities in the quadcopter such as unevenly distributed mass, combined with decreasing battery effect and resulting mismatching position model parameters. Figure 4.20 presents these nonlinearities in terms of small heading offsets. As a result, the heading compensation forces both attitude angles ϕ and θ to actively work. To each attitude angle plot, the associated torques are also presented. For heading control, the torque τ_ψ comes directly from the attitude MPC. For yaw and pitch control, both the torques $\tilde{\tau}_\phi$ and $\tilde{\tau}_\theta$ are feed forward disturbance compensated with integration action according to (3.55). The disturbance compensation is clearly active compared to the non compensated torques τ_ϕ and τ_θ. The nonlinearities of mass and mismatching motor constant are compensated for.

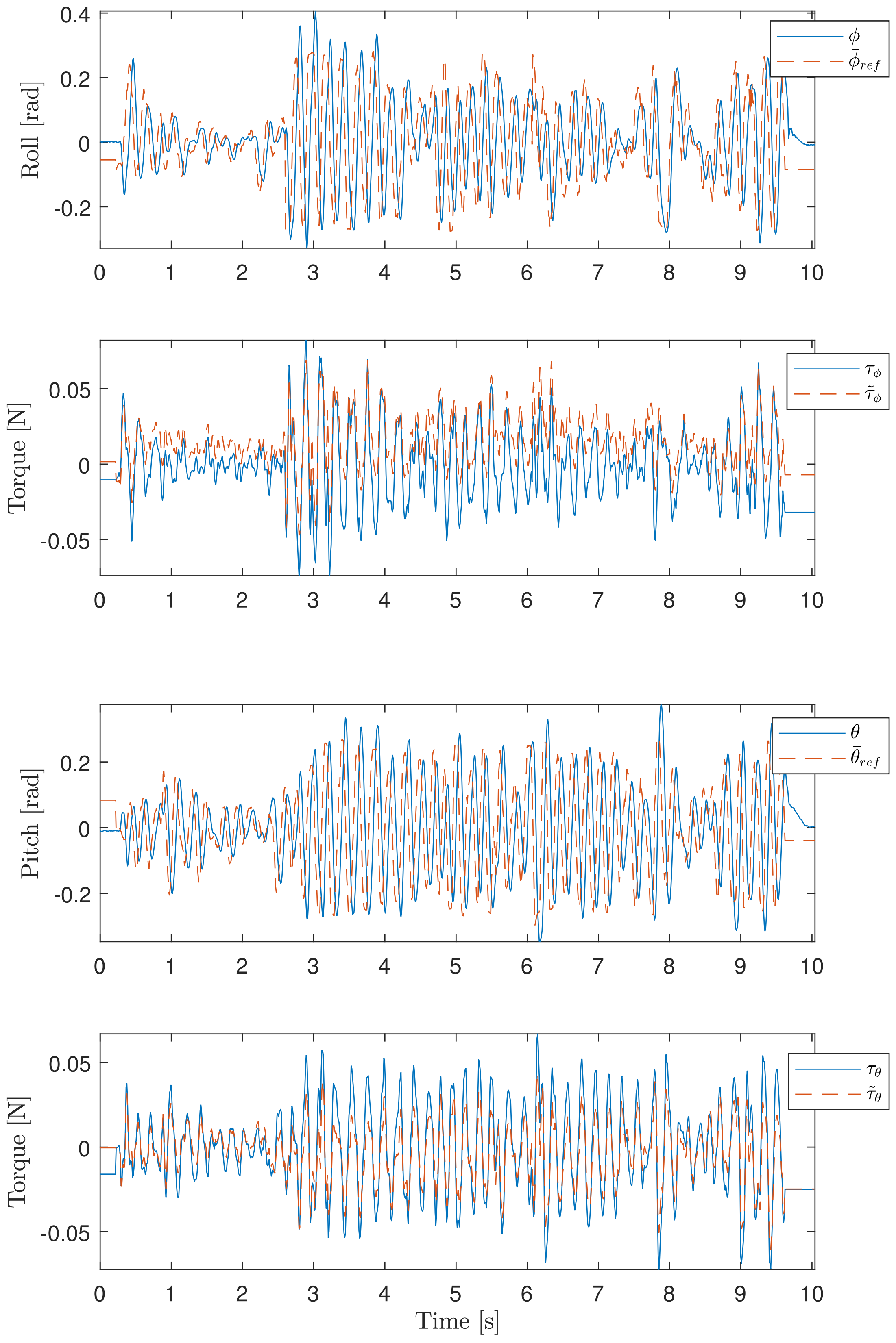


