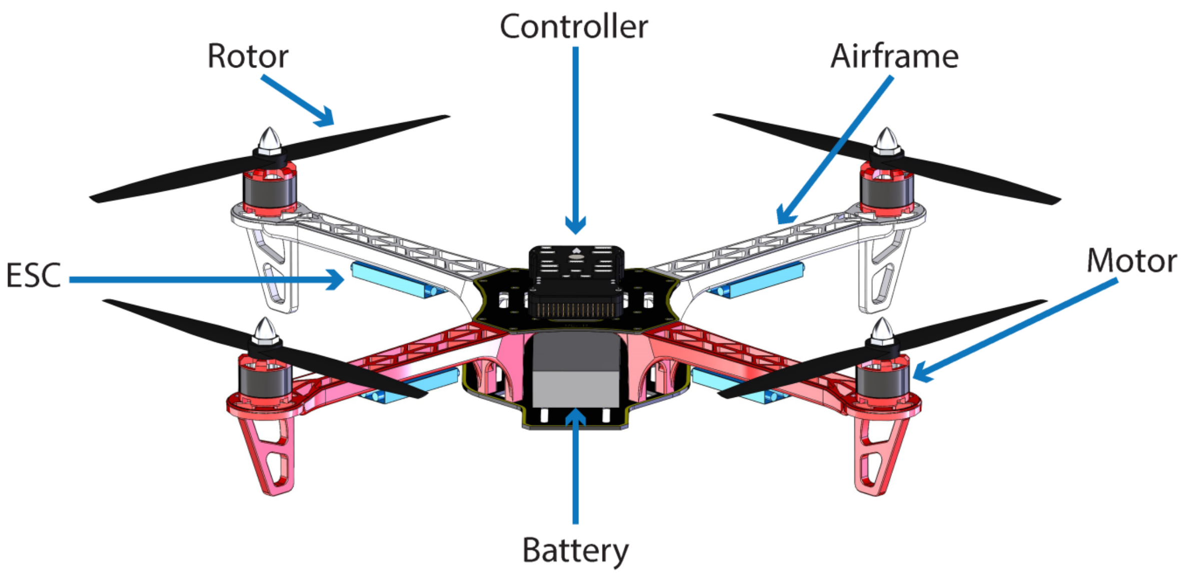
MODEL-BASED DEVELOPMENT AND EVALUATION OF CONTROL FOR COMPLEX MULTI-DOMAIN SYSTEMS: **ATTITUDE CONTROL FOR A QUADROTOR UAV**

Electrical and Computer Engineering
Technical Report ECE-TR-23



AARHUS UNIVERSITY
DEPARTMENT OF ENGINEERING

DATA SHEET

Title: : Model-Based Development and Evaluation of Control for Complex Multi-Domain Systems: Attitude Control for a Quadrotor UAV

Subtitle: Electrical and Computer Engineering

Series title and no.: Technical report ECE-TR-23

Authors: Ivan Grujic and René Nilsson - Department of Engineering – Electrical and Computer Engineering, Aarhus University

Internet version: The report is available in electronic format (pdf) at the Department of Engineering website http://www.eng.au.dk.

Publisher: Aarhus University© **URL:** http://www.eng.au.dk

Year of publication: 2016 Pages: 92
Editing completed: January 2016

Abstract: A Cyber-Physical System (CPS) incorporates sensing, actuating, computing and communicative capabilities, which are often combined to control the system. The development of CPSs poses a challenge, since the complexity of the physical system dynamics must be taken into account when designing the control application. The physical system dynamics are often defined within mechanical and electrical engineering domains, with the control application residing in software and control engineering domains. Therefore, such a system can be considered multi-domain.
With the constant increase in the complexity of such systems, caused by technological advances in all domains, new ways of approaching multi-domain system development are needed. One methodology, which excels in complexity management, is model-based development. Multi-domain systems require collaborative modeling, where the physical system dynamics are captured in the Continuous Time (CT) modeling domain and the digital control is captured in the Discrete Event (DE) modeling domain.
This thesis demonstrates how an extended CT-first model-based development approach can be applied to a complex multi-domain system. A collaborative model of a quadrotor Unmanned Aerial Vehicle (UAV) has been constructed and used to develop an attitude controller based on Model
Predictive Control (MPC). The MPC controller has been compared to an existing open source Proportional Integral Derivative (PID) attitude controller.
This thesis contributes to the discipline of model-based development with a methodological extension to the CT-first approach, which extends the conventional approach by expanding the physical modeling process into three consecutive steps. An evaluation of the extension is presented, describing how and when the extended methodology provides increased value.

Keywords: cyber-physicals-systems, modeling, co-model, co-simulation, drone, quadrotor, quadcopter, UAV, model predictive control, MPC, PID, attitude control, model fidelity

Supervisor: Peter Gorm Larsen

Please cite as: Ivan Grujic and René Nilsson, 2016. Model-Based Development and Evaluation of Control for Complex Multi-Domain Systems: Attitude Control for a Quadrotor UAV. Department of Engineering, Aarhus University. Denmark. 92 pp. - Technical report ECE-TR-23

Cover image: René Nilsson

ISSN: 2245-2087

MODEL-BASED DEVELOPMENT AND EVALUATION OF CONTROL FOR COMPLEX MULTI-DOMAIN SYSTEMS: ATTITUDE CONTROL FOR A QUADROTOR UAV

Ivan Grujic and René Nilsson
Aarhus University, Department of Engineering

Abstract

A Cyber-Physical System (CPS) incorporates sensing, actuating, computing and communicative capabilities, which are often combined to control the system. The development of CPSs poses a challenge, since the complexity of the physical system dynamics must be taken into account when designing the control application. The physical system dynamics are often defined within mechanical and electrical engineering domains, with the control application residing in software and control engineering domains. Therefore, such a system can be considered multi-domain.

With the constant increase in the complexity of such systems, caused by technological advances in all domains, new ways of approaching multi-domain system development are needed. One methodology, which excels in complexity management, is model-based development. Multi-domain systems require collaborative modeling, where the physical system dynamics are captured in the Continuous Time (CT) modeling domain and the digital control is captured in the Discrete Event (DE) modeling domain.

This thesis demonstrates how an extended CT-first model-based development approach can be applied to a complex multi-domain system. A collaborative model of a quadrotor Unmanned Aerial Vehicle (UAV) has been constructed and used to develop an attitude controller based on Model Predictive Control (MPC). The MPC controller has been compared to an existing open source Proportional Integral Derivative (PID) attitude controller.

This thesis contributes to the discipline of model-based development with a methodological extension to the CT-first approach, which extends the conventional approach by expanding the physical modeling process into three consecutive steps. An evaluation of the extension is presented, describing how and when the extended methodology provides increased value.

Acknowledgements

We would like to thank our academic supervisor, Peter Gorm Larsen, for attention and guidance during the preparation and execution of this thesis. We would also like to thank our industrial partner, Danish Aviation Systems, for an interesting subject proposal as well as supplying the quadrotor and testbed.

Additional thanks go out to the staff at Aarhus University, Department of Engineering, for their assistance in constructing the test environment including testbed modifications, mounting bracket production and power supply acquisition.

Finally, we would like thank our friends and family for their patience and support as well as constructive criticism in the final stages of writing.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the background and motivation underlying the work of this thesis. To set the context of the subsequent chapters, the thesis goals are presented along with the scope of the thesis and the approach taken. Chapter 2 continues with an introduction to the theory and technologies applied. Finally, chapter 7 provides an evaluation of the thesis goals and presents envisaged future work.

1.1. Overview

A *system* is defined as being "a combination of interacting elements organized to achieve one or more stated purposes" [INCOSEseh15]. *Cyber-Physical Systems (CPSs)*, refer to a subset of such systems which incorporate sensing, actuating, computing and communicative capabilities [Thompson13]. These diverse capabilities require CPSs to incorporate components belonging to different domains, and thus CPSs can be considered as *multi-domain* systems.
An example of a CPS is an *autonomous vehicle*. The autonomous vehicle must compute and actuate an appropriate course of action based on sensing the environment and communicating with other autonomous vehicles or surrounding infrastructure. The welfare of the passengers and environment of an autonomous vehicle depends on the correctness and safety of its operation, which necessitates an operation correctness and safety assurance. Figure 1.1 shows a CPS assurance framework, where a computer-based model is used to capture the behavior of the system as a *conceptualization facet*. The model can be used to aid the assurance process through model checking, formal verification and simulations. However, model-based assurance can only be trusted if there is sufficient congruency between the real system behavior and the model behavior, henceforth referred to as *model fidelity*.

This chapter introduces the background and challenges of multi-domain system development in section 1.2. Next, the authors motivation for choosing the thesis topic is presented in section 1.3. Subsequently, the thesis goals are described in section 1.4 and the scope and approach in section 1.5. Finally, this chapter is concluded with a reading guide in section 1.6 and a presentation of the overall structure of the thesis in section 1.7.

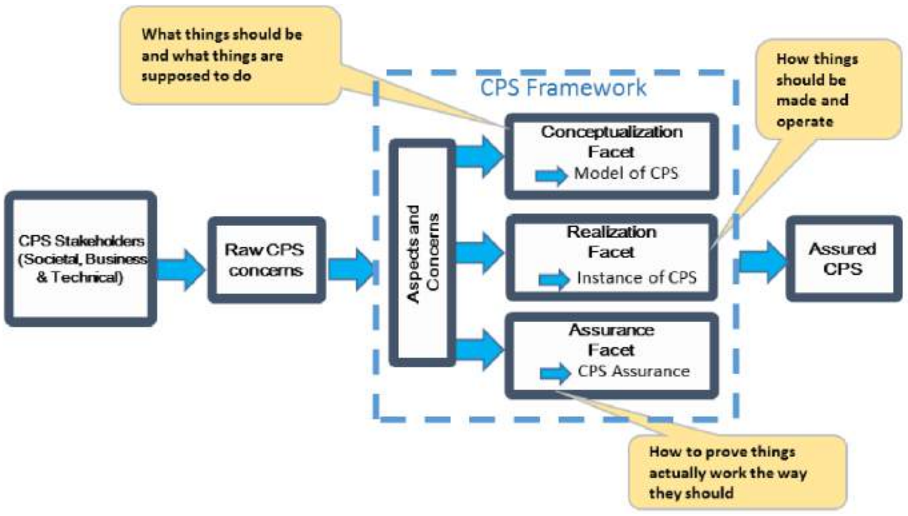

What things should be and what things are supposed to do
CPS Framework
How things should be made and operate
CPS Stakeholders (Societal, Business & Technical)
Raw CPS concerns
Aspects and Concerns
Conceptualization Facet
Model of CPS
Realization Facet
Instance of CPS
Assurance Facet
CPS Assurance
Assured CPS
How to prove things actually work the way they should

Figure 1.1: CPS assurance framework, taken from [NIST15a]

1.2. Background

The multi-domain nature of CPSs causes certain challenges to be associated with their development. The diverse domains and interfaces which comprise a CPS increase the width of technical knowledge required for successful development. This leads to a high level of design complexity, which increases the risk of design errors and makes system-wide consequences of single-domain design decisions harder to identify. Better complexity management can be obtained by constructing abstract cross-disciplinary models, that omit non-crucial components and details. This clarifies design trade-offs which span multiple domains and enable developers to efficiently reason about system-level properties. Cross-disciplinary models are also referred to as collaborative models or *co-models*.

Clear and concise communication between stakeholders from different domains is challenged by differing terminologies - similar terms can have different meanings whilst different terms can mean the same [Gooch05].
A system-wide model can serve as an effective communication medium, during discussions on the boundary between different domains, since it provides a common reference for all system properties. Additionally, a more effective communication effort can be achieved by exploiting whatever model representation is most suitable for the targeted stakeholder.

Even though a model-based development approach is suitable for CPSs, the methodology itself poses challenges as well. Model fidelity is crucial to the value of the model, as no value can be gained from a model that does not exhibit the intended functionality or property of the system, to a *satisfactory* degree. A satisfactory degree of model fidelity is elaborated in section 5.3. This requires the relevant system components to be modeled at a sufficiently high level of detail, which is challenged by system complexity and modeling time constraints. A selective approach guided by the *modeling goal* should be used to determine the suitable level of detail for each component.

As a result of modeling multiple domains, an appropriate domain or domain boundary should be selected as the starting point for the modeling process. Here, multiple factors should be taken

into account; such as model purpose, domain knowledge, location and relevance of system uncertainties etc. In the context of this thesis, two modeling domains are considered: A Discrete Event (DE) domain and a Continuous Time (CT) domain. The DE modeling domain captures system functionality that is executed at discrete time intervals, such as digital computation of control algorithms. The CT modeling domain captures system properties which are continuous in time, such as physical entities including forces, accelerations, voltages and currents etc. Note the difference between a *domain* and a *modeling domain*: The DE modeling domain captures the software and control engineering domains, whilst the CT modeling domain captures the mechanical and electrical engineering domains.
This leaves three possible modeling approaches: DE-first, CT-first and contract-first, which are described further in section 2.5.

1.3. Motivation

The single-domain nature of the authors' backgrounds[1] sparked an interest to explore multi-domain system development and inter-disciplinary collaboration. The authors got acquainted with the concept of modeling as a development tool while attending the *Modeling of Mission Critical Systems* course, which focused on software modeling.
Co-modeling and co-simulation capabilities were made available to the authors with the Crescendo tool, which is the result of an EU funded research project called Design Support and Tooling for Embedded Control Software (DESTECS) [DESTECS09].
This combination of interests and opportunity paved the way for a master thesis involving model-based development of multi-domain systems.
Besides being of personal interest to the authors, a quadrotor Unmanned Aerial Vehicle (UAV) relies on domains that were either known by the authors (electrical, software, control) or that the authors could familiarize themselves with during the preparation of the thesis (mechanical). This knowledge deficiency within the mechanical domain was the main reason for selecting the CT-first approach, in order to obtain a knowledge base wide enough to support the domains of the system to be modeled.
The authors wished to investigate a challenging and relevant technical issue related to quadrotors, and by doing so experience first-hand how a model-based methodology performs on a real-world problem. Danish Aviation Systems (DAS) were involved in the project through an already established collaboration with Peter Gorm Larsen. A Learning Based Model Predictive Control (LBMPC) controller was proposed by DAS since it was a technical challenge they faced at the time.
The proposal was accepted, but due to academic requirements and time constraints, it was decided to focus on a Model Predictive Control (MPC) controller, and postpone the learning based aspects to a later time. The intended learning based related activities are elaborated as future work in section 7.3.2.

1.4. Thesis goals

The overall thesis goals are:

[1] Both authors hold a bachelors degree in electrical engineering

G1 To develop a collaborative model of an existing complex CPS with the purpose of evaluating control strategies, using a CT-first approach.

G2 To suggest an extension to the CT-first methodology.

G3 To evaluate the extended CT-first methodology.

The methodology evaluation of goal **G3** is inspired by a similar case of model-based methodology evaluation by [Jørgensen12]. The overall value of the methodology is determined by assessing it with respect to the Evaluation Categories (EC) listed below. These categories will be revisited in chapter 6.

EC1 **Advantage:** What value does the extended CT-first methodology provide compared to the conventional CT-first methodology?

EC2 **Relevance:** In which cases is the extended CT-first methodology favorable compared to the traditional CT-first methodology?

1.5. Scope and approach

The DJI F450 Flamewheel quadrotor UAV was made available by DAS. It is controlled by an open source control application called APM:Copter, which runs on a Pixhawk controller. This setup facilitates the development, replacement and testing of a single control component, without having to spend time on interfacing. The *attitude controller* component was selected, since it is independent of other control components which simplifies modeling, development and testing [Alexis&11].

The existing Proportional Integral Derivative (PID) attitude controller is modeled, simulated and tested. The modeling and simulation process includes all relevant aspects of the digital controller in the DE modeling domain and all relevant physical properties of the quadrotor in the CT modeling domain, resulting in a *co-model* and providing input for goal **G1**.
An attitude controller based on MPC is developed, modeled, simulated, implemented and tested to provide input for goals **G1**, **G2** and **G3**. The interface of the MPC controller is equal to that of the PID controller, which enables identical simulation and testing setups for the two controllers. The performance of the MPC and PID controllers is compared in both simulation and test environments, which provides input for goal **G1**.
In order to assess how well the co-model exhibits the behavior of the actual quadrotor (model fidelity), a comparison of the simulated and tested behavior of the individual controllers is also performed, with respect to goal **G1**. It is attempted to reach a level of fidelity that is as high as possible regarding quadrotor components related to attitude control.

It would be beneficial for the reader to have prior knowledge of model-based development, the CT-first modeling approach and basic PID control. However, the focus of this thesis is not on control engineering and neither is the purpose to determine which of the aforementioned control strategies is favorable. Control engineering merely acts as the domain from which a technical challenge is identified and the model-based methodology is applied.
This thesis is not concerned with comparing traditional development methods with conventional model-based development methods. The scope is limited to comparing the findings of this thesis to

a conventional CT-first modeling approach. Furthermore, tool related concerns, such as simulation stability, are beyond the scope of this thesis.

1.6. Reading guide

The conventions used throughout this thesis are presented in this section.

Terminology and emphasis:
Words or terms of special significance are written in *italic*, such as *emphasized*. A complete terminology list is given in appendix A.1.

References:
References for cited work appear in square brackets such as [Fitzgerald&14]. "Fitzgerald" refers to the surname of the main author, while "14" refers to the year of writing. When several authors have contributed to referenced work, an "&" is included between the surname and year of writing. When multiple publications from the author within the same year are cited, a letter is appended at the end in alphabetic order, e.g. [Fitzgerald&14a, Fitzgerald&14b].

Abbreviations:
Abbreviations appear in parentheses following the full description, such as Model Predictive Control (MPC). A complete list of abbreviations is given in appendix A.2.

Mathematical notation and symbols:
The mathematical notation is presented in detail in section 2.2. A complete list of symbols is given in appendix A.3.

Quotations: Quotes appear within double quotation marks and are written in *italic* such as:

"This is a quote"

– Author

Model elements:
Elements of models are written in a `plain typewriter font`, such as `Model element`. A function or operation is referred to by appending a closed parenthesis to it's name, such as `operation()`, regardless of any possible arguments.

VDM listings:
VDM model listings are presented as shown below.

```
public static min : set of real -> real
min(numbers) ==
  let min in set numbers be st
    forall number in set numbers &
      min <= number in
        min;
```

Listing 1.1: VDM function returning the minimum number from a set of numbers.

1.7. Structure

This thesis consists of seven chapters and three appendices as visualized in figure 1.2. The solid lines indicate the suggested order of reading and the dashed lines illustrate which chapter each appendix is related to. Appendix A assists the reader with complete lists of terminology, abbreviations and symbols and should be referred to as necessary. The mathematical derivations on linearization and discretization have been left out of chapter 2 and can be found in appendix B. Likewise, the mathematical regressions referred to in chapter 3 are presented in appendix C.

Chapter 2: Introduces the theory and technologies used to develop a collaborative model of a quadrotor UAV. A brief introduction to quadrotor dynamics is given. Additionally, the DESTECS modeling tools and formalisms are presented.

Chapter 3: Introduces the physical quadrotor components and how the extended CT-first methodology has been used to model the physical system dynamics.

Chapter 4: Presents the DE-model of the control application, including both PID and MPC attitude controller implementations, and how the DE-model is connected with the CT-model to form the collaborative model. Additionally, the realization and integration of the MPC attitude controller is described.

Chapter 5: Presents the results obtained through co-simulations and realization test measurements.

Chapter 6: Describes the applied methodology extension of the CT-first approach. The extension is clarified through an example of an Anti-lock Braking System (ABS) and evaluated based on advantages and relevance.

Chapter 7: Concludes the thesis by presenting the achieved results and envisaged future work.

Appendix A: Provides complete lists of terminology, abbreviations and symbols.

Appendix B: Presents the linearization and discretization of the attitude state space model of the quadrotor, used in the MPC attitude controller.

Appendix C: Presents the mathematical regressions of the physical parameters used to refine the CT-model.

A CD is attached to this thesis, on which the thesis is available electronically in pdf format. Furthermore, the artifacts listed below, that have been produced during the thesis work, are available. For more information on the complete list of artifacts, including the authors contribution to each artifact, see table 4.1.

- Co-model including DE and CT models
- Control application source code
- Matlab scripts used for MPC development and mathematical regressions used for model refinement
- SolidWorks 3D CAD model of the quadrotor

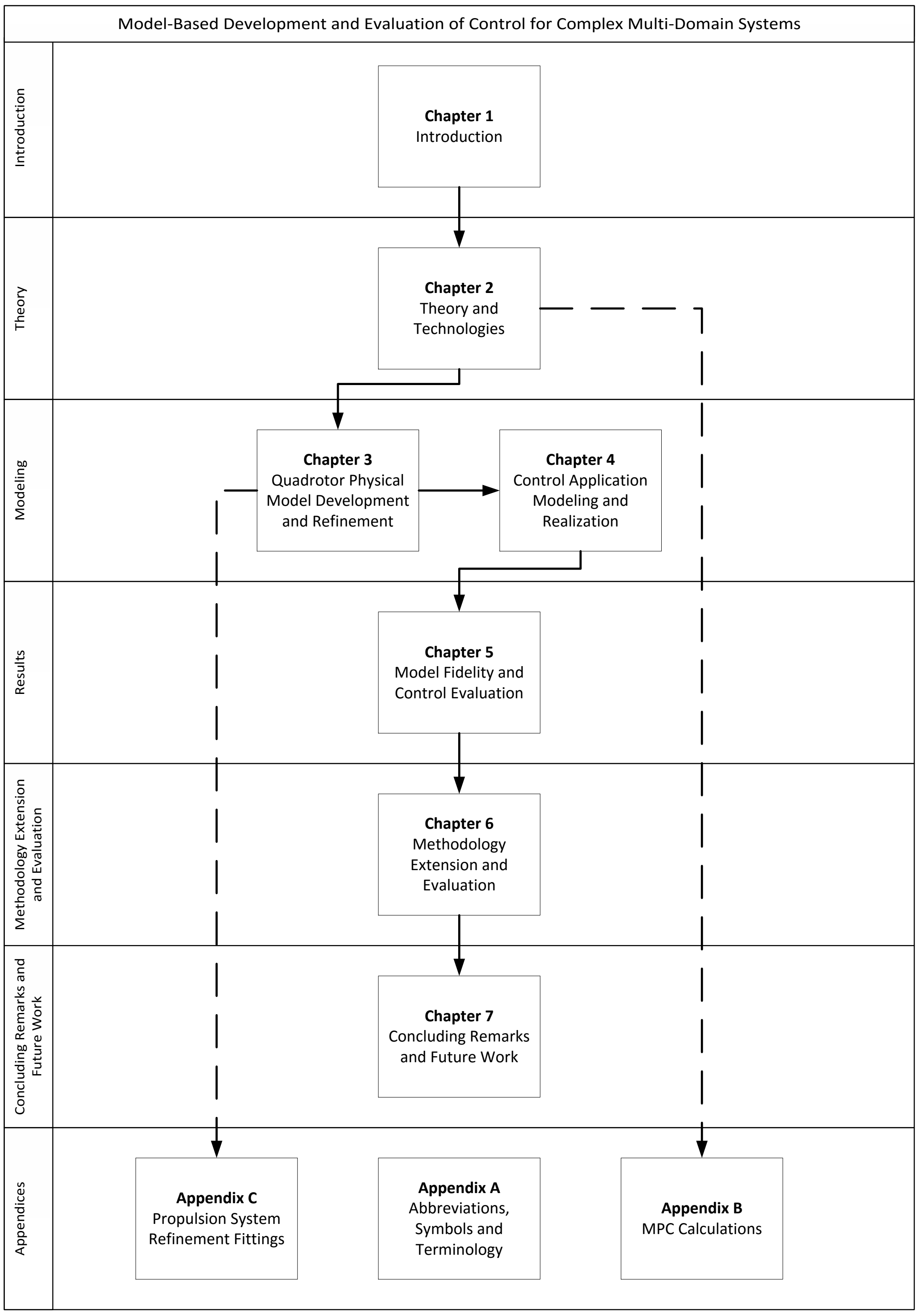

Model-Based Development and Evaluation of Control for Complex Multi-Domain Systems
Introduction
Theory
Modeling
Results
Methodology Extension and Evaluation
Concluding Remarks and Future Work
Appendices
Chapter 1 Introduction
Chapter 2 Theory and Technologies
Chapter 3 Quadrotor Physical Model Development and Refinement
Chapter 4 Control Application Modeling and Realization
Chapter 5 Model Fidelity and Control Evaluation
Chapter 6 Methodology Extension and Evaluation
Chapter 7 Concluding Remarks and Future Work
Appendix C Propulsion System Refinement Fittings
Appendix A Abbreviations, Symbols and Terminology
Appendix B MPC Calculations

Figure 1.2: Thesis structure

Chapter 2

Theory and Technologies

In chapter 1 the context of the thesis was set and the goals of developing a collaborative model and performing a methodology evaluation were presented. This chapter introduces the underlying theories and technologies necessary to understand the quadrotor modeling and MPC attitude controller realization, which are presented in subsequent chapters.