Figure 4.19: Performance of attitude ϕ and θ reference tracking on step response.

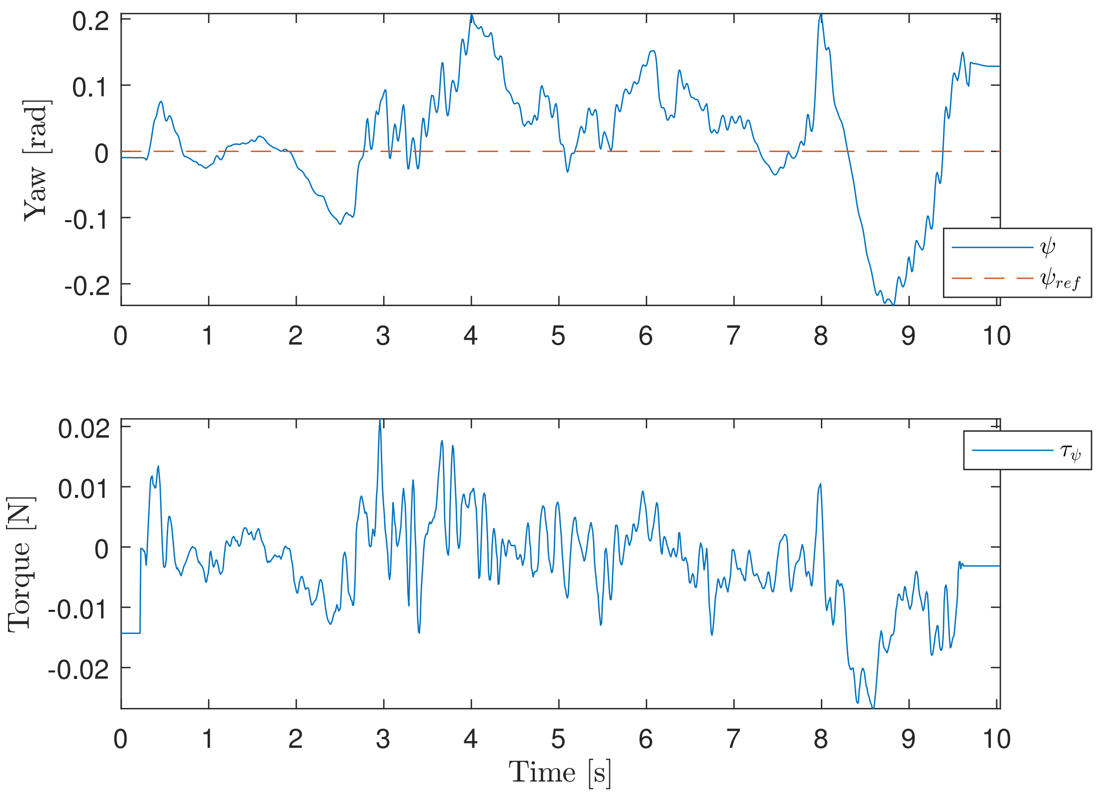


Figure 4.20: Performance of attitude ψ reference tracking on step response.

To summarize the results; regardless of the fluctuating position tracking, the overall performance is considered to be acceptable with respect to the low cost hardware used. The position controller is strongly affected by the inaccuracy of the indoor positioning system. The position controller does its best in order to keep the desired references by passing the necessary angle references to the attitude controller. The attitude tracking is considered to be sufficiently good, hence the resulting position tracking would be less fluctuating if the position measurements were better. Disturbance compensation using integral action is effective.

4.2.3 Suggestions on Improvement

The major bottleneck on better position tracking is the indoor positioning system, hence an upgrade here would benefit the overall system. Further, introducing stronger batteries in order to remove the rapidly decreasing voltage level would improve actuator performance and stabilize model parameters and overall nonlinearities. Lastly, closed loop altitude control would increase reliability of position MPC prediction model.

5
Conclusion

This thesis investigates using MPC in a distributed fashion in order to achieve coordinated behaviour in a multi-agent system, where the proposed application is a system of multiple quadcopters which would together lift a hanging payload from one point to another. The coordinated task would translate to formation flying between the agents with the purpose of keeping a desired payload orientation. Coordinated tasks performed by multiple agents would improve the performance when their decisions are in harmony. The coordinated task can in a systematic way be be included by means of modelling and decision making. The MPC approach, does not only benefit from being a model-based optimization controller in delivering an optimum solution, but also gains from the fact that it delivers a candidate optimum path into the future given the model. In context of multi-agent systems, algorithms and intelligence are to maintain coordination and harmony between the agents. They rely on sharing their states and given the benefit of the MPC optimum candidate path, they can be informed about each others future potential states. However validity of the future states rely on accuracy of the model, consequently these approaches would not be as beneficial unless the model is representing the real system to a decent degree of accuracy.
A nonlinear model is exploited representing the quadcopter, together a model of the payload dynamics. Model parameters were initially identified based on offline experimental data. However, these estimations proved to be not accurate enough for the purpose of the model-based controller. Consequently, new parameters were identified online resulting in improvement of the control performance. This emphasises the fact that initial rough estimations do not necessarily hold in reality.
The control scheme is based on three decoupled MPCs; altitude, position and attitude control. There are several benefits of decoupling the controllers over using a single, full state control scheme. Firstly, tuning a coupled controller is of big challenge in that e.g. holding zero attitude is in a way contradictory to manoeuvring in the XY plane. Secondly, the computational demand can be improved by using multiple, decoupled controller with smaller dimensions compared to running a coupled controller with large dimensions. This further benefits that each decoupled system can be sampled at different frequencies.
An EKF based observer using the nonlinear motion model is used for state and disturbance, motivated by the fact that a nonlinear model has more information regarding the system than a linearized model. Two different approaches to the observer design is proposed; one with the full state and disturbance estimation using the complete nonlinear quadcopter model and another decoupled version in order to improve real time computational demand. Observers are necessary in that they in-

form about non-measured states and disturbances. Initially, disturbance estimation was based on an approach with augmenting the model with a disturbance term and attempt to estimate such disturbances. While the approaches did work flawlessly in simulation, their credibility were more questioned when implemented in reality due to model inaccuracy and instability in estimations. This rises the question that if model-based disturbance compensation really worth the effort. As a result, a more conventional approach using feed-forward disturbance compensation is proposed. Using feed-forward in parallel to the decoupled MPC arises another question. Since the feed-forward term is added in parallel to the MPC, actuator constraints would be blind to such terms and would no hold. It is proposed to correlate the decoupled controllers with the feed-forward terms by actively changing the MPC input constraints of each controller, in order to adapt the actuator constraints. In this way each MPC would deliver the optimal control actions based on the available input range if the range is modified by the feed forward term.

Path-planning and in particular formation flying, requires the agents to dynamically set new references. A solution based on augmenting the position controller with copies of the position states is proposed. This allows a set of multiple references for the same physical state to be followed, in particular target and formation tracking. The disadvantage of this approach is that the velocity states are still coupled to both position states. As a result, the MPC position predictions are not solely mapped back to each position states individually. Collision avoidance is also proposed in order to improve reliability of the autonomous system. Algorithms for both static and inter-vehicle collision avoidance are based on temporary changing target references to go around an obstacle. There are a couple of disadvantages with this approach. The performance of collision avoidance would suffer; firstly if target references are not being tracked adequately and secondly, if the formation and target tracking are given equal weights and where using only formation references would result in a collision.