2.1. Introduction

A *quadrotor*, also referred to as a quadcopter or simply a drone, is a multirotor helicopter that is propelled by four rotors. By controlling the speed of each rotor individually, a quadrotor can change thrust direction and magnitude and thereby achieve horizontal and vertical movement. In spite of the four rotors, a quadrotor remains an under-actuated and dynamically unstable system [Bouabdallah&04, Czyba&12], for which a *controller* is necessary in order to achieve flight.
Conventional controllers steer a physical system by regulating the actuators of the system based on past system *state* information, which is usually obtained through a sensor.
Controllers based on Model Predictive Control (MPC) attempt to predict the future state of the system, based on a mathematical model of the system and state information, in order to achieve a more desirable control.
The capability of collaborative modeling and simulation of a control dependent Cyber-Physical System (CPS), such as a quadrotor, is made available by the DESTECS formalism and tools. The *Crescendo* tool provides simultaneous simulation of both the Discrete Event (DE) and Continuous Time (CT) modeling domains, which is required in order to simulate multi-domain systems.
This chapter starts out by covering mathematical notation style and conventions in section 2.2. This is followed by an introduction to the dynamics of a quadrotor including an explanation of the underlying physics in section 2.3. Afterward, the basic principles of MPC and the state space model behind it is presented in section 2.4. Finally, the used tools, formalism and platforms are presented in sections 2.5 through 2.7.

2.2. Mathematical notation style and conventions

When dealing with complex control systems, consistency in notation is a prerequisite for comprehending the matter. The following list describes the general mathematical notation style used throughout the thesis. The mathematical symbols used in the thesis will be described at first use, and a complete list of symbols including a short description can be found in appendix A.3.

Scalars and variables are denoted with a lowercase letter, including Greek letters, such as x or ϕ.

Vectors are boldfaced lowercase and the dimension is described alongside the vector definition. Unless stated otherwise, all vectors are column vectors, such as $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^2$.

A series of **discrete time samples** are considered as a vector and the samples are indexed with parentheses, e.g. in equations like: $\mathbf{x}(k+1) = \mathbf{x}(k) + c$.

Matrices are boldfaced uppercase and dimensions are described similar to vectors, such as $\mathbf{X} \in \mathbb{R}^{n \times m}$.

A diagonal matrix, with all non-diagonal elements equal to zero, is written in short as $\mathbf{diag}\{a, b, c\} = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix}$

The transpose of a vector or a matrix is denoted with a raised uppercase t, such as $\mathbf{x}^T$ or $\mathbf{X}^T$.

The time derivative of a variable is denoted with a dot, and the double time derivative with a double dot, such as $\dot{x} = \frac{dx}{dt}$ and $\ddot{x} = \frac{dx^2}{dt^2}$.

Coordinate systems are identified with a subscript lowercase blackboard-style letter. This applies for scalars, vectors and matrices, such as $x_{\mathbb{b}}$, $\mathbf{x}_{\mathbb{b}}$ or $\mathbf{X}_{\mathbb{b}}$ respectively.

2.3. Quadrotor dynamics

Quadrotor dynamics have been extensively studied and documented in literature, e.g. [Bouabdallah&04, Hoffmann&07, Mahony&12, Carrillo&13]. Here, a brief introduction into the dynamics of *cross configuration* quadrotors is given. The cross configuration refers to the rotors of the quadrotors, which are positioned in a cross, with the flying direction between the two front rotors as seen in figure 2.1.

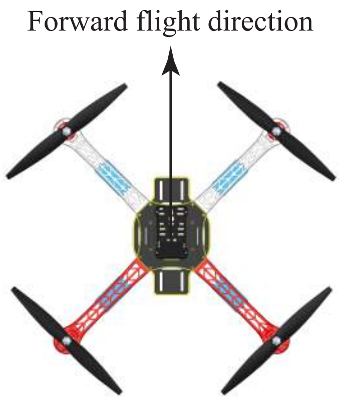


Figure 2.1: Cross configuration quadrotor

The dynamics presented here are based on an assumption of a rigid body and with no aerodynamic effects. The aerodynamic effects are neglected due to the fact that this thesis' primary focus is on near-hovering flight with minimal wind disturbances, where aerodynamic effects have little significance [Mahony&12].

2.3.1 Coordinate systems

The absolute position and orientation of a quadrotor is described with the use of two coordinate systems: A world-fixed ($\mathbb{w}$) and a body-fixed ($\mathbb{b}$) coordinate system. The axes of the world-fixed coordinate system are aligned with north, east and down. The body-fixed coordinate system is fixed to the centre of gravity of the quadrotor with $\mathbf{x}_{\mathbb{b}}$ in the forward direction, as seen in figure 2.2. Note that the coordinate systems will be referred to as the world reference frame and the body-fixed frame.

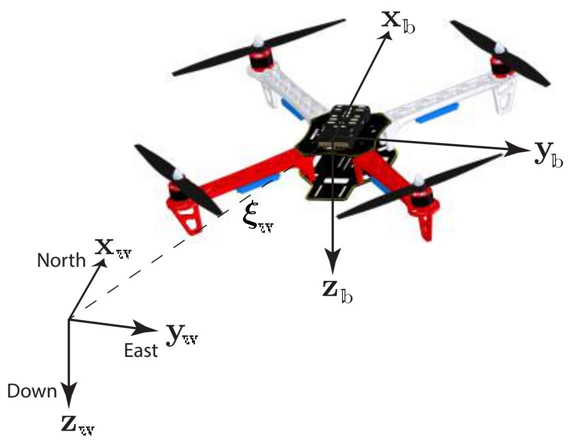

$\mathbf{x}_{\mathbb{b}}$
$\mathbf{y}_{\mathbb{b}}$
$\mathbf{z}_{\mathbb{b}}$
$\boldsymbol{\xi}_{\mathbb{w}}$
North
$\mathbf{x}_{\mathbb{w}}$
East
$\mathbf{y}_{\mathbb{w}}$
Down
$\mathbf{z}_{\mathbb{w}}$

Figure 2.2: Coordinate systems for a cross-configuration quadrotor with a North-East-Down world reference

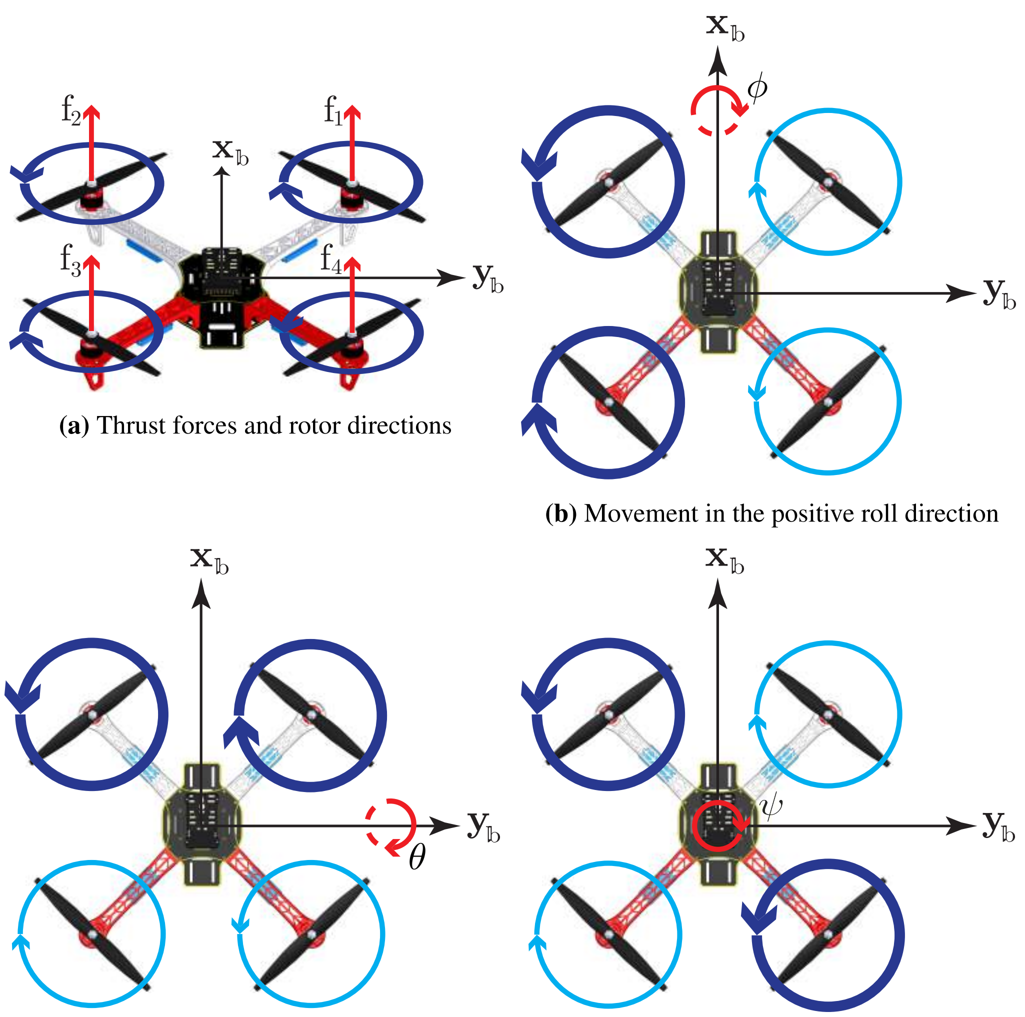
The position of the quadrotor is described with $\boldsymbol{\xi}_{\mathbb{w}} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3$, which is the position vector of the center of gravity of the quadrotor, relative to the world reference frame $\mathbb{w}$.

The orientation of the quadrotor is described using Tait-Bryan angles called yaw ($\psi_{\mathbb{b}}$), pitch ($\theta_{\mathbb{b}}$) and roll ($\phi_{\mathbb{b}}$), often referred to as Euler angles. An Z-Y-X rotation convention is used [Flores15].

2.3.2 Basic operation principles

The most intuitive way of understanding the motion of a quadrotor is by investigating how the rotation of the individual rotors affect the motion of the body. In figure 2.3, roll, pitch, yaw and thrust motions are visualized. The thickness and color of the arrows around the rotors indicates the speed and direction of the rotor, where a thick dark arrow indicates high speed and a thin light arrow indicates low speed. The red arrows indicate the motion of the quadrotor.

Figure 2.3a shows the quadrotor with equal speed on all rotors, resulting in equal forces from each rotor. This allows the quadrotor to move along the $\mathbf{z}_{\mathbb{b}}$ axis by increasing or decreasing the rotor speed. With a certain rotor speed the quadrotor will obtain a hovering state.


f_2
f_1
$\mathbf{x}_{\mathbb{b}}$
f_3
f_4
$\mathbf{y}_{\mathbb{b}}$
$\mathbf{x}_{\mathbb{b}}$
ϕ
$\mathbf{y}_{\mathbb{b}}$
$\mathbf{x}_{\mathbb{b}}$
$\mathbf{y}_{\mathbb{b}}$
θ
$\mathbf{x}_{\mathbb{b}}$
ψ
$\mathbf{y}_{\mathbb{b}}$

(a) Thrust forces and rotor directions

(b) Movement in the positive roll direction

(c) Movement in the positive pitch direction

(d) Movement in the positive yaw direction

Figure 2.3: Basic quadrotor movements

Figure 2.3b and 2.3c show how the roll $\phi_{\mathbb{b}}$ and pitch $\theta_{\mathbb{b}}$ angles are controlled. In both cases two adjacent rotors have increased speed compared to the other rotors, resulting in a movement around the $\mathbf{x}_{\mathbb{b}}$ and $\mathbf{y}_{\mathbb{b}}$ axes respectively.

Each rotor affects the quadrotor body with a torque in the opposite direction of the rotation of the rotor. These torques are canceled when two rotors rotate clockwise and two rotors rotate counterclockwise with the same speed. The torque from the rotors can be exploited to control the yaw angle $\psi_{\mathbb{b}}$, as illustrated in figure 2.3d, where two diagonal rotors rotate faster than the other rotors. This results is a movement around the $\mathbf{z}_{\mathbb{b}}$ axis, in this case in the positive yaw direction.

2.3.3 Equations of motion

The quadrotor dynamical equations are based on the Newton-Euler formalism, where the full non-linear dynamics can be expressed as:

$$m\ddot{\boldsymbol{\xi}}_{\mathbb{w}} = -mg\mathbf{z}_{\mathbb{w}} + \mathbf{R}\mathbf{f}_{\mathbb{b}} \tag{2.1}$$

$$\mathbf{I}\dot{\boldsymbol{\omega}}_{\mathbb{b}} = -\boldsymbol{\omega}_{\mathbb{b}} \times \mathbf{I}\boldsymbol{\omega}_{\mathbb{b}} + \boldsymbol{\tau}_{\mathbb{b}} \tag{2.2}$$

where m is the mass, $\boldsymbol{\xi}_{\mathbb{w}}$ is the position vector, g is gravity, $\mathbf{z}_{\mathbb{w}}$ is the down direction axis as seen in figure 2.2, $\mathbf{R}$ is the rotation matrix shown in equation 2.3 and $\mathbf{f}_{\mathbb{b}}$ is the total force applied to the quadrotor. $\mathbf{I}$ is the inertia matrix, $\boldsymbol{\omega}_{\mathbb{b}}$ is angular velocity and $\boldsymbol{\tau}_{\mathbb{b}}$ is angular moments of the quadrotor around the body-fixed frame [Carrillo&13].

$$\mathbf{R} = \begin{bmatrix} c\theta_{\mathbb{b}}c\psi_{\mathbb{b}} & s\phi_{\mathbb{b}}s\theta_{\mathbb{b}}c\psi_{\mathbb{b}} - c\phi_{\mathbb{b}}s\psi_{\mathbb{b}} & c\phi_{\mathbb{b}}s\theta_{\mathbb{b}}c\psi_{\mathbb{b}} + s\phi_{\mathbb{b}}s\psi_{\mathbb{b}} \\ c\theta_{\mathbb{b}}s\psi_{\mathbb{b}} & s\phi_{\mathbb{b}}s\theta_{\mathbb{b}}s\psi_{\mathbb{b}} + c\phi_{\mathbb{b}}c\psi_{\mathbb{b}} & c\phi_{\mathbb{b}}s\theta_{\mathbb{b}}s\psi_{\mathbb{b}} - s\phi_{\mathbb{b}}c\psi_{\mathbb{b}} \\ -s\theta_{\mathbb{b}} & s\phi_{\mathbb{b}}c\theta_{\mathbb{b}} & c\phi_{\mathbb{b}}c\theta_{\mathbb{b}} \end{bmatrix} \tag{2.3}$$

where $c\theta_{\mathbb{b}} = cos(\theta_{\mathbb{b}})$, $s\theta_{\mathbb{b}} = sin(\theta_{\mathbb{b}})$ and likewise for $\psi_{\mathbb{b}}$ and $\phi_{\mathbb{b}}$

2.4. Model predictive control

MPC was first presented in the 1970's [Richalet&78] and has developed considerably since then. One of the strengths of MPC is that it integrates optimal control, multivariable control, future references and constraints for non-linear processes [Camacho&07].

Compared to other controllers, such as Proportional Integral Derivative (PID) controllers, it requires far more complex mathematics and as a result, more computational power. This has for many years limited its application to slow systems[1], but the recent years technological development has enabled the use of MPC in fast systems as well [Camacho&07]. Industrial use of MPC in general has been documented thoroughly [Camacho&07, Boom&05, Maciejowski02], while MPC used for quadrotors remains an open research area and is not yet used in industry.

MPC for quadrotors has been addressed by researchers from both a simulation and a realization perspective [Raffo&08, Lopes&12, Alexis&11, Alexis&12]. Extensions to MPC exist, where Learning Based Model Predictive Control (LBMPC) is one of the most promising [Bouffard12, Aswani&11, Aswani&12]. LBMCP is however not addressed further as the scope of this thesis is limited to a general linear MPC controller.

The general idea about MPC is to incorporate state information, future references and constraints into the control, and predict how a system can be controlled in an optimum way. Standard PID control can be considered reactive, since it steers the system towards a desired state based on previous state information. Conversely, MPC is proactive, since it steers the system towards a future reference based on state information and a prediction of how the system can be controlled. In other words, MPC utilizes a mathematical model of the system, in order to obtain better control.

For an inherently unstable system, such as a quadrotor [Bouabdallah&04, Czyba&12], this model can be described either by transfer functions or a state space model. Transfer functions are widely used in industry, whereas state space models are often preferred in academia. In the particular case of a quadrotor, which is a Multiple-Input Multiple-Output (MIMO) system, a state space model is easier obtained and better suited for evaluation of stability and robustness [Camacho&07].

[1] A slow system is considered a system with a sample frequency of a few hertz

2.4.1 State space model

The state space model for a quadrotor can be decoupled into a position model and an attitude model [Bouabdallah07a]. This thesis only addresses attitude control. Therefore, only the attitude state space model is described and incorporated into the MPC controller. In equations 2.4 and 2.5 a general continuous-time state space model is presented. $\mathbf{x}_s$ is the state vector, $\mathbf{u}$ is the control input, $\mathbf{y}_s$ is the system output and $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are system dependent matrices.

$$\dot{\mathbf{x}}_s = \mathbf{A}\mathbf{x}_s + \mathbf{B}\mathbf{u} \tag{2.4}$$

$$\mathbf{y}_s = \mathbf{C}\mathbf{x}_s \tag{2.5}$$

For a quadrotor the attitude state space model can be written in compressed form as shown in equations 2.6 and 2.7. This model includes the forces exerted on the quadrotor by the rotors and the gyroscopic effect of the body, under the assumption that the moment of inertia of the quadrotor is a diagonal matrix: $\mathbf{I} = \mathbf{diag}\{I_{xx}, I_{yy}, I_{zz}\}$. This assumption is valid for the specific quadrotor in use, since the off-diagonal elements are negligible (see section 3.6). The state space model can be expressed as:

$$\dot{\mathbf{x}}_s = \begin{bmatrix} \dot{\phi}_{\mathbb{b}} \\ \ddot{\phi}_{\mathbb{b}} \\ \dot{\theta}_{\mathbb{b}} \\ \ddot{\theta}_{\mathbb{b}} \\ \dot{\psi}_{\mathbb{b}} \\ \ddot{\psi}_{\mathbb{b}} \end{bmatrix} = \begin{bmatrix} \dot{\phi}_{\mathbb{b}} \\ \dot{\theta}_{\mathbb{b}}\dot{\psi}_{\mathbb{b}}\frac{I_{yy}-I_{zz}}{I_{xx}} + \frac{d}{I_{xx}}\mathbf{u}_1 \\ \dot{\theta}_{\mathbb{b}} \\ \dot{\phi}_{\mathbb{b}}\dot{\psi}_{\mathbb{b}}\frac{I_{zz}-I_{xx}}{I_{yy}} + \frac{d}{I_{yy}}\mathbf{u}_2 \\ \dot{\psi}_{\mathbb{b}} \\ \dot{\theta}_{\mathbb{b}}\dot{\phi}_{\mathbb{b}}\frac{I_{xx}-I_{yy}}{I_{zz}} + \frac{1}{I_{zz}}\mathbf{u}_3 \end{bmatrix} \tag{2.6}$$

$$\mathbf{y}_s = \begin{bmatrix} \phi_{\mathbb{b}} \\ \theta_{\mathbb{b}} \\ \psi_{\mathbb{b}} \end{bmatrix} \tag{2.7}$$

where d is the quadrotor arm length and $\mathbf{u}$ is the control input.
The attitude state space model was linearized with a first order Taylor expansion around an equilibrium point, and subsequently discretized in order to be usable in the digital implementation. The equilibrium point was chosen as a hovering state with zero angular velocities and zero angular moments. The discretized state space model is shown in equation 2.8 and 2.9. A complete description of the linearization and discretization can be found in appendix B.

$$\mathbf{x}_d(k+1) = \mathbf{A}_d\mathbf{x}_d(k) + \mathbf{B}_d\delta\mathbf{u}(k) \tag{2.8}$$

$$\mathbf{y}_d(k) = \mathbf{C}_d\mathbf{x}_d(k) \tag{2.9}$$

where $\delta\mathbf{u}$ denotes the change in $\mathbf{u}$. The discretized state space model is used in the formulation of the MPC controller.

2.4.2 Model predictive control formulation

The basic working principles of a discretized MPC controller is illustrated in figure 2.4. At each sample time, a prediction of the future output is calculated based on past and present state information and a reference trajectory. With a perfect state space model and no external disturbances, the predicted output can be achieved by controlling the system with the predicted control input. To deal with model inaccuracies and disturbances, a *receding horizon* control strategy is used. With such a strategy, only the first predicted control input is applied, and the prediction is recalculated at every time step. The prediction length spans multiple sample times and is referred to as the *prediction horizon* (N). In addition to the prediction horizon another associated term, *control horizon* (M), is used in MPC. The control horizon describes how far into the prediction the control input can be changed. In figure 2.4, the control horizon and prediction horizon have equal length, which is often not the case.

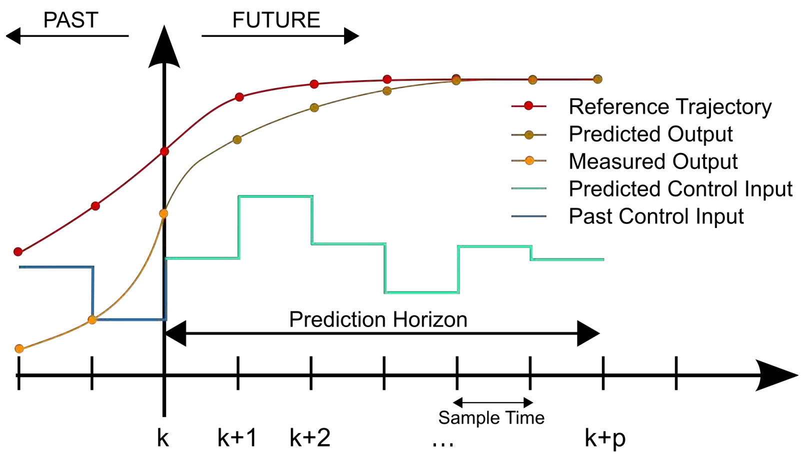


Figure 2.4: A basic working principle of Model Predictive Control, created by Martin Behrendt[2]

A discrete MPC controller can be split into three main blocks, as shown in figure 2.5: A *prediction model*, an *optimizer* and a *discrete integrator*.

The prediction model calculates the *predicted system output* ($\hat{\mathbf{y}}_d$), e.g. attitude angles roll, pitch and yaw. The prediction is based on the discrete state space model, state information ($\mathbf{x}_d$) from the physical system and a *predicted control input change* ($\delta\hat{\mathbf{u}}$).

The optimizer calculates the *optimum control input change* ($\delta\mathbf{u}^*$). The goal of the optimizer is to identify the control inputs that steer the system towards the *reference trajectory* ($\mathbf{r}$), in the most optimal way. The optimality is defined by an *objective function* and a number of *constraints*.

The discrete integrator integrates the optimum control input changes ($\delta\mathbf{u}^*$) to obtain the *optimum control input* ($\mathbf{u}^*$), which is applied to the physical system.

[2] Available at `https://commons.wikimedia.org/wiki/File:MPC_scheme_basic.svg`, visited 31-12-2015

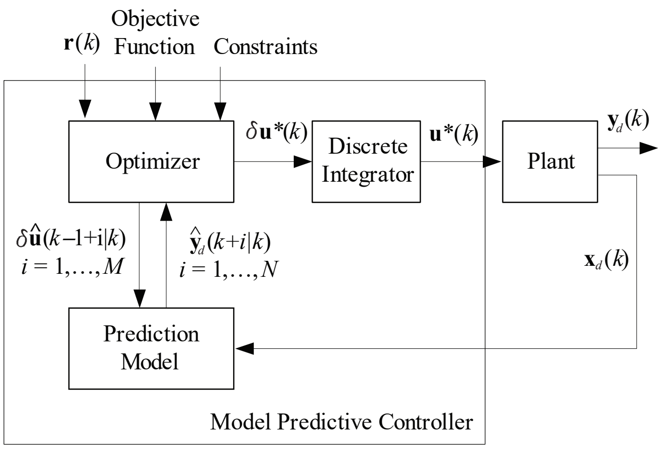


Figure 2.5: MPC overview, illustration is copied from [Lopes&12] and slightly modified.

The objective function used by the optimizer is adopted from [Lopes&12] and penalizes tracking errors as well as control variations (see equation 2.10). Tracking error is defined as the difference between the reference trajectory and the actual system output; for attitude control of a quadrotor the tracking error is the roll, pitch and yaw angular position errors. Control variations are a measure of how rapidly the control input changes. For the quadrotor this means that rapid and excessive changes of rotor speeds are avoided. However, tracking errors are often penalized far more than control variations. The objective function is:

$$J(\delta\hat{\mathbf{U}}) = \sum_{j=1}^{q}\sum_{i=1}^{N} \boldsymbol{\mu}_j[\hat{\mathbf{y}}_{dj}(k+i|k) - \mathbf{r}_j(k+i)]^2 + \sum_{j=1}^{p}\sum_{i=1}^{M} \boldsymbol{\rho}_j[\delta\hat{\mathbf{u}}_l(k-1+i|k)]^2 \tag{2.10}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\rho}$ are weights on the tracking errors and control variations respectively. $\delta\hat{\mathbf{U}}$[3] is a collection of the predicted control inputs for each sample in the control horizon, represented as a vector of a M stacked $\delta\hat{\mathbf{u}}$s.

MPC has the ability to incorporate constraints on both input and output variables. Constraints on the control inputs ($\mathbf{u}$) assure that the prediction model will be limited, in the same way as the physical system is limited. For a quadrotor this means, that the model should have the same limits as the physical propulsion system, taking maximum motor velocities, rotor lift characteristics etc. into account. It is common in literature to constrain the system output variables as well [Alexis&12, Lopes&12]. For a quadrotor this means, that the attitude angles roll, pitch and yaw can be limited, to ensure that the quadrotor never ends up with an undesirable attitude, e.g. turned upside down.

The objective function and the constraints can be transformed into a Quadratic Programming (QP) problem of the form shown in equations 2.11 and 2.12 [Lopes&12]. Such a QP problem with linear constraints is a well known optimization problem, that can be solved by numerical algorithms [Maciejowski02].

$$J(\delta\hat{\mathbf{U}}) = \frac{1}{2}\delta\hat{\mathbf{U}}^T Q \delta\hat{\mathbf{U}} + f^T\delta\hat{\mathbf{U}} + c \tag{2.11}$$

$$A\delta\hat{\mathbf{U}} \leq B \tag{2.12}$$

[3] Note that the mathematical notation convention is violated here, $\delta\hat{\mathbf{U}}$ is a vector, and not a matrix, but still written as uppercase.

At each sample instant the QP problem must be formulated, which involves a substantial amount of matrix calculations. The QP problem is subsequently solved with a numerical algorithm and the resulting control input is used to control the system. The performance of the MPC controller will be evaluated in the co-simulations of the co-model as well as with tests of the realization.

2.5. DESTECS formalism and tools

The DESTECS formalism supports co-modeling and co-simulation through the Crescendo tool [Fitzgerald&14]. The two modeling domains CT and DE are connected through a co-simulation engine in the Crescendo tool, as illustrated in figure 2.6.

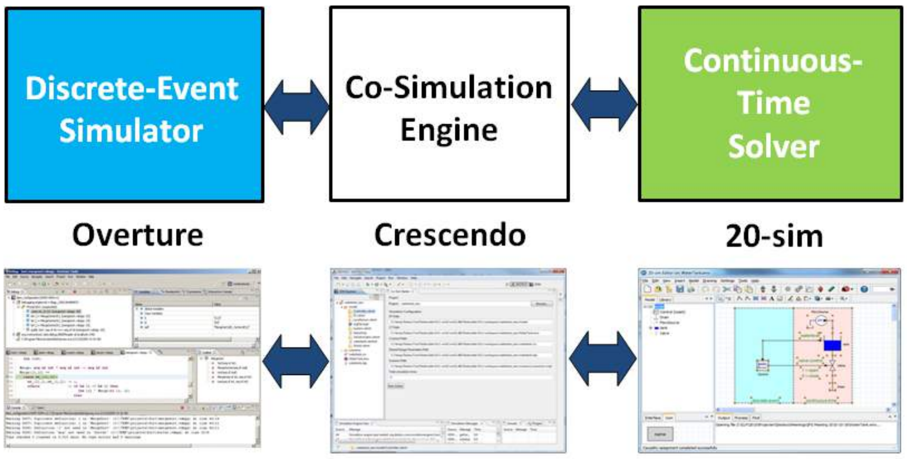


Figure 2.6: The DESTECS tool chain used for collaborative modeling and simulation

Overture is used to model discrete events, such as embedded control software, by the use of the VDM-RT formalism [Lausdahl&13a, Overture07]. Listing 2.1 shows a VDM implementation of a function capable of finding the minimum from a set of numbers as an example.

20-sim is used to model continuous time parts of a system, such as mechanics and electronics, by the use of differential equations, block diagrams and/or bond graphs [Broenink97].

An example of a differential equation implementation in 20-sim is presented in figure 2.7, implementing equation 2.2.

Figure 2.8 shows a snippet of a block diagram from 20-sim, including connections with causality, known from bond graphs.

```
public static min : set of real -> real
min(numbers) ==
  let min in set numbers be st
    forall number in set numbers &
      min <= number in
        min;
```

Listing 2.1: VDM example of a function returning the minimum number from a set of numbers.

```
// Angular equations: (Body-fixed coordinates)
acc_angular = inverse(I) *(-cross(vel_angular,(I*vel_angular)) + tau);
vel_angular = int(acc_angular, intinit);
pos_angular = int(vel_angular, intinit);
```

Figure 2.7: 20-sim example of equation 2.2 and the angular velocity and position

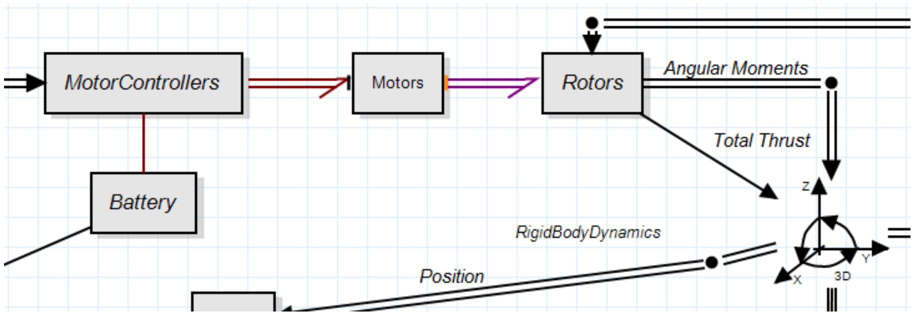


Figure 2.8: 20-sim block diagram example. Black connections have indifferent causality. Red/purple connections show causality information

The three main modeling methodologies proposed by [Fitzgerald&14] in connection with the DESTECS project are described as follows:

Table 2.1: Modeling methodologies proposed by the DESTECS project.

Methodology	**Description**
CT-First	Initial work is performed on the CT model, without a co-model, to gain a sufficient understanding of the CT side before commencing work on the co-model. Often used when the biggest challenges or uncertainties lie on the CT side.
DE-First	Similar to CT-first, but where initial work is performed on the DE model.
Contract-First	The first step is to agree on the contract specifying the interface between the DE and CT side. Often used when multiple teams must collaborate on the development of a co-model.

2.6. Other technologies

The DESTECS technology is used for co-modeling and co-simulation. A number of other tools have been used for various other purposes, as described briefly in table 2.3.

Table 2.3: Brief description of tools used beside the DESTECS technology.

Tool	**Use**
Matlab	Numerical computations and algorithm development regarding the MPC formulation. Data processing and visualization of both simulation results and test results.
Solidworks	Calculation of mass properties of the physical system, eg. moments of inertia.
Visual Studio	Development of a DLL, used as part of the DE model (see section 4.3.2). PC-based testing of the numerical solver used in MPC.
Eclipse	Development and programming of the MPC attitude controller realization.

The commercial tool Simulink[4] has been studied as an alternative to the Crescendo tool, since it has a lot of add-on packages including event-based modeling, physical modeling, control systems, code generation and real-time simulation. Other similar tools exist, such as Modelisar[5], CosiMate[6], Topcased[7] and Ptolemy[8]. Crescendo was chosen due to the availability of tool expertise and support at Aarhus University, Department of Engineering.

2.7. Realization platform

The realization of the control application is based on the DroneCode Software Platform, which is an open source platform for Unmanned Aerial Vehicles (UAVs).[9] The embedded software controller, called an autopilot, is based on the APM:Copter application, which is part of the APM Flight Code.[10] The autopilot is deployed to the open hardware platform Pixhawk, which runs a real-time operating system called NuttX [Meier&12].

Figure 2.9: Pixhawk autopilot[11]

[4] `http://se.mathworks.com/products/simulink/`, visited 31-12-2015
[5] `https://itea3.org/project/modelisar.html`, visited 01-01-2016
[6] `http://site.cosimate.com/`, visited 01-01-2016
[7] `https://www.polarsys.org/topcased`, visited 01-01-2016
[8] `http://ptolemy.eecs.berkeley.edu/`, visited 01-01-2016
[9] `https://www.dronecode.org/dronecode-software-platform`, visited 31-12-2015
[10] `https://www.dronecode.org/software/flight-code`, visited 31-12-2015
[11] `https://store.3drobotics.com/products/3dr-pixhawk`, visited 31-12-2015

Chapter 3

Quadrotor Physical Model Development and Refinement

Chapter 2 introduced the theoretical foundation for quadrotor dynamics, and the technologies which support the development of the Model Predictive Control (MPC) based attitude controller. This chapter focuses on the modeling effort related to the physical quadrotor, showing the development approach with the three continuous time models created during the project. The control application model, which is part of the second and third co-model, is presented in chapter 4.

3.1. Introduction

A Continuous Time (CT) model captures physical system dynamics, involving both electrical and mechanical domains. The dynamics are modeled with differential equations. This process is often assisted by a modeling tool, which provides a more abstract representation of the differential equations, and thereby simplifies the modeling.
This chapter presents the development of a CT model of the DJI F450 Flamewheel quadrotor, made with the 20-sim modeling software. A step-wise physical modeling approach is presented, demonstrating how a CT-first development strategy can be applied, when the physical system is given in advance. In conjunction with this approach, the concept of model refinement is introduced. This demonstrates how a model can be improved by performing measurements on the physical device, the details of which can be found in appendix C.
Firstly, the system components of the quadrotor are described in section 3.2. Afterward, section 3.3 outlines the methodological modeling approach. Next, the working principle and modifications of the test environment are described in section 3.4, before section 3.5 summarizes the propulsion system refinement experiments performed using the testbed. Finally, a moment of inertia analysis of the quadrotor is covered in section 3.6.

3.2. System components

The decomposition of a complex system breaks the system into smaller parts or components, which are easier conceived and understood. This section provides a brief overview of the components of a quadrotor, as a result of a system decomposition. Figure 3.1 shows the main components of the DJI F450 Flamewheel quadrotor, while a description of all identified components are presented in table 3.1.

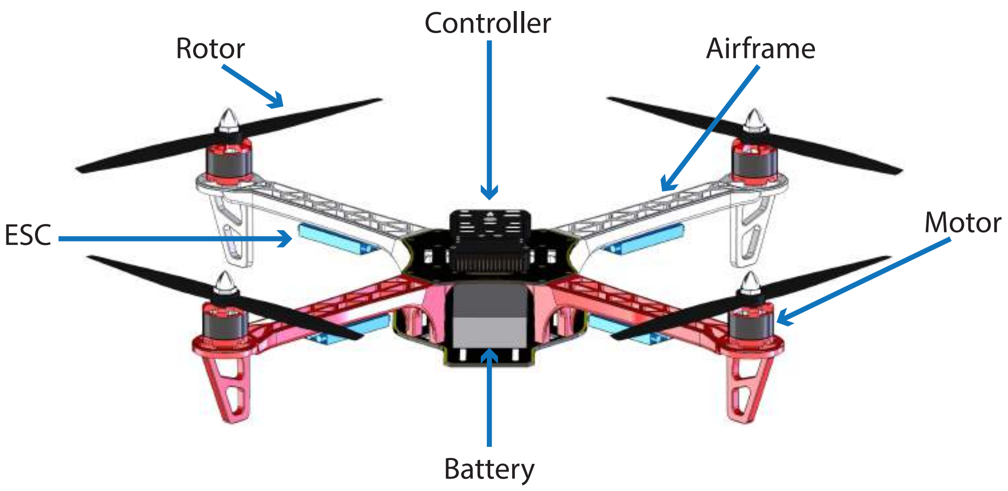


Figure 3.1: 3D model of the DJI F450 Flamewheel quadrotor in SolidWorks

Table 3.1: System components description

Name	**Description**
Airframe	The airframe is the main body of the quadrotor, including four arms. It houses the controller, telemetry, battery and the payload. Each arm carries a motor, a rotor and an Electronic Speed Controller (ESC).
Motor	The motor spins the rotor.
Rotor	When rotating, the rotor generates a lift force providing the aircraft with propulsion.
ESC	Motor drive, which converts the digital speed signal from the controller to three Pulse Width Modulation (PWM) signals to the motor.
Controller	Controls the speed of the rotors based on user input and sensor data. The controller contains internal accelerometers, gyroscopes, magnetometer (compass) and a barometer.
Battery	Power source.
Telemetry	Wireless communication between the quadrotor and the hand-held remote control. (Not shown in figure 3.1)
Payload	Optional: Camera, first response emergency equipment, sensors etc. (Not shown in figure 3.1)

3.3. Methodological modeling approach

A model-based development strategy is adopted for this project, which means that *model* are used to describe the design during development. A model is an abstract representation of a putative system. The model is abstract in the sense that details not relevant to the *purpose* of the model are omitted [Fitzgerald&14]. The modeling goal in this project is to aid the development and verification of an attitude controller for a quadrotor Unmanned Aerial Vehicle (UAV).
An extended CT-first modeling approach was used for this project since an in-depth understanding of quadrotor dynamics is a prerequisite for developing a controller. This approach is explained below.

3.3.1 Methodology

By taking advantage of the concept of abstraction, three models have been structured to support design throughout the entire development. The modeling effort starts at a high level of abstraction representing the modelers initial knowledge of the system. Gradually, more details are added to the components related to the modeling goal, resulting in a model with sufficient clarity and precision to confirm or refute the presence of desirable characteristics [Fitzgerald&14]. This principle was also described by a great thinker:

> *"Everything should be made as simple as possible, but no simpler."*
>
> – Albert Einstein

Each of the models below has a specific purpose, reflected in the components included and their level of detail.

3.3.2 Conceptual model: The flying box

At a very high level of abstraction, a quadrotor can simply be described as a flying object, or as in this case; a flying box. The box is propelled by lift forces applied at the corners enabling it to move vertically. Lateral movement is achieved by tilting the box in such a way that the X-Y plane faces towards the desired direction of movement. The flying box can be seen in a simulation environment in figure 3.2.

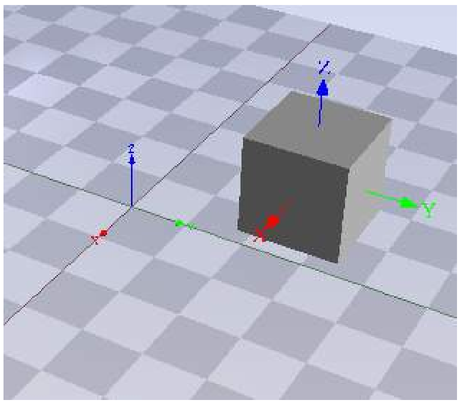


Figure 3.2: Flying box representation of a quadrotor

The purpose of the first model was to understand the movements of a quadrotor, which are mathematically described by equations 2.1 and 2.2 presented in section 2.3.3. The conceptual model also enables an analysis of how roll, pitch and yaw motion can be achieved by manipulating the size of the lift forces relative to each other, as described in section 2.3.2.

3.3.3 Generic component co-model: Fundamental quadrotor

The purpose of the generic component model was to introduce the problem of controlling a quadrotor from an embedded computer, and to investigate which components make up a quadrotor and how they influence the control. Figure 3.3 shows a Systems Modeling Language (SysML) Block Definition Diagram (BDD) of a quadrotor. The BDD shows the affiliation and quantity of the quadrotor components, also introduced in section 3.2.

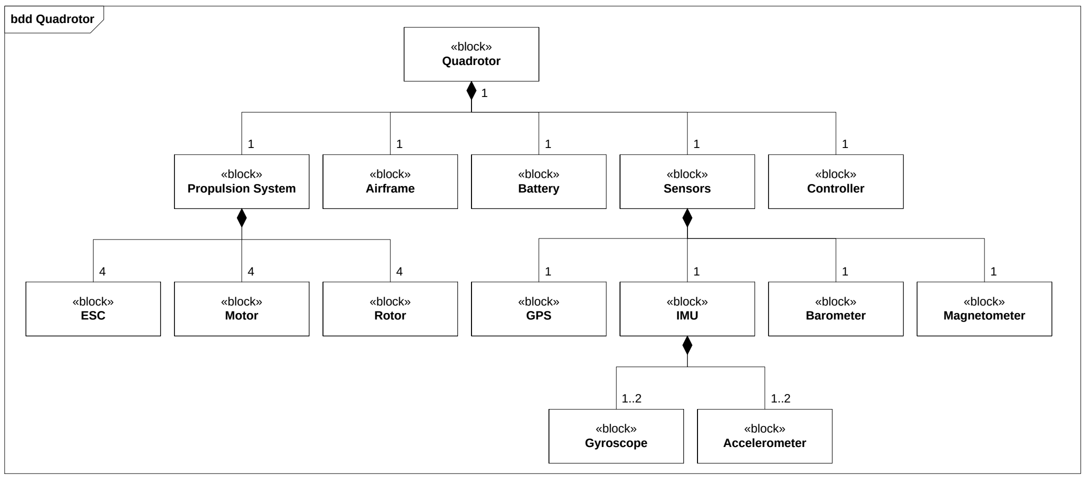


Figure 3.3: BDD of the quadrotor showing all components relevant for the modeling goal

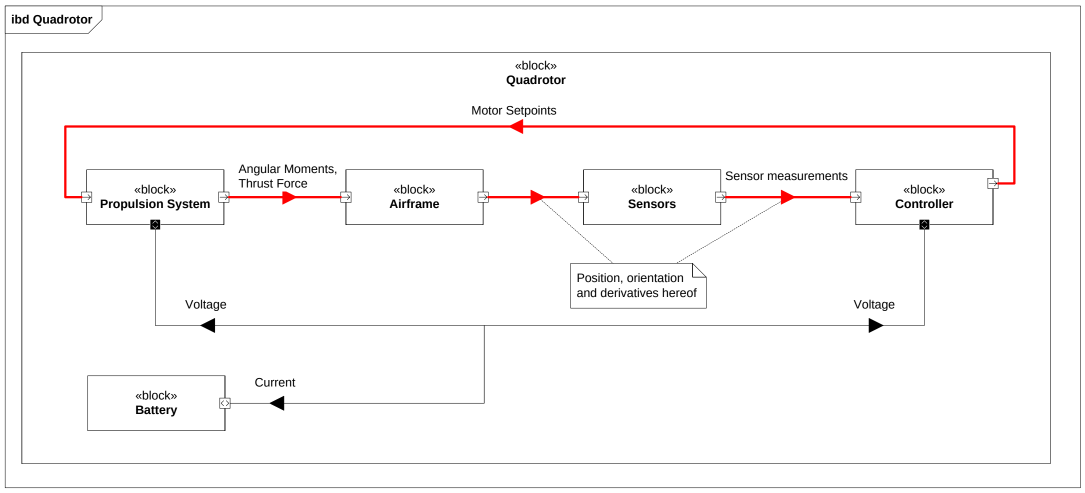


Figure 3.4: IBD of the quadrotor with emphasis on the control loop

Figure 3.4 shows an Internal Block definition Diagram (IBD) of the quadrotor including the dependencies and signal flow between the components. The red signal flow loop shows which components affect the control of the quadrotor. These components[1] have therefore been modeled at a higher level of detail in this model.