The MPC has a computation burden in being realized for today's embedded computers, but as the computers get stronger and cheaper, MPC approaches get more realizable. However, there exists methods on approximating the MPC solver in order to make it less demanding and still deliver a sufficient performance. In this work, a less demanding FMPC solver is used to verify the proposed methods in simulation and more importantly, implementation on real hardware. This work takes proud in implementing the MPC architecture on a cheap computer like RPi, exploiting the approximations of the FMPC. A further study can investigate if the sampling frequency requirements of a model-based approach is as frequent as more conventional approaches like PID.

A benchmark between the distributed and decentralized control strategies shows advantages of sharing predictive state information between the agents, in particular where there is communication delay. In a scenario including both target and formation tracking, the distributed control strategy shows an improvement of 50-690% for formation tracking error when the communication delay varies from 2-15 time steps, and improvement of 10-40% for target tracking time of arrival, when the communication delay varies from 1-15 time steps. In another scenario where the agents are on target but still actively keeping formation reference, the distributed control

strategy has 8.8-31.8 times more dampening effect on oscillating behaviour. This highlights the strength of the predictive control where it compensates for increasing communication delays and minimizes formation error over time.
Many approaches to control problems, including this work, are based on linearizing the process model in designing the control algorithm. As a result, the prediction information from the MPC is subject to approximations and do not match the reality. This is beside the fact nonlinear models are another approximation of the reality in themselves before linearization. This emphasizes the limitation on prediction accuracy after some certain horizon. Should it make the solution to formation flying better using more precise models in MPC, higher order linearization of the model, or even better, nonlinear MPC can be used. The credibility of the claim that it would make coordinated multi-agent algorithms better, and if *it is really worth it* to make the effort in using higher order approximations or designing much more complex controllers such as nonlinear MPC, is still a question.

6
Future Work

This chapter gives an overview of potential future work proposed in order to improve and continue on the work presented in this thesis.

- Compare the results from the decentralized and distributed control strategies with a centralized control strategy.
- Use the payload model and evaluate the resulting connector tension as it changes along with the varying formation error. Include this tension in to the simulation and evaluate the performance given this varying disturbance.
- Improve the orientation estimation with respect to the present heading drift. Evaluate the need of upgrading the magnetometer or the use of an appropriate filter to cut the surrounding indoor magnetic noise.
- Study effect of time-varying weights for formation flying without a reference ramp. The weights can be proportional to the errors of target and formation perhaps with more strength on weighing formation states.
- Another feature to test with time-varying weights is in terms of different controller's behavioural or mode. For an instance, target tracking gets substantially higher weights compared to formation tracking as the obstacle avoidance mode is activated; in case formation tracking would suggest a point inside the critical region of an obstacle.
- Investigate inter-vehicle collision between two sets of formation flying agents where the formation states are shared between the agents.
- Increase indoor position system quality and evaluate the complete multi-agent scenario in real time, using three quadcopters.

7
Sustainability and Ethics

Model and optimization based solutions are beneficial in terms of being optimum solution. MPC design is based on optimization problems and this has the advantage of being optimized to a number of objectives. Among such objectives, it is typical to take into account the energy consumed by the system. This results in minimizing the energy consumed by the actuators, while the controller delivers the task. MPC and model based approaches in general, are serving as a solution to technical problems nowadays, more than ever before. Research on the topic, such as this work that takes into account the implementation, will pave the way for further development and make the theory realizeable in industry far more than already today.

The application of this work, quadcopters or more publicly known as UAVs, are already serving as sustainable solutions automating tasks with minimum effort. The potentials of using UAVs in everyday life are vast. They range from delivery of different kinds of products, where the UAVs are carrying products as an autonomous system. Autonomous systems are immune to the factor of human faults as a driver. UAVs can be used in agriculture and with the increase of world population, the industry always foster for more innovative and sustainable ideas which help make plantation for such population feasible. They can be used in search and rescue missions, where they can locate people in danger, e.g. a drowning person or stuck in fire, without putting at risk the lives of rescue teams. UAVs have been tested as a construction aid for building bridges and houses. All these are just a few examples of how they can serve the society with minimum risk and energy.