The `controller` is modeled as an interface to the control application model, which is simulated using Overture. The control application modeling is presented in more detail in chapter 4.

3.3.4 Specific device co-model: DJI F450 Flamewheel

The purpose of the specific device model is to compare different attitude control strategies of the DJI F450 Flamewheel quadrotor. This demands that the specific device co-model represents the attitude control behavior of the Flamewheel as accurate as possible. The Flamewheel was modified for mounting in the testbed (explained further in section 3.4), in order to be able to perform measurement and tests in a safe environment.

A two-step model refinement approach was used to achieve a high accuracy between the model and the actual quadrotor. First, all readily available component parameters were updated in the model, originating from data sheets and supplier specifications. Second, the remaining control-critical parameters were tested on the actual quadrotor using the testbed and then included in the model. This process is described in sections 3.5 and 3.6.

By bringing the control-critical components of the model as close to reality as possible, only the consequences of the assumptions, made regarding the quadrotor and testbed, are left as differences between the co-model and the actual quadrotor when considering attitude control. Assumptions regarding the quadrotor include a rigid airframe, rotors and motor axles. Additionally, aerodynamic effects, such as ground effect and blade flapping, are not considered. Relevant assumptions regarding the testbed are presented in section 3.4.

3.4. Test environment

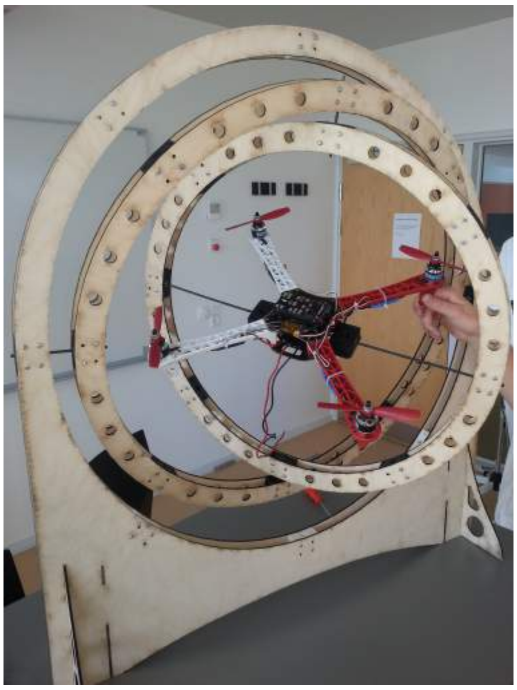
Tests have to be performed in order to verify the performance of the physical quadrotor and the fidelity of the co-model.
A testbed has been designed, which allows analysis of the angular motion of the quadrotor. The purpose of the testbed is to enable reproducible testing in a controlled environment.

3.4.1 Working principle

The testbed is based on the principle of a gimbal, also referred to as a Cardan suspension, which is a support structure that allows the rotation around a single axis [Needham65, Moon07]. A set of 3 gimbals combined, mounted inside each other with orthogonal pivot axes, provide the object mounted in the innermost gimbal with three degrees of freedom: roll, pitch and yaw. The idea originated from the authors, with the design and manufacturing being performed by Danish Aviation Systems (DAS).

Figure 3.5a shows the testbed with the Flamewheel mounted inside. The purpose of the testbed is to evaluate the attitude control of the quadrotor. Hence, translational movement is restricted, allowing measurements to focus only on the rotation of the aircraft.

[1] With the exception of the sensor components due to time constraints


(**a**) Testbed with quadrotor

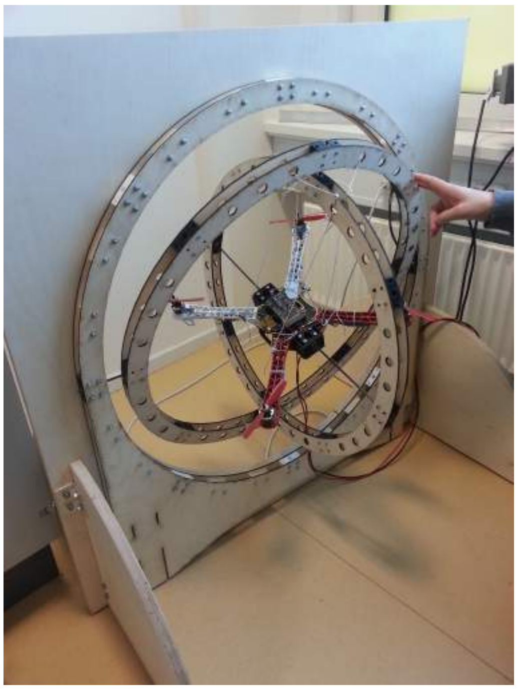

(**b**) Modified testbed with quadrotor

Figure 3.5: Test environment

3.4.2 Modifications

The Flamewheel was fitted with a bearing-mounted carbon-fiber rod going through the center along the first body axis ($\mathbf{x}_{\mathbb{b}}$, roll). The rod provides rotation around the first body axis and enables the quadrotor to be mounted in the testbed, as seen in figure 3.5a.
The testbed itself was modified to obtain a test environment, which enables reproducible tests to be performed, as seen in figure 3.5b. Since only the angular motion of the quadrotor is of interest, any translational motion of the quadrotor should be restricted. Translational motion originates from deflections in the structure of the testbed due to translational forces caused by the quadrotor. By attaching an 18 mm sheet of plywood to the testbed, the frame is reinforced. Furthermore, larger supports are added at the base of the testbed structure to increase stability.

Rotation around the second body axis ($\mathbf{y}_{\mathbb{b}}$, pitch) and the third body axis ($\mathbf{z}_{\mathbb{b}}$, yaw) requires the quadrotor to rotate the two testbed rings, which increases the moment of inertia and thereby impedes angular acceleration (as shown in equation 2.2). This is further described in section 3.6.
The increased moment of inertia around the second body axis depends on the instantaneous angular position of the inner ring - with maximum moment of inertia, when it is orthogonal to the outer ring, and minimum when it is parallel to it. This added complexity would increases the modeling effort of the test environment substantially. The entire mechanical structure of the testbed would have to be modeled as well, rather than simply including the moment of inertia of the testbed to that of the quadrotor. In order to obtain an appropriate level of model fidelity[2], within the time frame of the project, the inner ring has been fixated in an angular position orthogonal to the outer ring - at the cost of yaw motion. Even though the orthogonal position has the highest impact on the moment of inertia, it has been chosen to avoid the gimbal lock problem[3].

[2] See section 6.4 for a discussion of appropriate model fidelity.
[3] When the rotational axis of two testbed rings align, a degree of freedom is lost.

These modifications provide an increased credibility in the results from comparing test measurements with model simulations.

The co-model was modified to reflect the restrictions applied to the quadrotor movements as a result of being mounted in the testbed. Since the translational movement is restricted, translational acceleration in the CT model is set to zero for all three axes. Additionally, yaw motion is restricted. Assumptions regarding the modeling of the testbed includes:

1. The position of the testbed itself is fixed
2. There are no significant deflections in the testbed structure
3. The quadrotor center of gravity and hinge point coincide
4. The friction in the bearings of the testbed is equal to zero

3.5. Propulsion system refinement

The propulsion system of a quadrotor consists of the ESCs, the motors and the rotors. Figure 3.6 shows a detailed model of the propulsion system for a single arm, including non-ideal parasitic components in the 20-sim environment.

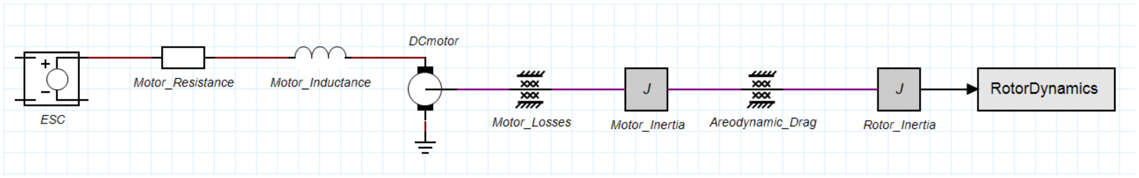


Figure 3.6: Propulsion system 20-sim model

An ESC is an embedded controller tasked with controlling a motor. The performance of the ESCs depend on both hardware and software implementation, and is not readily available in data sheets. In order to model the ESCs, as accurate as possible, a transfer function is derived based on test measurements, as described in appendix C.

Based on the assumptions taken in [Cai&10], a permanent magnet Brushless Direct Current (BLDC) motor can be modeled as a commutative DC motor with a series resistance and inductance. Furthermore, the motor loss, including friction and windage loss [Toliyat&04], is modeled as a bearing providing rotational resistance, whilst the inertia of the rotating part of the motor is simply modeled as an inertia.

The rotor is modeled as a bearing, an inertia and a dynamics block, inspired by [Czyba&12]. The bearing friction accounts for the aerodynamic drag of the rotor. The rotation is input to the $RotorDynamics$ block, where lift force and reaction torque are calculated. These quantities are used to calculate the resulting angular moments on the airframe of the quadrotor, which are used in the equations of motion introduced in section 2.3.3.

The approach for refining each parameter is outlined in table 3.2, the values are given in table 3.3, whilst the calculations and mathematical regressions can be found in appendix C.

Table 3.2: Propulsion system refinement approach

Parameter	**Refinement method**
ESC transfer function	Since the hardware and software implementation were unknown, and to save time, the ESC performance was measured with a *black box* approach [Nidhra&12]. The objective was to determine the relationship between the motor speed input and the output voltage[4]. Measurements at different input speeds were performed using the setup of figure 3.7a, leading to an exponential transfer function, see appendix C.
Motor resistance	Measured between two terminals.
Motor inductance	Calculated from experiment with known AC voltage, current and frequency.
Motor transfer function	Even though the voltage to speed relationship of an ideal DC motor can be considered linear [Cai&10], the measurements and [Tempo&11] suggest that an affine[5] relationship is more accurate. Measurements at different speeds were performed using the setup of figure 3.7a, leading to an affine transfer function, see appendix C.
Motor inertia	Determined with SolidWorks using the mass of the rotating part of the motor.
Motor loss	Motor loss is considered equal to the torque required to turn the motor with no load [Toliyat&04] and is assumed linearly proportional to the motor speed [Virgala&13]. The motor speed was determined by measuring the frequency of the cross-phase sinusoidal signal. Given the proportionality between motor torque and current, measurements of the current at different speeds provide the torque load on the motor. This along with the frequency was plotted and fitted using Matlab to obtain the motor loss coefficient, see appendix C.
Rotor lift	The lift force is assumed to be proportional to the square of the rotor speed [Mahony&12, Alexis&11], which was determined by measuring the frequency of the cross-phase sinusoidal signal on the motor. An experiment to measure the lift force at different speeds was performed using the setup shown in figure 3.7a.

[4] The actual output is a PWM signal, however since the motor is modeled as a commutative DC motor, an equivalent DC voltage calculated and used in the model.

[5] An affine function is the sum of a linear function and a constant

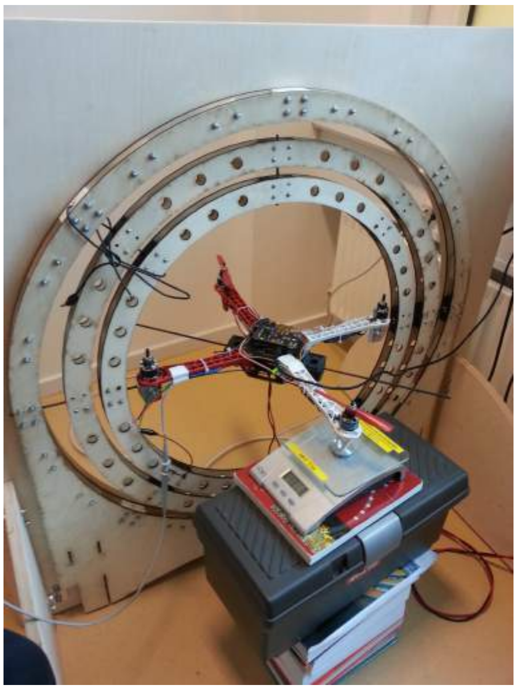
Table 3.2: Propulsion system refinement approach

Parameter	**Refinement method**
Rotor reaction torque	The reaction torque is assumed to be proportional to the square of the rotor speed [Mahony&12, Alexis&11], which is determined as in the rotor lift case. An experiment to measure the reaction torque at different speeds was performed using the setup shown in figure 3.7b.
Rotor aerodynamic drag	The aerodynamic drag is assumed to be proportional to the square of the rotor speed [Czyba&12]. The drag is measured as the load difference on the motor with and without the rotor attached, at different speeds, using the setup in figure 3.7a.
Rotor inertia	The rotor inertia was determined using SolidWorks. The mass and bulk dimensions[6] of the rotor were taken into account.

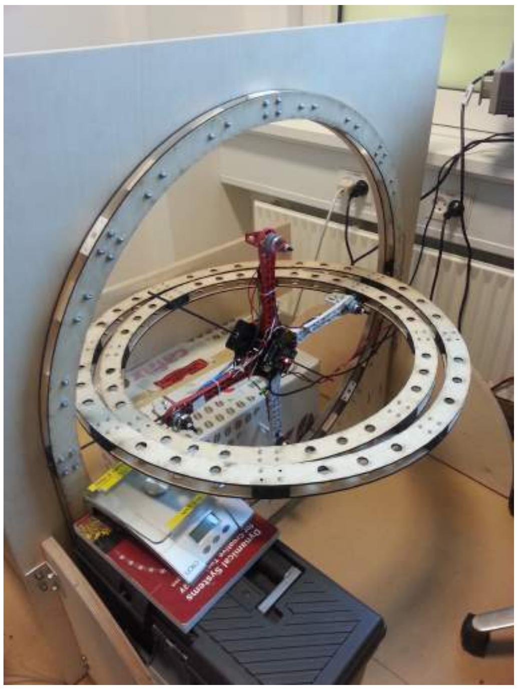
Table 3.3: Propulsion system parameters

Parameter	**Description**	**Value**	**Unit**
R_m	Motor resistance	0.127	Ω
L_m	Motor inductance	1.6968E-4	H
I_m	Motor rotor inertia	3.38E-7	kg m^2
M_{lo}	Motor loss coefficient	4.4528E-6	N m s
R_l	Rotor lift coefficient	1.5064E-6	N s^2
R_t	Rotor reaction torque coefficient	5.6977E-8	N m s^2
R_d	Rotor aerodynamic drag coefficient	2.0066E-8	N m s^2
I_r	Rotor inertia	2.800E-6	kg m^2

[6] The rotor blade length and hub diameter. The thickness and curving of the rotor blade have not been considered



(a) Lift coefficient test setup. The rotor in the background is spun at different speeds, which exerts an upward force on the arm on which it is mounted. Since the suspension axis of the quadrotor acts like a lever, the opposing arm exerts an equal but downward force on the scale.



(b) Reaction torque coefficient test setup. The two inner testbed rings of the testbed are positioned vertically with the larger one fixed to the wooden plate in the background. The innermost ring can rotate, such that when the uppermost rotor spins clockwise, the reaction torque will cause the innermost ring to rotate in the opposite direction and push down on the scale.

Figure 3.7: The quadrotor is suspended on an axis through the airframe using custom manufactured brackets (see figure 3.8). By utilizing the testbed, an airflow that takes into account the airframe arm, but is otherwise undisturbed, is achieved in both test setups.

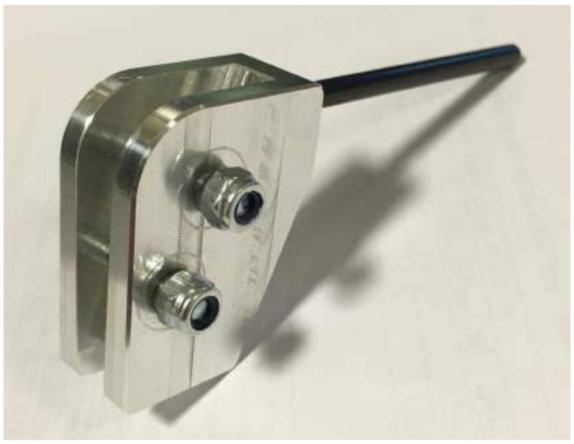


Figure 3.8: Mounting bracket

3.6. Moment of inertia

The total mass and mass distribution of the quadrotor are measured in order to determine the moment of inertia, which affects the angular acceleration of the quadrotor described in equation 2.2. SolidWorks was used to determine the moment of inertia based on a 3D model[7] (see figure 3.9) by setting the appropriate mass of each component of the quadrotor.

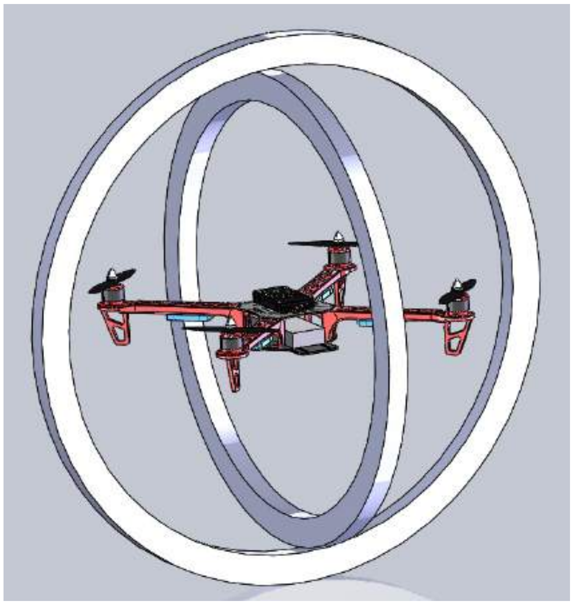


Figure 3.9: SolidWorks assembly of the quadrotor and two testbed rings.

Table 3.4 shows the resulting inertia tensor of the Flamewheel quadrotor. The quantities I_{xx},I_{yy} and I_{zz} are the moments of inertia with respect to the $\mathbf{x}_{\mathbb{b}}$, $\mathbf{y}_{\mathbb{b}}$ and $\mathbf{z}_{\mathbb{b}}$ axis respectively.
The inertia of the innermost testbed ring has been included in the calculation of I_{zz}. The inertia of both testbed rings have been included in the calculation of I_{yy}. Note that the inertia around the $\mathbf{y}_{\mathbb{b}}$ and $\mathbf{z}_{\mathbb{b}}$ axes is a factor of 3 and 2 larger than that of the $\mathbf{x}_{\mathbb{b}}$ axis respectively, which is very uncharacteristic for quadrotors. Therefore, the rotation around the $\mathbf{y}_{\mathbb{b}}$ and $\mathbf{z}_{\mathbb{b}}$ axes will be slower compared to a conventional quadrotor as a consequence of being mounted in the testbed.

Table 3.4: Moment of inertia tensor

Moment of inertia $[\boldsymbol{kg * m^2}]$		
$I_{xx} = 0.011\,511\,8$	$I_{xy} = 5.677\,48$E-6	$I_{xz} = 2.401\,52$E-6
$I_{yx} = 5.676\,46$E-6	$I_{yy} = 0.036\,565\,1$	$I_{yz} = 4.759$E-8
$I_{zx} = 2.429\,51$E-6	$I_{zy} = 4.737$E-8	$I_{zz} = 0.029\,074\,2$

[7] The 3D model was provided by DAS and is included on the attached CD

Chapter 4

Control Application Modeling and Realization

The development of the Continuous Time (CT)-model of the physical quadrotor was covered in chapter 3. This chapter presents the Discrete Event (DE)-model of the control application and how it is connected with the CT-model to form the co-model. Furthermore, realization details are presented, which are necessary to understand the co-simulation and test results presented in chapter 5.

4.1. Introduction

Collaborative modeling exploits two different modeling domains to capture the behavior of systems comprised of both physical elements and a digital controller. The physical system dynamics are modeled in the CT modeling domain to produce a CT-model, as described in chapter 3. The control application model of the digital controller is modeled in the DE modeling domain.
The combination of the CT and DE models is referred to as a collaborative model or *co-model*. Simultaneous simulation of both models is performed, while allowing information to be shared between them; termed *co-simulation*. The information shared between the models is defined in a *contract*, which usually contains sensor readings being sent to the control application model and actuator setpoints being sent to the physical system model.
The control application model in this thesis is a DE model of APM:Copter, an open source quadrotor control application, as described in section 2.7. The initial DE model was based on the control application with no alterations. Subsequently, a Model Predictive Control (MPC) based attitude controller was developed using Matlab, modeled using VDM, and inserted into the DE model as an alternative to the original Proportional Integral Derivative (PID) based attitude controller.
First, section 4.2 presents an overview of the co-model. Afterwards, a description of the control application model, including the two alternative attitude controllers based on MPC and PID control, is provided in section 4.3. Afterwards, section 4.4 provides a discussion of realization details and finally, section 4.5 presents an overview of all relevant artifacts produced as part of this thesis.

4.2. Collaborative model overview

The co-model is presented as an abstract representation of *blocks*, using a Block Definition Diagram (BDD) from the Systems Modeling Language (SysML)[1]. The BDD of figure 4.1 shows which blocks comprise the quadrotor co-model:

Physical system model A CT model of the physical quadrotor, developed through three iterations at different abstraction levels, as described in section 3.3.

Contract A contract which defines shared design parameters and variables exchanged between the models.

Control application model A DE model of the control application, which contains two different implementations of the attitude control block.

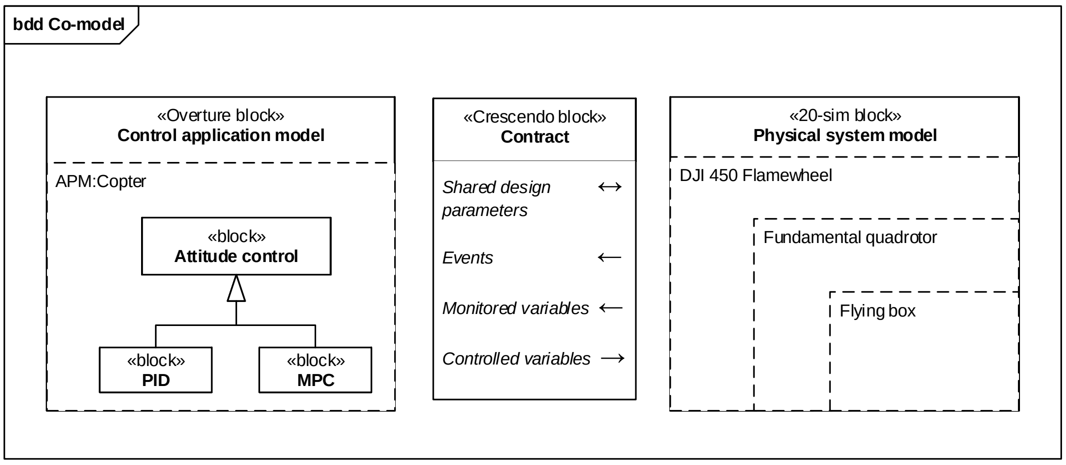


Figure 4.1: BDD of the quadrotor co-model, inspired by figure 2.1 from [Fitzgerald&14]

The contract specifies shared design parameters, events and monitored and controlled variables. These define the nature of the communication between the CT and DE models (arrows specify direction) [Fitzgerald&14].
Since both the DE and CT models use gravitational acceleration as a parameter, it has been defined as a shared design parameter. There are no events in the current version of the co-model. However, safety and fault handling aspects would likely introduce events as part of the future work, see section 7.3. The monitored variables, which can also be considered as the *state* of the quadrotor, are the angular position and velocity of the quadrotor. The controlled variables are the motor setpoints.