Bibliography

[1] S. S. Mansouri, G. Nikolakopoulos, and T. Gustafsson, "Distributed model predictive control for unmanned aerial vehicles," in *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, Nov 2015, pp. 152–161.

[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] S. M. Huck, M. Rueppel, T. H. Summers, and J. Lygeros, "Rcopterx - experimental validation of a distributed leader-follower mpc approach on a miniature helicopter test bed," in *2014 European Control Conference (ECC)*, June 2014, pp. 802–807.

[4] E. Semsar and K. Khorasani, "Adaptive formation control of uavs in the presence of unknown vortex forces and leader commands," in *2006 American Control Conference*, June 1992, p. pp. 373–385.

[5] Z. Chao, L. Ming, Z. Shaolei, and Z. Wenguang, "Collision-free uav formation flight control based on nonlinear mpc," in *2011 International Conference on Electronics, Communications and Control (ICECC)*, Sept 2011, pp. 1951–1956.

[6] O. M. Agudelo and B. D. Moor., "Computergestuurde regeltechniek exercise session case study : Quadcopter," 2016. [Online]. Available: http://homes.esat.kuleuven.be/~maapc/static/files/CACSD/exercises/Session%203/quadcopter_exercise.pdf

[7] D. J. L. Mapopa Chipofya and K. T. Chong, "Trajectory tracking and stabilization of a quadrotor using model predictive control of laguerre functions,," in *Abstract and Applied Analysis, vol. 2015*, 2015.

[8] W. U. M. Group, "The mathematics of the stewart platform." [Online]. Available: https://web.archive.org/web/20130506134518/http://www.wokinghamu3a.org.uk/Maths%20of%20the%20Stewart%20Platform%20v5.pdf

[9] N. Michael, J. Fink, and V. Kumar, "Cooperative manipulation and transportation with aerial robots," *Autonomous Robots*, vol. 30, no. 1, pp. 73–86, 2011. [Online]. Available: http://dx.doi.org/10.1007/s10514-010-9205-0

[10] B. R. Albus, J. and N. Dagalakis, "The nist spider, a robot crane," vol. 97(3), p. pp. 373–385, 1992.

[11] S. R. Oh and S. K. Agrawal, "A control lyapunov approach for feedback control of cable-suspended robots," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 4544–4549.

[12] C. LLC, "Using accelerometers to estimate position and velocity." [Online]. Available: http://www.chrobotics.com/library/accel-position-velocity

[13] A. H. Toshak Singhal and D. N. Vishwakarma, “Kalman filter implementation on an accelerometer sensor data for three state estimation of a dynamic system,” *International Journal of Research in Engineering and Technology (IJRET)*, vol. 1, pp. 330–334, 2012.

[14] Z. G. Sławomir Romaniuk, “Kalman filter realization for orientation and position estimation on dedicated processor,” *acta mechanica et automatica*, vol. 8, pp. 330–334, 2014.

[15] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan, “Estimation of imu and marg orientation using a gradient descent algorithm,” in *2011 IEEE International Conference on Rehabilitation Robotics*, June 2011, pp. 1–7.

[16] S. O. Madgwick, “An efficient orientation filter for inertial and inertial/magnetic sensor arrays.” [Online]. Available: http://www.x-io.co.uk/res/doc/madgwick_internal_report.pdf

[17] A. Norrdine, “An algebraic solution to the multilateration problem,” in *2012 International Conference on Indoor Positioning and Indoor Navigation*, nov 2012.

[18] J. Rawlings and D. Mayne, *Model Predictive Control: Theory and Design*. Nob Hill Pub., 2009. [Online]. Available: https://books.google.se/books?id=3_rfQQAACAAJ

[19] Y. Wang and S. Boyd, “Fast model predictive control using online optimization,” *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, March 2010.