4.3. Control application model

The modeling effort has been structured such that the control related components of the application have been modeled at a high level of detail[2] whilst the remaining components have been modeled

[1] `http://www.omgsysml.org/`, visited 02-01-2016
[2] With the exception of the Attitude Heading Reference Systems (AHRS) block, due to time constraints

at a lower level of detail. Figure 4.2 shows an Internal Block definition Diagram (IBD) of the control related components of the model.

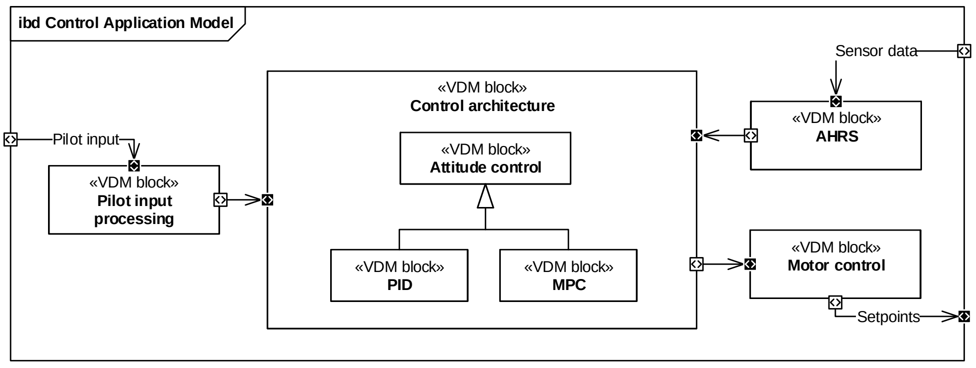


Figure 4.2: IBD of the control application model

The main loop of the control application model, shown in listing 4.1, elaborates on the tasks performed by the individual blocks. The periodic operation `fastLoop()`, is the main loop of the control application model and is executed at the *main loop rate* of 400 Hz by default.
First, at line 4, sensor data, including accelerometer and gyroscope measurements, is updated from the `AHRS` by the `ahrs.update()` operation.
Then, at line 7, in the case of PID attitude control, the lowest level PID controller is run by the `rate_controller_run()` operation. In the case of MPC attitude control, this operation is skipped, and only implemented to conform to the same interface as the PID implementation.
Afterwards, at line 10, the control values for the four motor setpoints are output to the motors by the `motors.output()` operation.
Finally, at line 13, the *flight mode* is updated by the `fltMode.update()` operation. In the PID case, this entails execution of the higher level PID controllers, further explained in section 4.3.1. In the MPC case, the entire control algorithm is executed including matrix calculations and optimization, further explained in section 4.3.2. The computationally nondeterministic parts of the two control algorithms are executed *after* the motor outputs are updated, in order to avoid jitter caused by varying execution times. However, this introduces a delay of one sample period.
The flight mode provides additional assistance to the pilot. The `APM:Copter` autopilot features several flight modes, such as *Stabilize*, *Altitude Hold*, *Position Hold* etc. The Stabilize flight mode prioritizes maintaining the desired attitude, and is therefore suitable for the test sequence used to compare the two attitude controllers, which is described in section 5.2.

```
1  fastLoop: () ==> ()
2  fastLoop() == (
3      -- Update AHRS
4      ahrs.update();
5
6      -- Run low level attitude control
7      attController.rate_controller_run();
8
9      -- Motors output
```

```
    motors.output();

    -- Update flight mode (run higher level attitude/position
       controllers)
    fltMode.update();
);
```

Listing 4.1: Main loop in the DE model

The subsequent subsections describe the two modeled attitude controllers.

4.3.1 Proportional integral derivative

The attitude controller of the existing `APM:Copter` application is structured as a cascaded PID-controller, as seen on the IBD in figure 4.3. The responsibility of the attitude controller is to provide the `Motor control` block with target angular accelerations. These are based on the difference between the desired angular references originating from the `Pilot input` block, and the angular orientation of the quadrotor estimated by the `AHRS` block.
The Square Root `(SQRT) controllers` block calculates angular position errors as the square root of the difference between desired angular references and current angular positions. These angular position errors are multiplied with a gain to compute target angular rates by the `P controllers` block. The target angular rates are compared with the current angular rates to yield angular rate errors, which are passed to the `PID controllers` block resulting in target angular accelerations. Note that, the above description and figure 4.3 does not describe the sequence of execution, which is described in section 4.3, but solely the flow of information between the blocks.

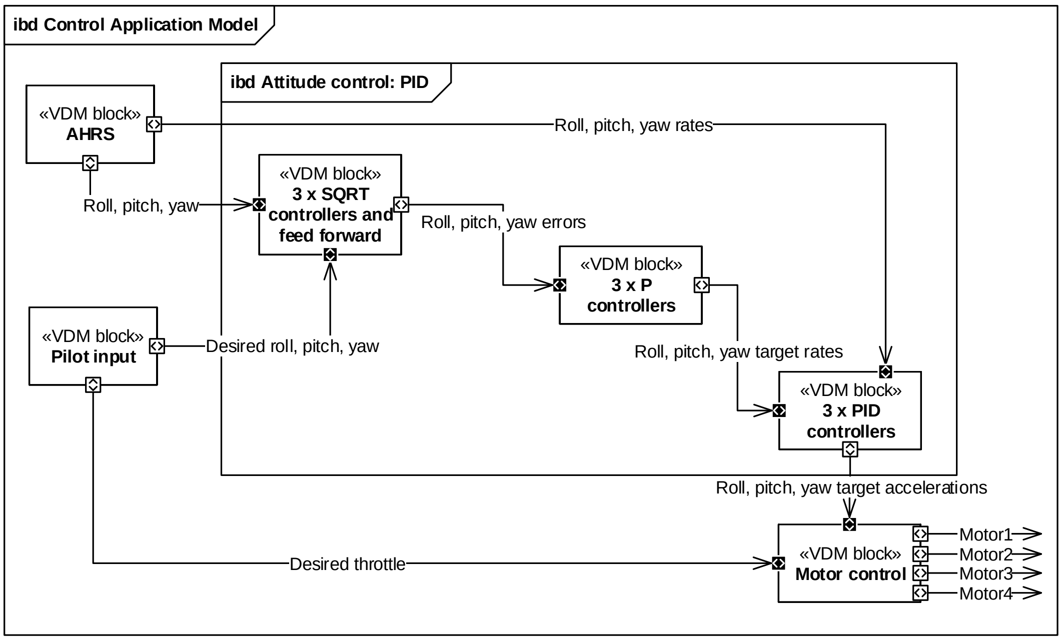

ibd Control Application Model
ibd Attitude control: PID
«VDM block» AHRS
Roll, pitch, yaw rates
«VDM block» 3 x SQRT controllers and feed forward
Roll, pitch, yaw
Roll, pitch, yaw errors
«VDM block» 3 x P controllers
«VDM block» Pilot input
Desired roll, pitch, yaw
Roll, pitch, yaw target rates
«VDM block» 3 x PID controllers
Roll, pitch, yaw target accelerations
«VDM block» Motor control
Desired throttle
Motor1
Motor2
Motor3
Motor4

Figure 4.3: IBD of the PID attitude controller model with control application dependencies

4.3.2 Model predictive control

Compared to the PID attitude controller, the MPC controller depends on the same control application blocks, as seen in figure 4.4. This property makes the two controllers interchangeable.
The internal blocks of the MPC controller differ significantly from those of the PID. As described in section 2.4, MPC can be reduced to a formulation of a Quadratic Programming (QP) problem and an optimization of this. The formulation involves a series of matrix calculations, in order to obtain the QP problem in the correct form. Once the QP problem is formulated, it is optimized with a solver library called Midaco [Schluter09].

The Midaco library is available as source code (C++). Crescendo does not support a direct call to C++. Therefore, a Java wrapper is necessary in order to call the C++ library from the model through Crescendo. The `Midaco` library is compiled to a Dynamic Link Library (DLL), with a Java Native Interface (JNI), in order to be callable from the `Wrapper`. This enables the use of the C++ version of the `Midaco` library in the DE model. The same version of the library is used in the realization, guarantying optimum fidelity between model and realization library.

Initially the QP formulation and all matrix calculations were implemented in VDM, causing a 30 seconds simulation to take more than one hour to run. To decrease simulation time, the matrix calculations were implemented in C++ and compiled to a DLL, in the same way as the Midaco library, reducing the aforementioned simulation time to approximately 5 minutes. The additional implementation effort was minimal, as the matrix calculations are used in the realization as well, and were therefore to be implemented in C++ in any case. Furthermore, the use of the same matrix calculations in the model and realization improves fidelity.

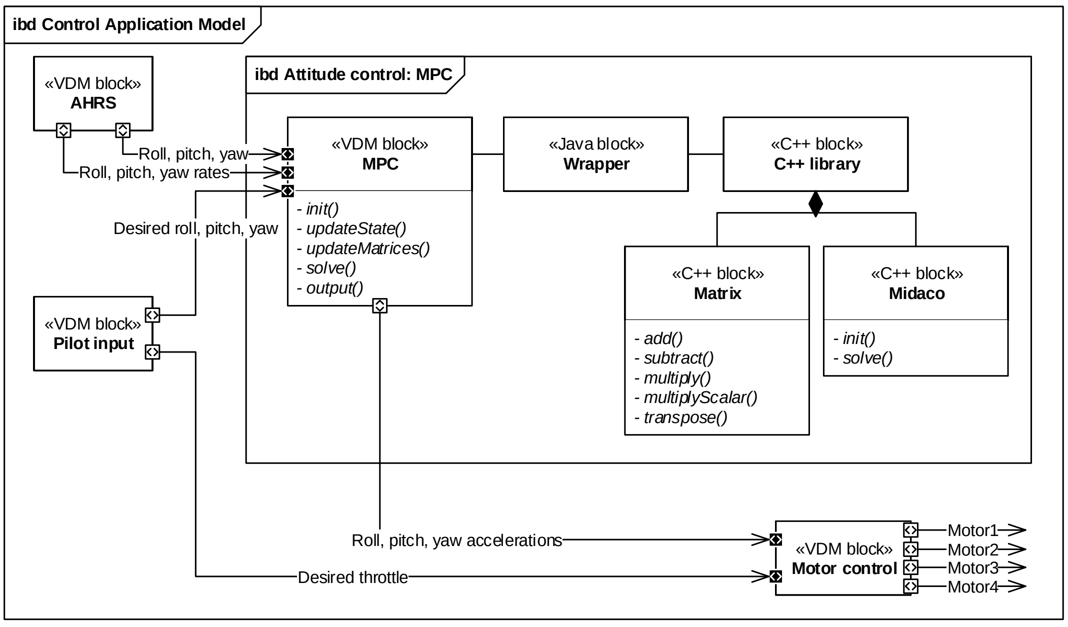


Figure 4.4: IBD of the MPC control application model with control application dependencies

The MPC attitude controller is called through the `fltMode.update()` operation from line 13 of listing 4.1. Listing 4.2 shows the `update()` operation of the MPC attitude controller. First, at line 5, the current angular positions and velocities are updated by the `updateState()` operation. Then, at line 8, the QP problem is formulated by a series of matrix calculations within the `updateMatrices()` operation. Afterwards, at line 11, the QP problem is passed to the `Midaco` solver library with the `solve()` operation. Finally, at line 12, the new target angular accelerations, contained in the `controlChanges` tuple, are output to the `Motor control` block by the `outputToMotors()` operation.

```
 1  public update : real * real * real * real ==> ()
 2  update(rollTarget,pitchTarget,yawTarget,smoothingGain) ==
 3  (
 4    -- Update angular positions and velocities
 5    updateState();
 6
 7    -- QP problem formulation by matrix calculations
 8    updateMatrices(rollTarget, pitchTarget, yawTarget);
 9
10    -- QP problem optimization by Midaco solver
11    let controlChanges : (real * real * real) = midaco.solve(mk_(
         K_seq_100,Matrix`asSequence(f))) in
12       outputToMotors(controlChanges);
13  );
```

Listing 4.2: MPC update operation from the DE model

4.4. Realization

The realization platform is a Pixhawk board with the APM:Copter application, as described in section 2.7. The APM:Copter application is an autopilot, with a number of different flight modes all based on PID controllers. One flight mode suits the purpose of a decoupled attitude control; *stabilize*. In the stabilize flight mode no position control is used. The pilot is in full control of throttle level and attitude angles, which are controlled by the PID-based attitude controller. The interface around the attitude controller is similar to the model interface in figure 4.2, making the PID and MPC attitude controllers interchangeable in the realization as well.
The implementation of the MPC controller posed a great challenge on time requirements. The Pixhawk board has a clock frequency of 168 MHz, thereby limiting the amount of instructions available in each main loop. Measurements of matrix calculations and optimization time requirements were performed, proving that a 400 Hz main loop rate could not be achieved. The following restrictions have been applied to be able to run the MPC controller on the board:

- The main loop rate is lowered to 100 Hz. Note that only 400 Hz and 100 Hz are supported by the application.
- The *prediction horizon*, which is the length of the prediction as described in section 2.4.2, is set to 30. A lower horizon, e.g. 20, results in an unstable control.
- The *control horizon*, which is the length of the predicted control input as described in section 2.4.2, is set to 1.
- The number of iterations performed by the Midaco optimizer is limited to 70.

Both horizons are set as low as possible in order to minimize the amount of matrix calculations. The number of Midaco iterations is lowered just enough to be able to run the MPC controller with a 100 Hz main loop rate. In figure 4.5 two simulations of MPC control are shown; one with 70 iterations and the second with an unlimited number of iterations. It visualizes the effect of limiting the number of iterations to 70. Only small differences are observed, meaning that the limitation do not have a large effect on system performance. However, similar tests were made with only 20 and 50 iterations, both causing the system to be unstable. The test simulation procedure is further described in section 5.2.

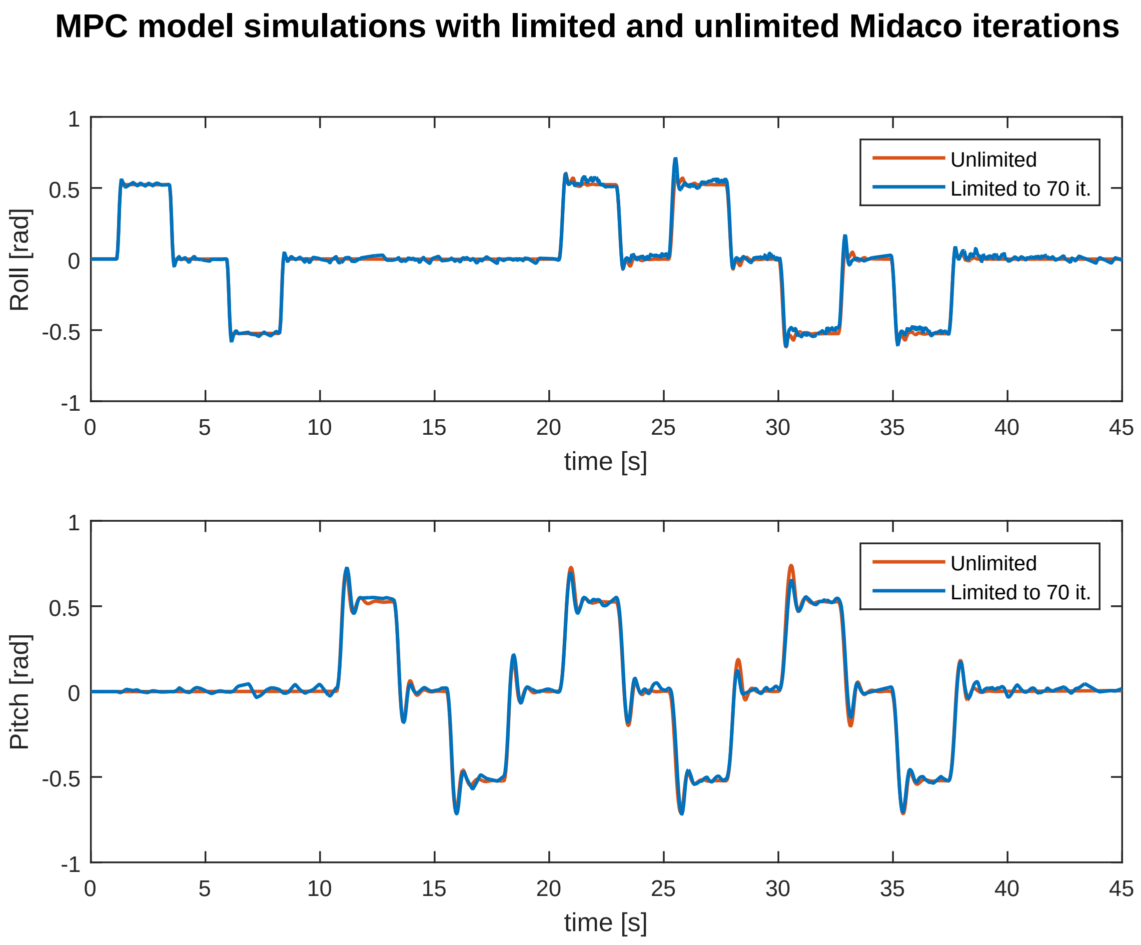


Figure 4.5: Effect of limited Midaco iterations, simulated.

A code segment of the C++ realization of the MPC attitude controller has been included in listing 4.3. It describes the `Copter::MPC_test_run()` operation, which has been used to test the performance of the MPC attitude controller as described in section 5.2.
The content of the `Copter::MPC_test_run()` operation corresponds to the `update()` VDM operation described in listing 4.2. The only exception is found at line 10, where the target angular positions used in the QP problem formulation are taken from a predefined test sequence, also described in section 5.2.

```cpp
// MPC_test_run - runs the main stabilize controller
void Copter::MPC_test_run()
{
    float target_roll, target_pitch, target_yaw_rate = 0;

    // Update angular positions and velocities
    attitude_control_mpc.updateState();

    // QP problem formulation based on the test sequence
    attitude_control_mpc.updateMatrices(test_sequence[
       test_iterator].targetRoll, test_sequence[test_iterator].
       targetPitch, target_yaw_rate);

    // QP problem optimization
    attitude_control_mpc.solve();

    // Output new setpoints to the motors
    attitude_control_mpc.outputToMotor();
}
```

Listing 4.3: MPC update function in C++ source code

4.5. Modeling and realization artifact overview

In order to provide the reader with an idea of the workload associated with modeling, realization, testing and data processing, a complete list of created artifacts is presented in table 4.1. The artifacts are categorized according to the work phases. For each artifact the used development tool and formalism is listed, together with the total Lines Of Code (LOC) and the LOC created by the authors of this thesis.
The Midaco optimization and Matrix source code have been included as both Control Application Modeling and Realization artifacts. The control application modeling artifacts are larger, due to interface code and library headers.

Table 4.1: Overview of created and modified artifacts.

Component name	Tool	Formalism	LOC total	LOC authors
Control Application Modeling				
Quadrotor DE model	Overture	VDM	3.000	3.000
Java wrapper (VDM to C++ bridge)	Eclipse	JAVA	150	150
Midaco optimization library (DLL)	Visual Studio	C++	5.900	600
Matrix library (DLL)	Visual Studio	C++	2.400	400
MPC controller development	Matlab	Script	250	250
Physical System Modeling				
Quadrotor CT model	20-sim	Diff. equations	700	700
Propulsion system refinement fittings	Matlab	Script	500	500
CAD 3D model	SolidWorks	Assembly	-	-
Realization				
APM:Copter application	Eclipse	C++	200.000	500
Midaco optimization library	Eclipse	C++	3.800	500
Matrix library	Eclipse	C++	250	250
Simulation and Testing				
Data processing and visualization	Matlab	Script	350	350

Chapter 5

Model Fidelity and Control Evaluation

Chapters 3 and 4 covered the modeling effort of both the Continuous Time (CT) and the Discrete Event (DE) models, leading to the creation of the co-model, as well as the realization details. This chapter presents the results obtained through co-simulations of the co-model and realization test measurements. Afterwards, in chapter 6, the proposed methodology extension is presented and evaluated, making use of the results from this chapter.

5.1. Introduction

A co-model can be used for many different purposes such as investigation of fault handling capabilities, Design Space Exploration (DSE) and performance evaluation. The primary purpose of the co-model in this thesis is to compare alternative control strategies. This requires an appropriate model fidelity (see section 6.4) of the modeled system and the surrounding environment. The model refinement method presented in chapter 3 covered how the level of fidelity is achieved. This chapter presents the obtained results of comparing co-simulations of the co-model, with tests of the realization, both described in chapter 4.

Figure 5.1 presents an overview of the performed tests and simulations, and how these are compared to gain knowledge about fidelity and control performance. Each box represents a simulation or test. Each arrow indicates a comparison and have been labeled with the purpose of the comparison. The comparisons and corresponding results are elaborated in this chapter.

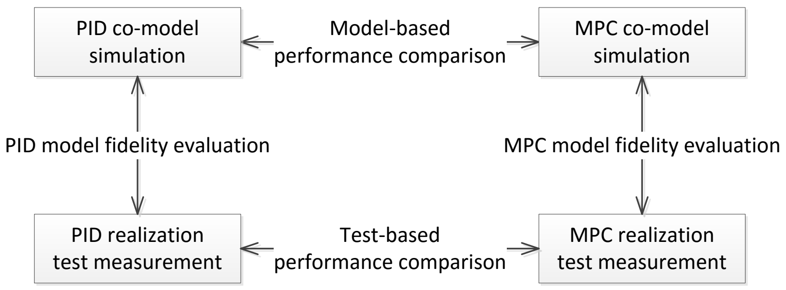

PID co-model simulation
Model-based performance comparison
MPC co-model simulation
PID model fidelity evaluation
MPC model fidelity evaluation
PID realization test measurement
Test-based performance comparison
MPC realization test measurement

Figure 5.1: Overview of simulations, tests and comparisons hereof

First, a description of the test procedure is presented in section 5.2 Afterwards, the achieved model fidelities are presented and discussed in section 5.3. Finally, the co-model's capability of comparing control strategies and the test results are covered in section 5.4.