[20] S. Särkkä, *Bayesian Filtering and Smoothing*, 2013.

[21] P. O. M. Scokaert and J. B. Rawlings, “Constrained linear quadratic regulation,” *IEEE Transactions on Automatic Control*, vol. 43, no. 8, pp. 1163–1169, Aug 1998.

[22] R. Nave, “Common moments of inertia.” [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/mi.html#mi

[23] T. L. Foundation, “Howto setup linux with preempt_rt properly,” Oct 2017. [Online]. Available: https://wiki.linuxfoundation.org/realtime/documentation/howto/applications/preemptrt_setup

A
Appendix

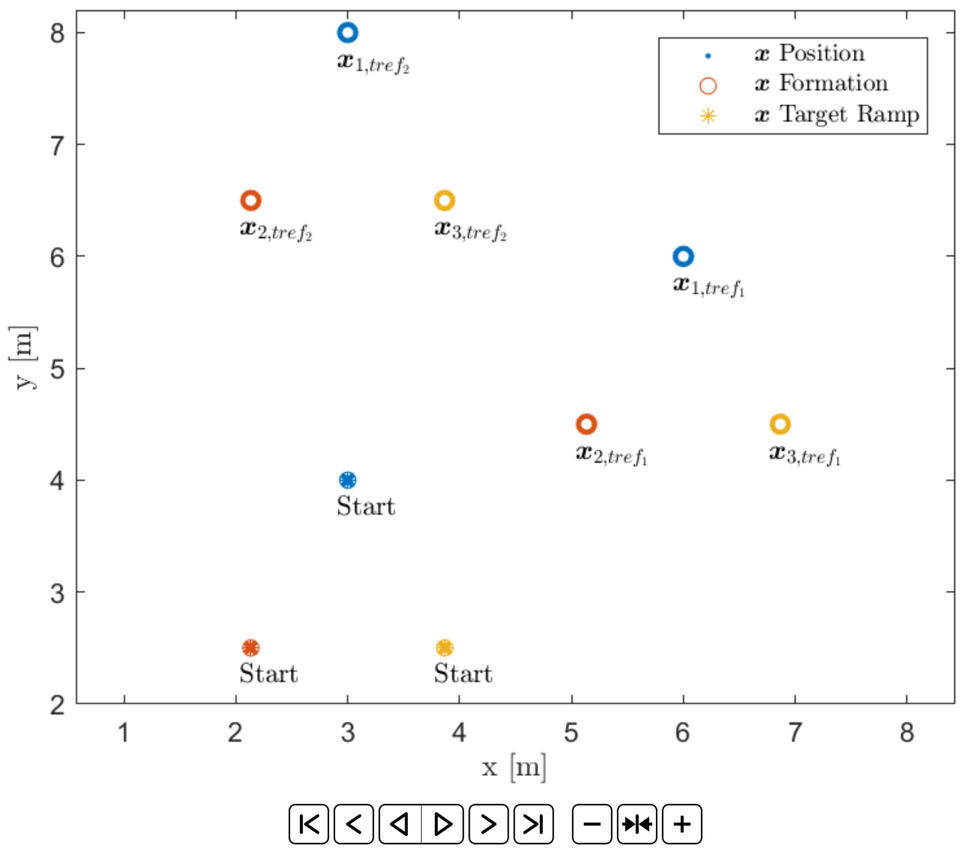


Figure A.1: Benchmark scenario of formation flying.