5.2. Test procedure

In order to be able to compare co-simulations and test measurements, the effects of the testbed on the system have been incorporated into the co-model and the control application. Therefore, the moment of inertia of the testbed rings is added to the moment of inertia of the quadrotor in the CT-model. This combined moment of inertia is also used in the formulation of the state space model used by the Model Predictive Control (MPC) control. Furthermore, the center of gravity of the quadrotor has been fixed in the co-model, so that translational motion is limited. These modifications enable a comparison of co-simulations and real tests. However, a downside of this approach is that the quadrotor may not be able to fly or hover outside the testbed. Since the moment of inertia of the testbed is incorporated into the MPC controller, removing the quadrotor from the testbed could lead to instability, as the controller would still compensate for the moment of inertia of the testbed and oversteer as a result.

A test sequence is designed to exercise roll and pitch angles individually and simultaneously. This test sequence forms the foundation of comparisons between co-simulations and tests, as well as between control strategies. The specific test chosen is visible in the subsequent figures, in which it is labeled as *Reference*, since it is a reference trajectory for roll and pitch angles. The sequence starts with individual movements around a single axis, followed by simultaneous movements around both axes, as shown in figure 5.2 and listed below. Between each step the quadrotor returns to the hover position.

0. Hover
1. Roll right
2. Roll left
3. Pitch nose up
4. Pitch nose down
5. Roll right and pitch nose up
6. Roll right and pitch nose down
7. Roll left and pitch nose up
8. Roll left and pitch nose down

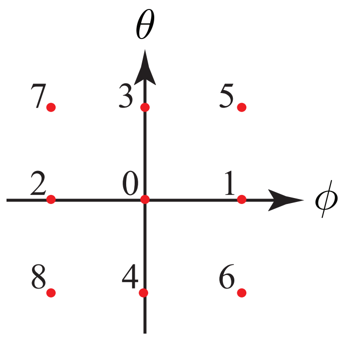


Figure 5.2: A two-dimensional illustration of desired roll and pitch angles in the test sequence

Note that each movement is a 30 degree rotation, which is equivalent to 0.52 radians.

In the co-simulations, the reference trajectories (roll and pitch) are specified through a *script* by emulating pilot input according to the test specification. The same approach is taken in the realization, where pilot input from the *telemetry control* is overridden accordingly.

5.3. Model fidelity

Model fidelity is a measure of how well a model captures the behavior of the modeled system. One way of assessing model fidelity is by determining the average difference between the behavior of the model and the behavior of the real system. The output variables considered for the quadrotor are the attitude angles roll, pitch and yaw. However, the presented results only include roll and pitch angles, due to the retention of yaw, as explained in section 3.4.2.

As illustrated in figure 5.1, the Proportional Integral Derivative (PID) co-model fidelity and the MPC co-model fidelity are evaluated separately. The model fidelity measure is based on the entire co-model and co-simulation setting, including DE and CT models, as well as the modeled environment. The model fidelity is documented through graphical comparisons of figures 5.3 and 5.4, and computations of the average output error presented in table 5.1.

5.3.1 Model predictive control

Figure 5.3 shows the results of comparing MPC co-simulation with MPC realization test. Multiple results can be extracted from this figure. First, the capabilities of the controller can be derived by comparing the angles with the reference trajectory. Secondly, the model fidelity can be derived by comparing the trajectory of the simulated and the tested MPC control.

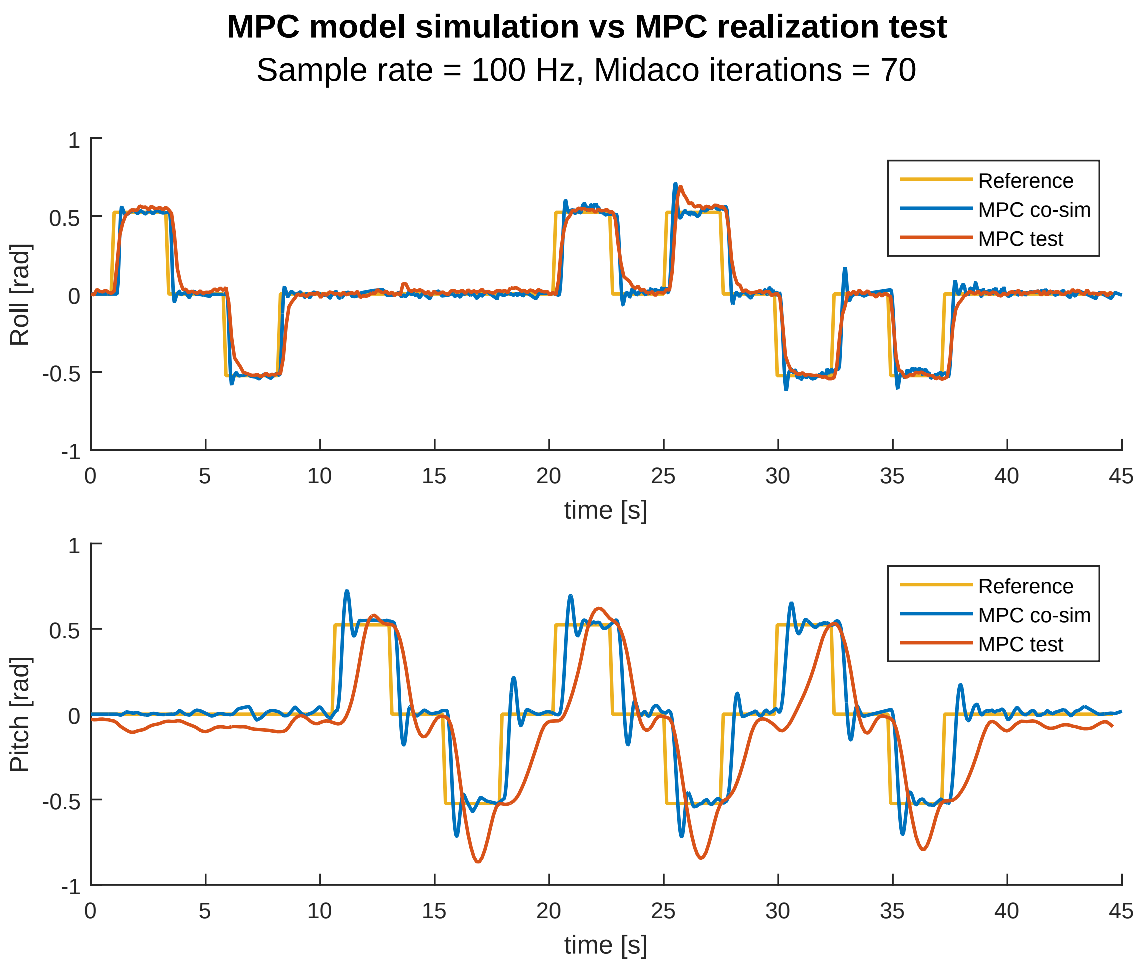


Figure 5.3: Comparison of co-simulated and tested MPC control.

Roll and pitch orientations are presented in separate graphs. By simple visual inspection it is clear that a major fidelity difference exists between them. The roll angle fidelity seems fair, whereas the pitch angle fidelity is considerably worse, which is most likely caused by the testbed. The Center of Gravity (CG) of the inner rings in the testbed is not centered in the middle of the testbed. When an object is suspended, it will rotate until its CG is located directly below the lifting point [FEMA08]. This causes an angular moment on the inner rings. As a result, the quadrotor cannot be stabilized in a hovering position without power. Instead it will be dragged towards a negative pitch angle of -90 degrees, where it is stable. The effect of this angular moment is present in the pitch results, where the negative deviations by far exceed the positive deviations. This is seen both with a reference of 0 degrees as well as in the overshoots after a transition. The overshoot after a positive rotation is minimal, whereas negative rotations have large overshoots. This is caused by the undesired angular moment from the testbed, which counteracts positive pitch movement and aids negative pitch movement.

It should be noted that the co-simulation is performed with the same parameters as the test, in order to maintain model fidelity. The most important parameters are the sample rate, which is 100 Hz, and the number of iterations performed in the Midaco solver, which is 70. As explained in section 4.4 this is the maximum sample rate and solver iterations possible on the realization platform.

5.3.2 Proportional integral derivative

Figure 5.4 shows the co-simulation and test of the PID controller. The co-simulation and test are both run with a sample rate of 400 Hz, which is the default (and maximum) sample rate available in the APM:Copter application. Note that the reference trajectory has been smoothed as part of the PID control.

The PID parameters were tuned manually on the realization platform, to the best of the authors knowledge, and copied to the model. It is the belief of the authors, that some discrepancy might have been introduced between the control application model and the control application realization. The realization was continuously updated with bug fixes and modification from the open source community, possibly affecting the attitude control. Due to time constraints these modifications were not investigated and transfered to the model. The realization version, from which the control application model was made, could not be used for testing due to a lack of proper version control. The extent of the modifications is not clear, but it may have caused some of the fidelity errors seen in figure 5.4.[1]

The roll fidelity is decent, although the simulation seems continuously dampened compared to the test. The pitch fidelity is far worse, as was the case with the MPC controller. For a 0 degree pitch reference, the same misbehavior as with the MPC controller is observed, although not quite as apparent. In the activation of pitch in the test, large overshoots are observed. This difference between the test and the simulation may be due to the model discrepancies, caused by the version mismatch, or the improper modeling of testbed effects.

[1] This could have been avoided by using a baseline approach, where the same realization version is used for both modeling and testing.

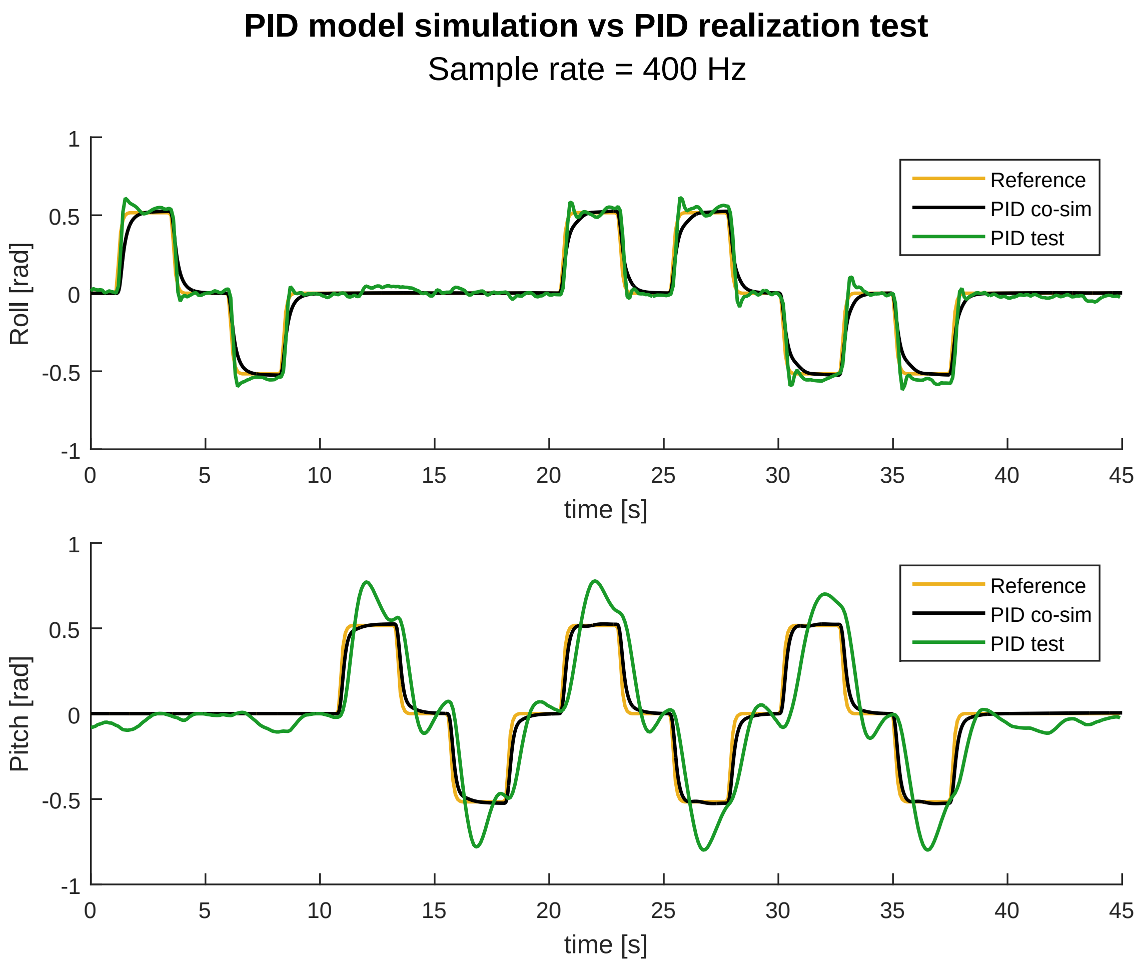


Figure 5.4: Comparison of co-simulated and tested PID control.

5.3.3 Quantitative fidelity measure

The previous sections presented a qualitative discussion of fidelity based on the graphical visualizations. In order to get a quantitative measure of the fidelity, a calculation of the average deviation of attitude angles, between the simulations and the tests, is performed. The test results are logged with a frequency of 10 Hz, whereas the simulation results are logged with a frequency of more than 5000 Hz, but not with a constant time interval. In order to correctly calculate the average deviations, the simulation results are downsampled, and the time line is equated with the test results.

The calculated average deviation of attitude angles are presented in table 5.1.

Table 5.1: Average deviations of angles between realization measurements and model simulation in degrees.

Control method	Roll deviation	Pitch deviation
MPC	1.84°	8.08°
PID	2.26°	5.80°

The roll deviation with the PID controller is only slightly higher than with MPC, likely caused by the DE version mismatch presented earlier.
The deviations in pitch are considerably higher than roll and most likely caused by model inaccuracies in the CT model. Additionally, the pitch deviation with MPC is higher than that of PID. One logical explanation is, that MPC is affected more by CT model inaccuracies, which makes sense, since the MPC control is dependent on an accurate state space model. The state space model contains the same inaccuracies as the CT model. As a result, the MPC controller in the DE model is also affected. Since the PID controller is not model-based, the CT model inaccuracies will not affect the PID controller.
It is interesting to note that with an accurate state space model, a higher model fidelity is achieved with the MPC controller as seen on the roll deviation. However, due to the sensitivity to inaccuracies in the state space model, the fidelity of the MPC model is considerably worse in the pitch case.
Lower pitch deviations would be preferable, but an average deviation of 1.84 degrees for the roll angle with the MPC controller is a great result, considering component uncertainties.

5.4. Control evaluation

The main purpose of the co-model is to compare control strategies. However, the reader should bear in mind, that comparison of control strategies is not a main objective of this thesis. Figures 5.5 and 5.6 present graphical comparisons of MPC and PID in simulation and test environments. A quantitative comparison is not performed, nor is it guaranteed that the control parameters are tuned optimally. This could be carried out by an expert within the control domain and is left for future work, as described in section 7.3.1.
In both simulation and test settings, the PID controller is run with a frequency of 400 Hz whilst the MPC controller is run with a frequency of 100 Hz, due to time constraints on the test platform. The significance of a changed frequency was investigated by testing the PID controller with both 100 and 400 Hz. Based on a visual comparison, no noteworthy difference was seen.
In the co-simulation comparison in figure 5.5 the PID results are heavily dampened, whereas the MPC results have significant overshoots, especially in the pitch results. This PID damping was addressed in section 5.3, and could likely be reduced with different PID parameters tuning, while the overshoots in MPC could be decreased by changing the weights in the objective function (see section 2.4).
Figure 5.6 shows the comparison of the MPC and PID control tests on the realization platform. The roll and pitch performance of the two controllers are comparable, although the pitch results are slightly deviating.
The high impact of the testbed on pitch rotations causes a high diversity between roll and pitch results. However, it is expected that the roll and pitch performance would be similar, if the controllers were modeled and tested outside the testbed. Other considerations regarding removing the quadrotor from the testbed, such as controller stability, robustness to disturbance, yaw performance etc., are left for future work.

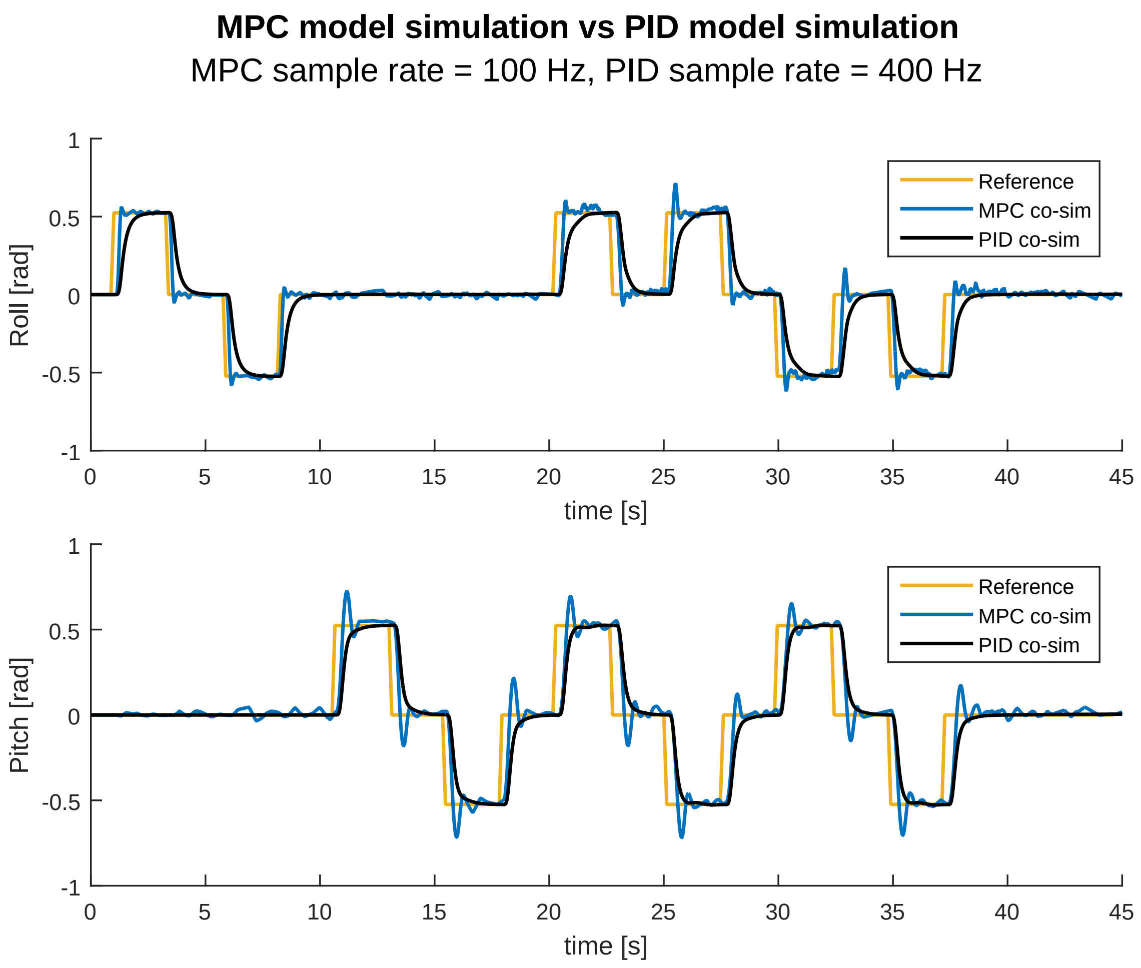


Figure 5.5: Comparison of MPC and PID control through co-simulations

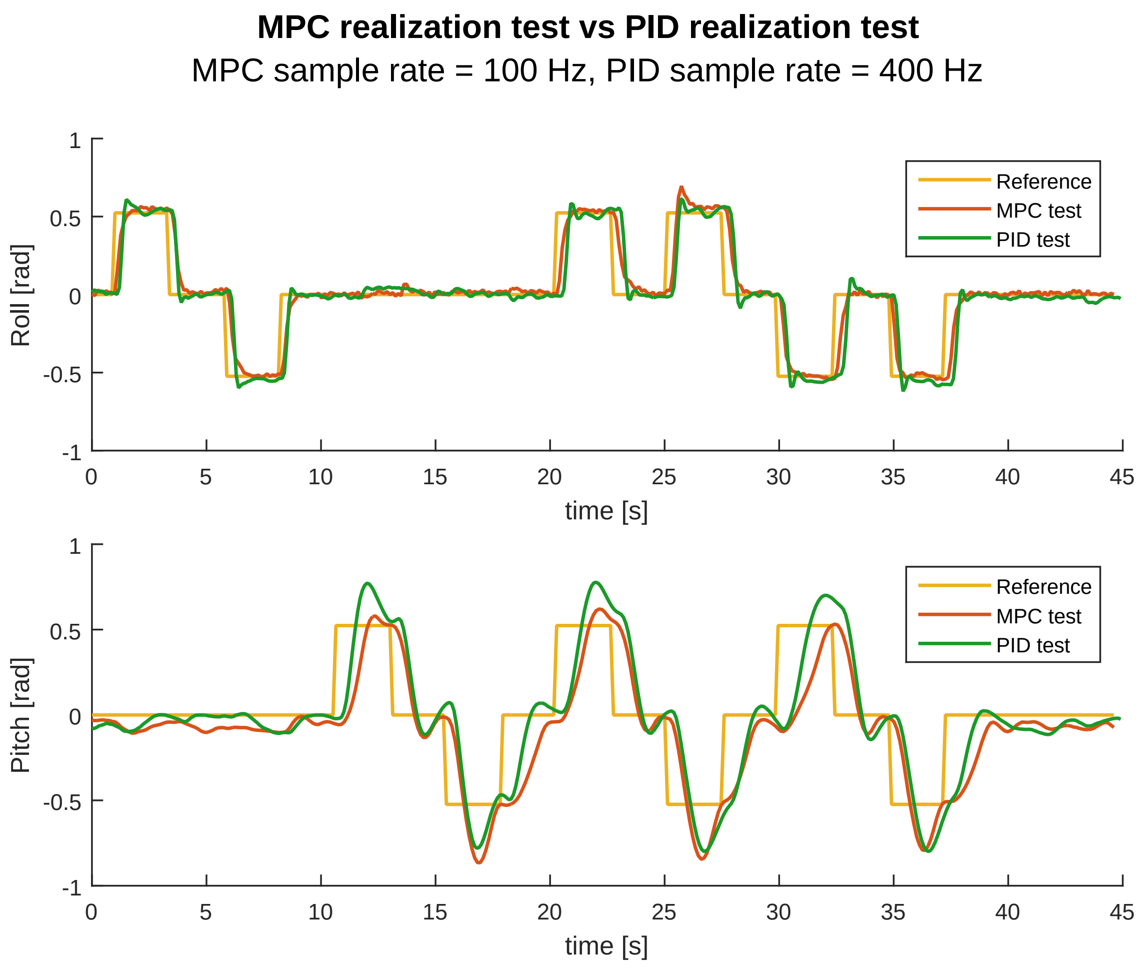


Figure 5.6: Comparison of MPC and PID controllers through tests

Chapter 6

Methodology Extension and Evaluation

Chapters 3 and 4 have described how an extended CT-first methodology has been applied to model a quadrotor Unmanned Aerial Vehicle (UAV), the results of which have been presented in chapter 5. This chapter presents the applied methodology extension of the CT-first approach including an example and an evaluation. Finally, chapter 7 concludes this thesis by presenting the achieved results and the future work.

6.1. Introduction

When using a model-based development approach on a complex system, an in-depth understanding of system functionality is essential for modeling the system competently. A *competent* model should accurately exhibit the function of the system, to the extent that the modeling goal requires. The competence is challenged by system complexity, which argues that a methodology with good complexity management capabilities is suitable for complex systems.
In the conventional CT-first approach, initial modeling is performed in the Continuous Time (CT) modeling domain with focus on developing a single CT model of physical system dynamics [Fitzgerald&14]. This chapter presents an extension to the conventional CT-first approach, which expands the physical system modeling process into a three-step approach. The three-step approach involves a *conceptual*, a *generic* and a *specific* model. This refinement aids the modeler in obtaining system understanding.
Following this introduction, section 6.2 introduces the structure of the methodology extension. This is followed by a comparison of two model refinement methods in section 6.3 and a discussion of appropriate model fidelity in section 6.4. Afterwards, the concept of the methodology extension is clarified through an example of an Anti-lock Braking System (ABS) for a Tesla Model S in section 6.5. Finally, the extended CT-first methodology is evaluated in section 6.6.

6.2. Methodology extension

The underlying concept of the extension is to achieve better complexity management with the use of abstraction. The extension is structured as three subsequent models, each adding more detail and system understanding (see figure 6.1).
It should be noted, that each of the individual models has a specific *purpose*, which is related to,

however not to be confused with, the overall modeling *goal*. The overall modeling goal reflects the modelers overall incentive for modeling the system. The purpose of any specific model reflects what intended insight is to be gained from modeling and simulating that particular model.

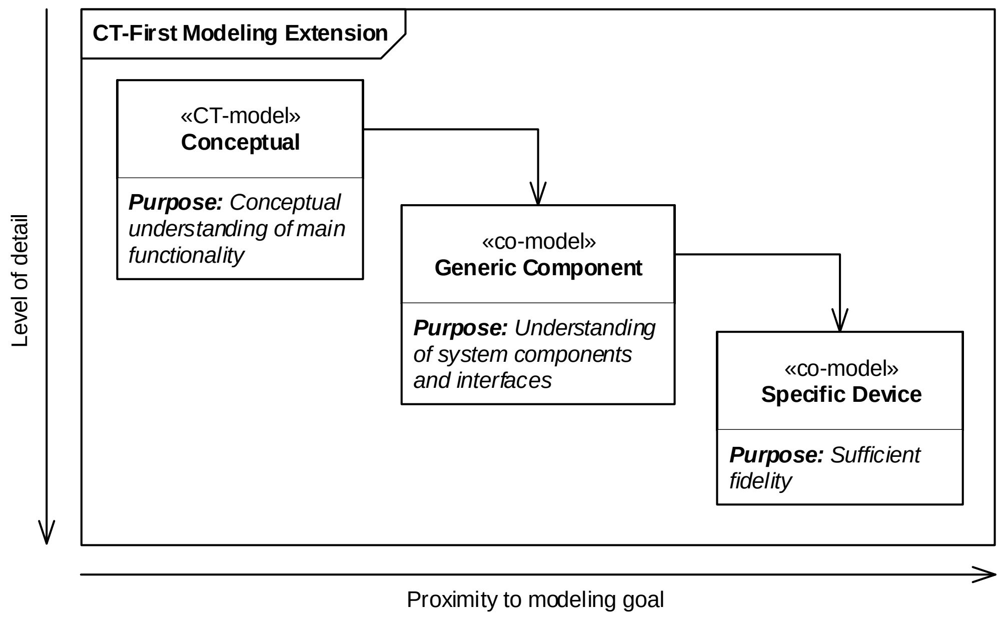

CT-First Modeling Extension
«CT-model» Conceptual
Purpose: Conceptual understanding of main functionality
«co-model» Generic Component
Purpose: Understanding of system components and interfaces
«co-model» Specific Device
Purpose: Sufficient fidelity
Level of detail
Proximity to modeling goal

Figure 6.1: Illustration of the three-step refinement of the physical system models

The conceptual CT-model: The first step involves creating a CT-model, which captures the *main* functionality of the system. It is important to make the distinction between main functionality and auxiliary functionality. The main functionality should be considered as the single most important function of the system and should answer *what* the system does. Physical properties, such as forces and accelerations related to the main functionality, can be included, to answer *how* the main functionality is realized. The origin of the physical properties however, is left as a concern for later. If prior domain knowledge renders the conceptual model pointless, in the sense that no additional system understanding is gained, the conceptual model can be omitted.

In a commercial setting, a single conceptual model may be sufficient for a line of products in one area. Such a model may be useful for internal training purposes. A conceptual model of a Tesla Model S is exemplified in section 6.5.1.

The generic component co-model: Once the fundamental understanding of the system has been obtained, the second step of the extension continues with a *system decomposition*. The purpose of the generic component co-model is to identify, which components should be included in the co-model based on their significance with respect to the modeling goal (as in figure 3.3). Components that are not relevant for the modeling goal should be omitted. Each included component should be modeled as simple as possible, but still detailed enough to be able to verify the interface between components (as in figure 3.4). Here, it is assumed that at least one relevant system component should be modeled in the Discrete Event (DE)

modeling domain. The component model is therefore assumed to be a co-model. For a company, it would probably make sense, to produce only one generic component co-model for a product series.
A generic component model of a Tesla Model S is exemplified in section 6.5.2.

The specific device co-model: The final step of the methodology extension involves refining the generic component model to obtain a co-model, which exhibits the behavior of the particular system in question.
The refinement entails increasing the level of detail of each individual component by the amount required by the modeling goal. Furthermore, the values of all co-model parameters should correspond to the real system. For a company, it would probably make sense, to produce such a device specific co-model for each specific product.
Refinement methods for high levels of fidelity are elaborated on in section 6.3. A partial specific device model of an ABS system for a Tesla Model S is exemplified in section 6.5.3.

Note that the refined subsequent models may not be related to the initial models in a formal setting, since aspects may be entirely abstracted away in the initial models [Back78].

6.3. Model refinement methods

Ideally, mathematical transfer functions can be defined for all physical model components, and all relevant parameters are available from data sheets. However, a modeler can be faced with a fidelity requirement which supersedes the readily available information on the system. If the physical system is available to the modeler, experiments can be performed to determine the missing parameters. Two fundamentally different approaches are proposed:

Transfer function: The theoretical approach defines components with a *transfer function*, which is a mathematical expression with a number of parameters. An example could be, to define a transfer function for a rotor, which provides the lift force given a rotational speed. The parameters of the transfer function depend on the physical parameters of the rotor such as size and shape. All parameters would have to be measured.
The benefit of this approach is that the Design Space Exploration (DSE) capability of the model remains intact. The drawback however, is that model fidelity might suffer due to simplifying assumptions made during the construction of the transfer function.

Look-up table: The practical approach involves performing an experiment with the same input and output as the transfer function mentioned above. The resulting data is placed in a *look-up table*, which is made available to the model at runtime. The data points of the look-up table are connected with linear interpolation. In the rotor case, an experiment measuring the lift force would be performed at ranging rotor speeds. When the rotor lift force is needed in the model, a lookup of the table is performed based on the rotor speed.
This approach achieves the highest possible fidelity, with only experiment uncertainties and interpolation induced inaccuracies differing the model from the physical system. The downside is that the DSE capability of the component in question is lost.

These two approaches represent a trade-off between DSE capability and model fidelity. A compromise can be made between them however, as exemplified in section 3.5. A mathematical model can be fitted onto measurement data from experiments, to produce a transfer function based on

empirically determined coefficients, see appendix C for details. The resulting fidelity will be worse than that of the look-up table approach, by an amount equal to the error introduced in the mathematical regression.
However, an abstract form of DSE will be possible through a *better or worse* assessment. In the rotor case, the value of investing in a rotor which produces 20 % more lift force could be simulated. Alternatively, if a systems performance is impeded by a bottleneck component, the question of, *"How much better does this component need to be, in order to achieve some system-wide performance goal?"*, could be answered.
However, the modeler should be aware of the fact, that increasing a coefficient for a certain component property does not take into account the effect on the other component properties. For instance, how will the increased aerodynamic drag, resulting from a rotor with a higher lift coefficient, affect the battery drainage and hence the flight time of the quadrotor? In order to account for the remaining component properties and maintain model cohesiveness, the component would have to be acquired and experiments re-run.

6.4. Appropriate model fidelity

The methodology extension, and modeling in general, rely heavily on the goal of the model to define which level of fidelity the components should be modeled at. This discussion attempts to provide a fidelity level guideline for a few examples of a model goal. Generally, due to expenses such as time, the level of fidelity should not exceed what is required to provide the modeler with the desired insight into the modeling goal.

First, it should be noted, that there is a practical limit to the level of model fidelity, that can be reached. Aside from the unrealistic scenario, where every single component of a system is uniquely identified and accurately measured[1], component tolerances will limit model fidelity. Even if the modeler went through the trouble of measuring every single component, the high level of fidelity would only apply to the unique system on which the measurements were performed. Monte Carlo methods could be used to account for component tolerances and provide worst-case simulations [Gao&95]. This could be simulated with Automated Co-model Analysis (ACA) [Fitzgerald&14], where the simulation is performed multiple times with different component values.

Two examples of modeling goals, one case of a performance increase feasibility study and another of a contingency plan analysis, are presented below. The purpose of the examples is to clarify how different modeling goals require different levels of fidelity.

Performance increase feasibility study: In order to ascertain whether or not it is possible to improve the performance of a certain system by a fixed amount, a model-based feasibility study is performed.
A *specific device co-model* would be required for this particular modeling goal. Components related to the performance measure are enhanced within the model, until the performance target is met or exceeded. If the new system configuration is realizable, then the performance increase can be considered feasible.
However, in order to obtain a conclusive result from the feasibility study, the simulated

[1] Every single resistor, capacitor etc. It is assumed that all system components are deterministic.

system performance must exceed the performance target by an amount equal to the fidelity uncertainty. A fidelity requirement can then be specified based on the desired performance increase.

Contingency plan analysis: In order to determine how a system should react to critical component failure, a model-based contingency plan analysis is performed.
A *generic component co-model* would suffice for this particular modeling goal. Once contingency procedures for the critical components have been modeled, single or multiple component faults could be injected into the simulation. This would show the impact of the faulty component(s) and contingency procedure(s) on the rest of the system.
The model fidelity should be high enough to ensure that all component and functionality interdependencies are clarified. In other words, the system impact of a component failure in the model and in the real system must be equal on a component level. An example could be, a fault in component `X` affects components `Y` and `Z` and as a result, system function `A`.

6.5. Example: Anti-lock braking system

To exemplify the extended CT-first approach, the outline for a Tesla Model S is drawn in this section. The goal of the model is to assure that a newly developed ABS braking system performs within the regulatory definitions for all weather conditions.

6.5.1 Conceptual model: The moving box

At a conceptual level, the Tesla Model S is a vehicle. A vehicle can be considered as a box moving on top of a surface. Assuming front wheel drive, the motion is caused by forces on the front corners, as seen in figure 6.2. The size and direction of the forces on the front corners can be manipulated to accelerate the box and steer the movement. The forces are constrained to be parallel.

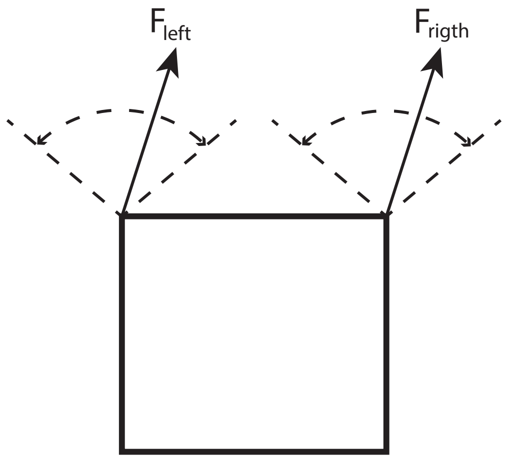


Figure 6.2: Visualization of the conceptual moving box seen from above

Such a conceptual model can be used to obtain a better understanding of the translational motion of a vehicle whilst braking. Also, the effect of varying the surface friction to reflect different weather conditions can be investigated.

6.5.2 Generic component co-model: Fundamental vehicle

The generic component co-model should identify which system components are relevant with respect to the modeling goal. Figure 6.3 shows a Block Definition Diagram (BDD) of the components likely relevant to an ABS braking system for a vehicle. A description of the relationship between the components (as in figure 3.4) has been omitted, as the reader is expected to be familiar with the relationship of vehicle components.
A generic component co-model can be used to investigate how the individual system components affect the modeling goal. If a certain component proves decisive, it may be the subject of a more detailed modeling effort as part of a specific device co-model.

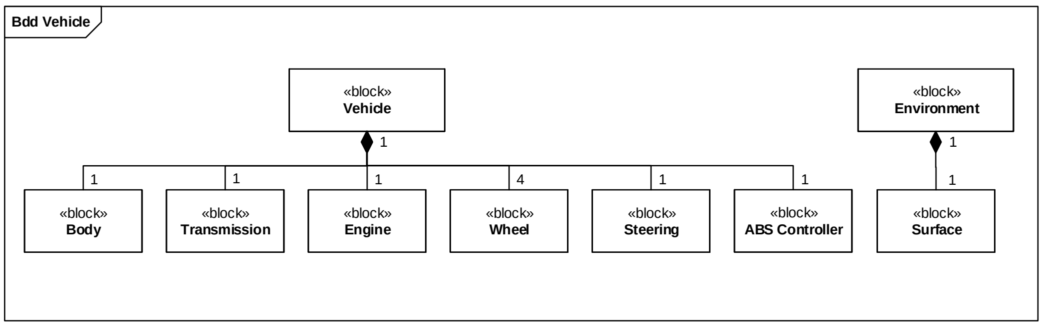


Figure 6.3: BDD of the ABS braking related components of a Tesla Model S

6.5.3 Specific device co-model: Tesla Model S

The specific device model should refine the generic component model and add subcomponents relevant to the modeling goal; the assurance of the ABS braking system of a Tesla Model S. Here, a partial specific device model, showing a refinement of the `Wheel` component from figure 6.3, has been included in figure 6.4. The components of the Wheel are briefly described below.

Rim The `Rim` block represents the center part of the wheel and is used to keep track of the direction of the wheel, which influences the braking length.

Brake The `Brake` block represents the braking assembly including discs, brake pads and hydraulics (disc brakes assumed). The `Brake` component contains parameters such as the friction coefficient (temperature dependent) and contact area size between the disc and brake pads. Additionally, the brake pressure, which is a measure of how hard the braking pedal is pressed down, is also modeled in the `Brake` block.

Tire The `Tire` block represents the rubber outer part of the wheel and models the friction coefficient (weather dependent) and contact area between the tire and the surface, on which the vehicle is moving.

Sensor The `Sensor` block represents the sensing mechanism, which detects if the wheel is blocked. The modeled sensor could achieve this by monitoring the angular position of the wheel and the velocity of the vehicle. If the angular position of the wheel is constant for a non-zero velocity, the wheel can be considered blocked.

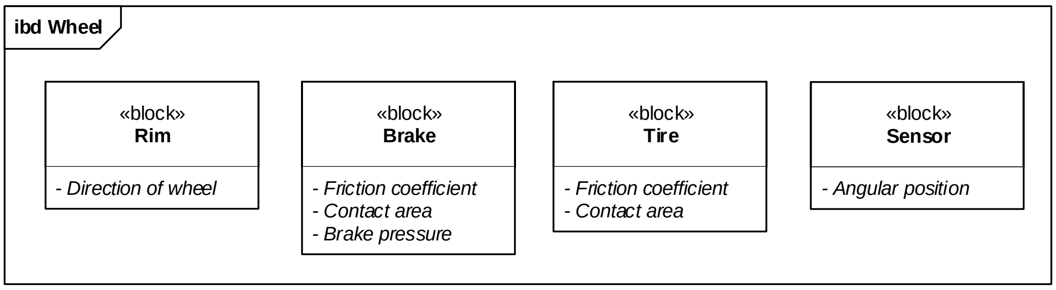


Figure 6.4: Internal block diagram of the Wheel from figure 6.3

A specific device co-model could be used to perform DSE on a highly detailed subcomponent level, revealing how individual parameters affect system performance. Here, the interaction between the `ABSController` (figure 6.3), the `Sensor` and the `Brake` could be analyzed to determine the best braking pattern in terms of stopping distance. The braking pattern is considered to be the pattern with which the brake is engaged and disengaged when the wheel is blocked.

6.6. Extended CT-first methodology evaluation

The extended CT-first methodology is evaluated with respect to the conventional CT-first methodology within the Evaluation Categories (EC) listed in section 1.4. A direct comparison between the CT-first and the extended CT-first methodologies, based on seven *advantage* evaluation criteria, is presented in section 6.6.1. Based on the advantage evaluation, section 6.6.2 provides a broader perspective discussion of the *relevance* of the extended CT-first methodology.

6.6.1 Evaluation Category 1: Advantage

Figure 6.5 shows a comparison of the CT-first and the extended CT-first methodologies, with respect to the seven evaluation criteria presented below. The results of the comparison are illustrated on a radar chart made by the authors. Each criteria is given a ranking between 0 and 5, where 5 is the best rank.

Complexity management This criterion assesses how well the methodologies aid the modeler in managing complexity. The extended CT-first is given a higher rank, since the conceptual and generic component models provide system and modeling goal understanding.

Model realization This criterion assesses how well the methodologies support realization of the modeled system. The methodologies are considered equal in this respect.

Maintenance This criterion assesses how much effort has to be invested into maintaining the model as a result of modifications made to the modeled system. Since the extended CT-first methodology features three modeling artifacts, it requires a larger maintenance effort and is therefore considered worse in this respect.

Reusability This criterion assesses how well the methodologies support reuse. There is no apparent reuse opportunity with the conventional CT-first approach except with a very similar system *and* modeling goal.

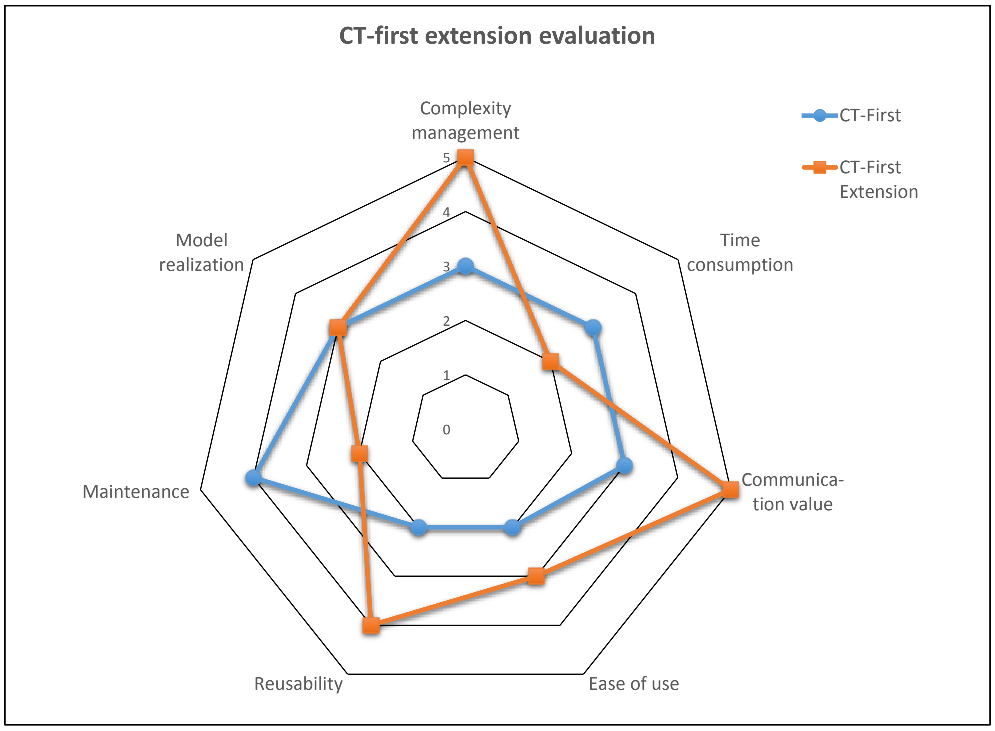


Figure 6.5: Methodology evaluation radar chart

The extended methodology on the other hand supports reuse from the conceptual model to other conceptually similar systems. The generic component model can be reused to address a different modeling goal on the same system, or to model a system of similar components. Efficient reuse may be realized through a kind of library with such reusable components.

Ease of use This criterion assesses how easily the methodologies can be applied by a novice modeler. The conventional CT-first methodology exposes the modeler to the entire complexity of the system at once. The conceptualization and decomposition of the extended methodology reduces the amount of complexity, that the modeler needs to comprehend, which makes the modeling process easier to grasp.

Communication value This criterion assesses how well the methodologies support communication of the value provided by the models. Depending on tool support, both methodologies support visualization of the specific device model. The extended methodology however, also supports communication of more abstract views of the system, which may be valuable depending on the target audience. Therefore, the extended methodology is considered to have a better communication value.

Time consumption This criterion assesses how much time is spent on modeling with the two methodologies. It is expected that the extended CT-first approach will be more time consuming due to the creation of the first two models. Additionally, the inferior maintainability of the extended CT-first will consume more time during the lifecycle of the modeled system. However, the complexity management advantage of the extended methodology may save time, in cases where the complexity of the system can cause modeling errors. Likewise, the reusability advantages of the extended methodology may save time, in circumstances where the initial models can be reused as described above. The extended methodology is expected to be slightly less favorable with respect to time. Although, this depends heavily on the modeling task in question

6.6.2 Evaluation Category 2: Relevance

Based on the evaluation of figure 6.5, the extended CT-first methodology should be used instead of the conventional CT-first methodology. That is, if the expected value from the better complexity management, reusability, ease of use and communication opportunities warrants the additional time spent on modeling and maintenance. An example could involve the modeling of a highly complex product, where the model needs to be reused for an entire product series, employees of ranging modeling competence are to take part in the modeling and the model should be communicated to diverse stakeholders. If the aforementioned criteria provides no value for the modeler, the conventional CT-first methodology should be used.

The extended CT-first methodology should not be considered for modeling tasks unfit for the conventional CT-first methodology. However, the fundamental rationale of the extension, which is to expand the physical modeling process into three consecutive steps, can be applied to any physical modeling scenario - also in conjunction with a DE-first methodology.
Furthermore, the methodology extension is highly suitable for acquiring modeling competence, because it involves modeling at different levels of abstraction. The exploitation of abstraction is a key competence for any modeler. Another decisive factor, for a methodology intended for novice users, is the required proficiency in the modeling tool in use. This is gradually increased when applying the methodology extension. These aspects suggest that the methodology extension could be exploited in an employee training or teaching setting.