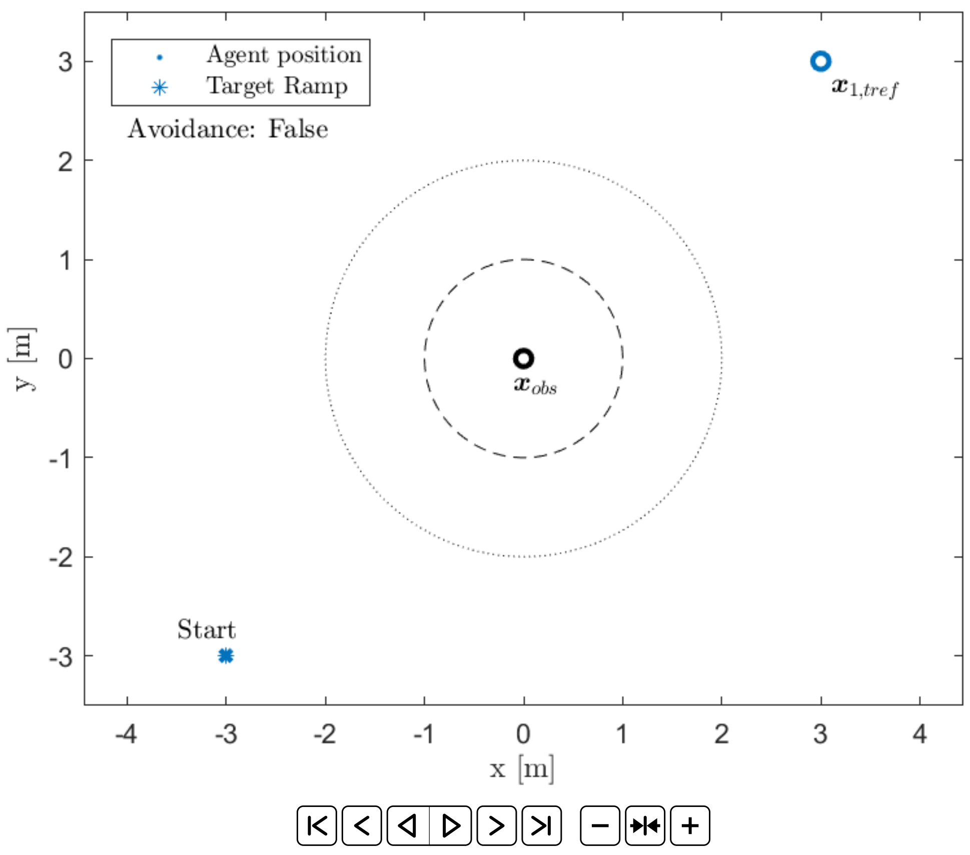


Figure A.2: OCA of static obstacle.

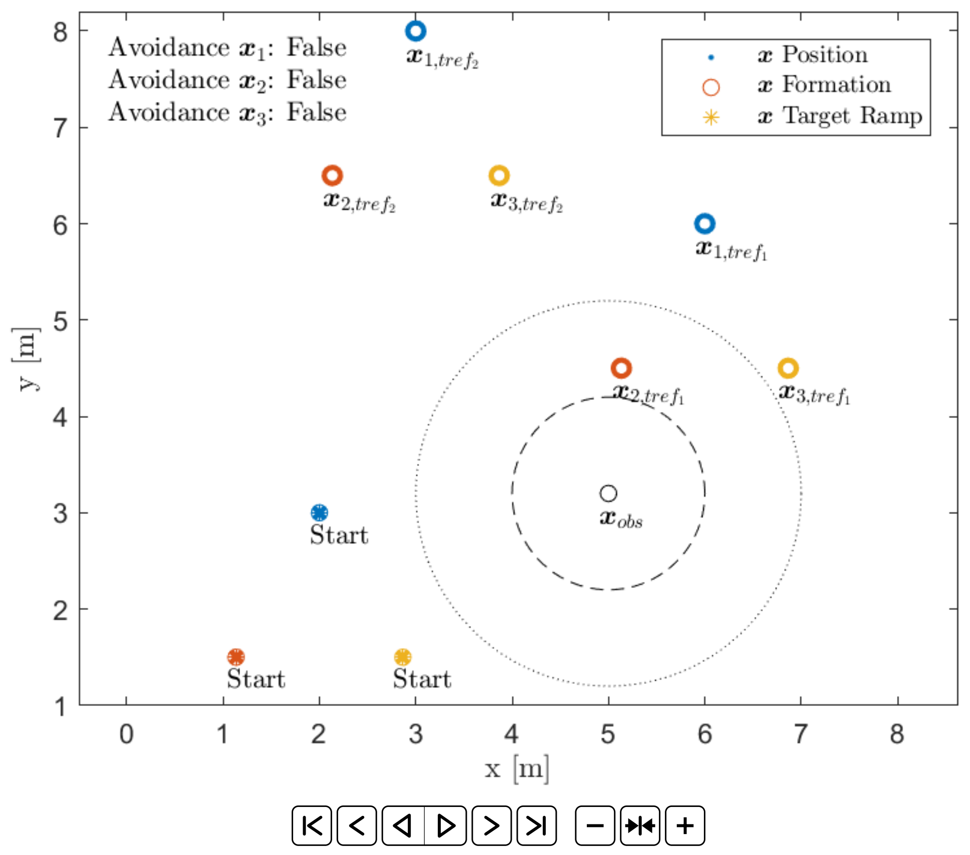


Figure A.3: OCA of static obstacle during formation flying.

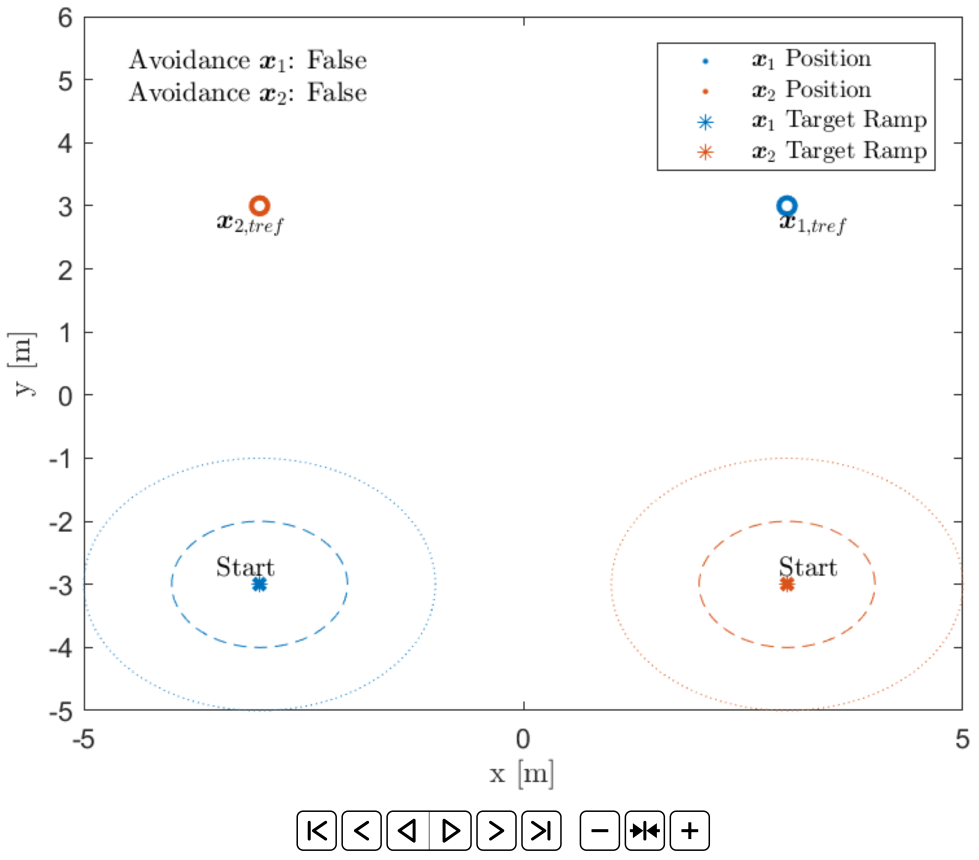


Figure A.4: IVCA during formation flying.