Chapter 7

Concluding Remarks and Future Work

In chapter 1 the thesis goals were presented, for which the required theory and technologies were introduced in chapter 2. Chapters 3 and 4 have described how an extended CT-first methodology has been applied to model a quadrotor UAV, the results of which have been presented in chapter 5. Based on the approach used, chapter 6 has described the methodology extension and evaluated it with respect to the conventional CT-first approach. In this chapter the achieved results and envisaged future work activities are presented.

7.1. Introduction

This thesis has demonstrated how collaborative modeling can be used to illuminate a technically challenging aspect of a complex Cyber-Physical System (CPS). The attitude controller of a quadrotor Unmanned Aerial Vehicle (UAV) can be considered as a technically challenging aspect of a system consisting of multiple domains. The constructed model combines mechanical, electrical, software and control engineering domain knowledge to provide attitude control evaluation capability. An extended CT-first approach has been used, which extends the conventional CT-first approach by expanding the physical modeling process into three consecutive steps. This improves a number of methological characteristics at the expense of the modeling time and maintenance effort, as described in section 6.6.
This chapter presents the achieved results in section 7.2. This is followed by a description of the envisaged future work activities in section 7.3. Final remarks conclude this thesis in section 7.4.

7.2. Achieved results

Recall the goals of this thesis presented in chapter 1:

G1 To develop a collaborative model of an existing complex CPS with the purpose of evaluating control strategies, using a CT-first approach.

G2 To suggest an extension to the CT-first methodology.

G3 To evaluate the extended CT-first methodology.

The thesis goals are considered achieved as stated briefly below. Details, elaborating the assessment are presented in the sub-sections following.

G1 achieved: A CT-first approach has been used to develop a collaborative model of a quadrotor UAV, which has been successfully used to compare two attitude controller strategies: Proportional Integral Derivative (PID) and Model Predictive Control (MPC).

G2 achieved: A CT-first methodology extension has been suggested, which extends the conventional CT-first approach, by expanding the physical modeling process into three consecutive steps: A *conceptual* CT model, a *generic component* co-model and a *specific device* co-model.

G3 achieved: The extended CT-first methodology has been evaluated with respect to the conventional CT-first approach, based on *advantage* and *relevance* criteria.

7.2.1 Goal 1: Use a CT-first approach to develop a collaborative model

A collaborative model of a DJI F450 Flamewheel quadrotor UAV has been developed based on the theory and technologies presented in chapter 2. The co-model consists of a Continuous Time (CT) model capturing the dynamics of the physical system, as described in chapter 3, and a Discrete Event (DE) model capturing the control application, as described in chapter 4. A MPC attitude controller was developed and compared to the existing PID controller in both simulation and testing environments, the results of which are presented in chapter 5. The activities below summarize the work performed, related to this goal, during the preparation and execution of this thesis:

Modeling competence acquisition Software modeling competence was initially gained from *Modeling of Mission Critical Systems* course, which was followed by a co-modeling principles and tools study from the book *Collaborative Design for Embedded Systems – Co-modelling and Co-simulation* [Fitzgerald&14]. Quadrotor research, based on a number of articles[1], uncovered a competence gap within modeling of dynamical systems, which was closed through the creation of the conceptual CT model of a quadrotor.

Collaborative modeling A quadrotor decomposition and analysis of the control application led to the creation of the DE model. A contract was constructed to enable co-simulation of the DE and CT models, which constitute the generic component co-model.

Co-model refinement The acquisition of the DJI F450 Flamewheel quadrotor and testbed initiated the co-model refinement process. The fidelity level of the propulsion system components was increased, testbed modifications and mounting brackets were constructed, refinement experiments and mathematical regressions performed, resulting in the specific device co-model.

MPC controller development The theoretical foundation for MPC was acquired (described in section 2.4) leading to mathematical development, DE modeling, co-simulation, realization, including matrix algebra and Quadratic Programming (QP) solver, and finally testing in the testbed.

Data processing and visualization In order to compare co-simulation results with test measurements, data processing, including re-sampling and sample alignment, was performed.

[1] [Mahony&12, Lee&10, Hoffmann&07, Hoffmann&11, Bouabdallah&04, Bouabdallah&07b, Carrillo&13, Czyba&12, Hamel&02, Huang&09, Orsag&12, Tayebi&09]

7.2.2 Goal 2: Suggest an extension to the CT-first methodology

As described in section 1.3, the onset of this thesis work was based on the authors motivation to explore and become proficient within model-based development. As a result of the thesis preparation the authors were inspired to suggest a methodology extension to the CT-first approach. The conventional CT-first approach is extended by expanding the physical modeling process into three consecutive steps, as is described in chapter 6. The extension is based on the experience gained by the authors during the activities described in section 7.2.1. The empirical origin of the underlying reasoning for the methodology extension, explains the lack of scientific methodological references to support the structure of the extension.

7.2.3 Goal 3: Evaluate the extended CT-first methodology

The extended CT-first methodology has been compared with the conventional CT-first methodology, as described in section 6.6. The comparison was based on complexity management, model realization, maintenance, reusability, ease of use, communication value and time consumption. It showed that the extended methodology is superior or equal in all regards except for maintenance and time consumption, as seen in figure 6.5. Hence, the extended CT-first methodology should be used, when the benefits of the remaining criteria outweigh the additional time spent on modeling and maintenance. Additionally, due to the modeling competence and tool proficiency gained when applying the methodology extension, the extended CT-first approach is suitable for employee training or teaching purposes.

7.3. Future work

This section describes some of the future work envisaged regarding model based development in general and MPC for quadrotors.

7.3.1 Control tuning

As described in section 5.4, an optimal control tuning is beyond the scope of this thesis. Additionally, the optimum tuning often depends on the application. Control performance is often evaluated based on *rise time*, *overshoot*, *settling time* and *steady state error*. Some of these measures are conflicting, e.g. improving the rise time of a system will most likely cause a larger overshoot, which is often undesirable.
The authors imagine two possible ways of improving control tuning:

Expert tuning: The co-model can be used to communicate the control tuning problem to a control domain expert, who has the needed capabilities to tune the control.

DSE tuning: The optimum control parameters could be found through Design Space Exploration (DSE) by using the Automated Co-model Analysis (ACA) feature in the Crescendo tool. This enables the modeler to run an arbitrary amount of co-simulations with alternative control parameters. The results of these co-simulations could be compared based on the aforementioned performance criteria. As the criteria are conflicting, optimum Pareto points [Teich01] could be created from the results, giving the modeler a number of optimum control parameter settings to choose from.

Another important issue, with the co-model created in this thesis, is that it incorporates the testbed in both the model and the MPC control. This means that the quadrotor might not be able to hover or fly outside the testbed. It would be interesting to remove the testbed from the control and test the quadrotor without it. This could clarify some of the unknowns: is roll and pitch performance equal? How is the yaw performance? Is the MPC stable without the testbed damping? Is it robust against external disturbance?

7.3.2 Model predictive control

Not all aspects of MPC have been investigated in this thesis. On the contrary, only the most simple linear MPC control has been implemented. The state space model of the implemented MPC controller is linearized around a hover state. The further the state of the quadrotor is from a hover state, the larger the mismatch between the model and the reality. One way to reduce this mismatch is with the use of Switching Model Predictive Control (SMPC), where multiple state space models, linearized around different state points, are defined and used in a switching manner [Alexis&11].

A more comprehensive approach is to use Learning Based Model Predictive Control (LBMPC) [Bouffard12, Aswani&11, Aswani&12]. In LBMPC the state space model is learning based, meaning that the control will, at run-time, adjust the state space model in an adaptive manner, based on predictions and measurements. As a result, even changing the payload on a quadrotor would not require new control parameters, as is the case with standard MPC.

If LBMPC is to be implemented on the Pixhawk platform, some technical challenges must be dealt with. First of all, a faster QP problem optimizer is needed. The used Midaco optimizer is not designed for speed or small optimization problems with only a few variables. Three other optimization algorithms for LBMPC are compared by [Aswani&14], which could be of use.

Secondly, in order to reduce time spent on matrix calculations the sparsity of the matrices involved could be exploited. An optimized sparse matrix library should be used.

7.3.3 The SIDUR project

The work of this thesis could be used in a forthcoming project called *Safety Innovations for Drones/UAVs enhancing the Reliability (SIDUR)*. This project will target safety and reliability of UAVs by improving individual components and system capabilities of handling component faults. As an example, a motor on a UAV could be considered. Currently, a motor is driven with a one way communication. That is, a signal is sent to the motor, causing it to spin at a certain speed. If the motor breaks down, no information hereof is sent back to the system controller. In order for the system to be able to handle this kind of fault, two issues must be solved. First of all, the motor itself must provide two-way communication, enabling it to send an error message to the system. Secondly, the system must be able to react to the motor error. A plausible scenario is to change to a control strategy, which can safely bring the UAV to a safe state (land in a safe place). Such fault handling capabilities will be developed with a model-based approach and guaranteed through co-simulations and formal model verification. Furthermore, the modeling effort should provide proof to regulation authorities, to relax the high restrictions on UAVs, to enable a broader use of UAVs in a commercial setting.

7.4. Final remarks

This thesis set out to explore the discipline of model-based development, to experience the tools and technologies and to contribute with the knowledge gained in the process. A collaborative model of a quadrotor UAV has been used to compare existing PID and newly developed MPC control strategies, leading to a methodology extension for the CT-first model-based approach. It was expected that the MPC strategy would perform better than the PID, however neither simulations nor tests were able to definitively demonstrate this.

The evolution of technology and the ingenuity of the people using it will continuously lead to more complex, multi-domain systems in the future. A portion of these systems will offer commodities such as communication, automation and transportation, making them mission critical systems for which assurances must be provided.
Collaborative modeling is envisaged as an appropriate tool for verification and documentation of mission critical system assurance, which could prove invaluable in the standardization and regulatory compliance effort of such systems. However, challenges still remain in showing the value of complex systems modeling and empowering domain experts with the required modeling tools and competencies. Hopefully, the contributions of this thesis can take part in overcoming these challenges and expand the use of collaborative modeling within academia and industry.
Additionally, it has been the ambition of the authors to illustrate that even though models are not completely accurate, they can still provide tangible value. This vision was shared by one of the forefathers of statistical modeling:

"All models are wrong, but some are useful."

– George E. P. Box [Box79]

References

[Alexis&11] Kostas Alexis, George Nikolakopoulos, Anthony Tzes. Switching model predictive attitude control for a quadrotor helicopter subject to atmospheric disturbances. *Control Engineering Practice*, (19):1195–1207, June 2011. 13 pages. [cited at p. 4, 13, 28, 29, 64, 90]

[Alexis&12] Kostas Alexis, George Nikolakopoulos, Anthony Tzes. Model predictive quadrotor control: attitude, altitude and position experimental studies. *Control Theory & Applications, IET*, 6(12):1812–1827, March 2012. 16 pages. [cited at p. 13, 16]

[Aswani&11] Aswani, A. and Gonzalez, H. and Shankar Sastry, S. and Tomlin, C. Provably Safe and Robust Learning-Based Model Predictive Control. *ArXiv e-prints*, July 2011. [cited at p. 13, 64]

[Aswani&12] Aswani, A. and Bouffard, P. and Tomlin, C. Extensions of learning-based model predictive control for real-time application to a quadrotor helicopter. In *American Control Conference (ACC), 2012*, pages 4661–4666, June 2012. [cited at p. 13, 64]

[Aswani&14] Anil Aswani, Patrick Bouffard, Xiaojing Zhang, and Claire Tomlin. Practical Comparison of Optimization Algorithms for Learning-Based MPC with Linear Models. 1–7, April 2014. 7 pages. [cited at p. 64]

[Back78] Back, Ralph-Johan. *On the Correctness of Refinement Steps in Program Development.* PhD thesis, Åbo Akademi, Department of Computer Science, Helsinki, Finland, 1978. Report A–1978–4. [cited at p. 53]

[Boom&05] Ton J.J. van den Boom, Ton C.P.M. Backx. *Model Predictive Control.* Delft University of Technology, 2004. Lecture Notes for the Dutch Institute of Systems and Control. [cited at p. 13]

[Bouabdallah&04] Samir Bouabdallah, Pierpaolo Murrieri and Roland Siegwart. Design and Control of an Indoor Micro Quadrotor. In *Proceedings of the 2004 IEEE International Conference on Robotics 8 Automation*, pages 4393–4398, IEEE, New Orleans, LA, April 2004. 6 pages. [cited at p. 9, 10, 13, 62]

[Bouabdallah07a] Samir Bouabdallah. *Design and Control of Quadrotors with Application to Autonomous Flying*. PhD thesis, University of Tlemcen, Algeria, 2007. 155 pages. [cited at p. 14]

[Bouabdallah&07b] Bouabdallah, S. and Siegwart, R. Full control of a quadrotor. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 153–158, Oct 2007. [cited at p. 62]

[Bouffard12] Bouffard, Patrick. *On-board Model Predictive Control of a Quadrotor Helicopter: Design, Implementation, and Experiments*. Master's thesis, EECS Department, University of California, Berkeley, Dec 2012. [cited at p. 13, 64]

[Box79] George E. P. Box. Robustness in the strategy of scientific model building. In Launer, R. L., Wilkinson, G. N., editor, *Robustness in Statistics*, pages 201–236, Academic Press. 36 pages. [cited at p. 65]

[Broenink97] Jan F. Broenink. Modelling, Simulation and Analysis with 20-Sim. *Journal A Special Issue CACSD*, 38(3):22–25, 1997. [cited at p. 17]

[Cai&10] Congzhao Cai, Hui Zhang, Jinhong Liu, Yongjun Gao. Modeling and Simulation of BLDC motor in Electric Power Steering. 1–4, March 2010. 4 pages. [cited at p. 27, 28, 87]

[Camacho&07] Eduardo F. Camacho and Carlos Bordons Alba. *Model Predictive Control. Advanced Textbooks in Control and Signal Processing*, Springer-Verlag London, 2007. 405 pages. [cited at p. 13]

[Carrillo&13] L.R. Garcia Carrillo et al. *Quad Rotorcraft Control*, chapter 2, pages 23–34. Springer-Verlag London, 2013. [cited at p. 10, 13, 62]

[Czyba&12] Roman Czyba, Grzegorz Szafranski. Control Structure Impact on the Flying Performance of the Multi-Rotor VTOL Platform - Design, Analysis and Experiemental Validation. *International Journal of Advanced Robotic Systems*, 1:1–9, June 2012. 9 pages. [cited at p. 9, 13, 27, 29, 62, 91]

[DESTECS09] DESTECS (Design Support and Tooling for Embedded Control Software). European Research Project, June 2009. `http://www.destecs.org`. [cited at p. 3]

[FEMA08] Federal Emergency Management Agency (FEMA). Student Manual: Heavy Equipment and Rigging Specialist Training - Module 2 unit 2: Calculating Weights and Center of Gravity. September 2008. 12 pages. [cited at p. 46]

[Fitzgerald&14] John Fitzgerald and Peter Gorm Larsen and Marcel Verhoef, editors. *Collaborative Design for Embedded Systems – Co-modelling and Co-simulation*. Springer, 2014. [cited at p. 5, 17, 18, 23, 34, 51, 54, 62]

[Flores15] Flores, Paulo. Euler Angles, Bryant Angles and Euler Parameters. In *Concepts and Formulations for Spatial Multibody Dynamics*, pages 15–22, Springer International Publishing, 2015. [cited at p. 11]

[Gao&95] Jinsong Gao, Kenneth W. Chase, Spencer P. Magleby. Comparison of Assembly Tolerance Analysis by the Direct Linearization and Modified Monte Carlo Simulation Methods. 1995. [cited at p. 54]

[Gooch05] John C. Gooch. The Dynamics and Challenges of Interdisciplinary Collaboration: A Case Study of "Cortical Depth of Bench" in Group Proposal Writing. In *IEEE Transactions on Professional Communication*, pages 177–190, June 2005. 13 pages. [cited at p. 2]

[Hamel&02] Tarek Hamel, Robert Mahony, Rogelio Lozano and James Ostrowski. Dynamic modeling and configuration stabilization for an x4 flyer. *15th Triennial World Congress*, 2002. 6 pages. [cited at p. 62]

[Hoffmann&07] Gabriel M. Hoffmann, Haomiao Huang, Steven L. Waslander and Claire J. Tomlin. Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment. In *Proc. of the AIAA Guidance, Navigation, and Control Conference*, AIAA, Hilton Head, South Carolina, August 2007. 20 pages. [cited at p. 10, 62]

[Hoffmann&11] Gabriel M. Hoffmann and Haomiao Huang and Steven L. Waslander and Claire J. Tomlin. Precision flight control for a multi-vehicle quadrotor helicopter testbed. *Control Engineering Practice*, 19(9):1023 – 1036, 2011. Special Section: DCDS'09 – The 2nd {IFAC} Workshop on Dependable Control of Discrete Systems. [cited at p. 62]

[Huang&09] Haomiao Huang, Gabriel M. Hoffmann, Steven L. Waslander and Claire J. Tomlin. Aerodynamics and Control of Autonomous Quadrotor Helicopters in Aggressive Maneuvering. *2009 IEEE International Conference on Robotics and Automation*, 3277–3282, May 2009. 6 pages. [cited at p. 62]

[INCOSEseh15] INCOSE. *Systems Engineering Handbook. A Guide for System Life Cycle Processes and Activities, Version 4.0.* Technical Report INCOSE-TP-2003-002-04, International Council on Systems Engineering (INCOSE), January 2015. [cited at p. 1]

[Jørgensen12] Peter W. V. Jørgensen. *Evaluation of Development Process for co-models*. Master's thesis, Aarhus University/Engineering College of Aarhus, December 2012. [cited at p. 4]

[Lausdahl&13a] Kenneth Lausdahl and Joey W. Coleman and Peter Gorm Larsen. *Semantics of the VDM Real-Time Dialect*. Technical Report ECE-TR-13, Aarhus University, April 2013. [cited at p. 17]

[Lee&10] Taeyoung Lee, Melvin Leok, and N. Harris McClamroch. Geometric Tracking Control of a Quadrotor UAV on SE(3). *49th IEEE Conference on Decision and Control*, 5420–5425, December 2010. 6 pages. [cited at p. 62]

[Lopes&12] Renato Vilela Lopes, Pedro Henrique de Rodrigues Quernel e Assis Santana, Geovany Araujo Borges, Joao Yoshiyuki Ishihara. Model Predictive Control applied to tracking and attitude stabilization of a VTOL quadrotor aircraft. In *ABCM Symposium Series in Mechatronics*, pages 176–185, ABCM, 2012. 10 pages. [cited at p. 13, 16]

[Maciejowski02] Jan Marian Maciejowski. *Predictive Control with Constraints*. Pearson Education Limited, 2002. [cited at p. 13, 16]

[Mahony&12] Robert Mahony and Vijay Kumar and Peter Corke. Multirotor Arial Vehicles: Modeling, Estimation and Control of Quadrotor. *IEEE Robotics & Automation Magazine*, 20–33, September 2012. [cited at p. 10, 11, 28, 29, 62, 90]

[Meier&12] Lorenz Meier, Petri Tanskanen, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, Marc Pollefeys. PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Autonomous Robots*, 33(1):21–39, August 2012. 19 pages. [cited at p. 19]

[Moon07] Francis C. Moon. *The Machines of Leonardo da Vinci and Franz Reuleaux: Kinematics of Machines from the Renaissance to the 20th Century*. Volume 2, Springer Netherlands, Het Spoor 2, 3994 AK Houten, Netherlands, 1 edition, 2007. 419 pages. [cited at p. 25]

[Needham65] Joseph Needham. *Science and Civilisation in China: Volume 4, Physics and Physical Technology, Part 2, Mechanical Engineering*. Cambridge University Press, Shaftesbury Road, Cambridge, United Kingdom, 1965. 816 pages. [cited at p. 25]

[Nidhra&12] Srinivas Nidhra and Jagruthi Dondeti. Black Box and White Box Testing Techniques –A Litteature Review. *International Journal of Embedded Systems and Applications (IJESA)*, 2(2):29–50, June 2012. 22 pages. [cited at p. 28]

[NIST15a] NIST, Cyber Physical Systems Public Working Group. *Draft Framework for Cyber-Physical Systems*. Technical Report Draft Release 0.8, National Institute of Standardards and Technology, September 2015. [cited at p. 2]

[Orsag&12] Matko Orsag, Stjepan Bogdan. Influence of Forward and Descent Flight on Quadrotor Dynamics. In Dr. Ramesh K. Agarwal, editor, *Recent Advances in Aircraft Technology*, chapter 7, pages 141–156, InTech, University Campus STeP Ri, Slavka Krautzeka 83/A, 51000 Rijeka, Croatia, February 2012. 16 pages. [cited at p. 62]

[Overture07] Overture-Core-Team. Overture Web site. http://www.overturetool.org, 2007. [cited at p. 17]

[Raffo&08] Guilherme V. Raffo, Manuel G. Ortega, Francisco R. Rubio. MPC with Nonlinear H-infinity Control for Path Tracking of a Quad-Rotor Helicopter. pages 8564–8569, The International Federation of Automatic Control, Seoul, Korea, July 2008. 6 pages. [cited at p. 13]

[Richalet&78] Richalet, J. and Rault, A. and Testud, J. L. and Papon, J. Model Predictive Heuristic Control. *Automatica*, 5(14):413–428, September 1978. 16 pages. [cited at p. 13]

[Schluter09] Schlüter, Martin and Egea, Jose A and Banga, Julio R. Extended ant colony optimization for non-convex mixed integer nonlinear programming. *Computers & Operations Research*, 36(7):2217–2229, 2009. [cited at p. 37]

[Tayebi&09] A. Tayebi and S. McGilvray. Attitude stabilization of a four-rotor aerial robot. *43rd IEEE Conference on Decision and Control*, 1216–1221, December 2004. 6 pages. [cited at p. 62]

[Teich01] Teich, Jürgen. Pareto-front exploration with uncertain objectives. In *Evolutionary multi-criterion optimization*, pages 314–328, Springer, 2001. [cited at p. 63]

[Tempo&11] Roberto Tempo and Francho Blanchini. *Control System Advanced Methods*, chapter 7. CRC Press, Taylor & Fancis Group, 6000 Broken Sound Parkway NW, Suite 300, 2 edition, 2011. 18 pages. [cited at p. 28, 87]

[Thompson13] Haydn Thompson, editor. *Cyber-Physical Systems: Uplifting Europe's Innovation Capacity*. European Commission Unit A3 - DG CONNECT, December 2013. [cited at p. 1]

[Toliyat&04] Hamid A. Toliyat, Gerald B. Kliman. *Handbook of Electrical Motors*. Marcel Dekker, Inc, 270 Madison Avenue, New York, NY 10016, U.S.A, 2 edition, April 2004. 850 pages. [cited at p. 27, 28, 87]

[Virgala&13] Ivan Virgala, Peter Frankovsky, Maria Kenderova. Friction Effect Analysis of a DC Motor. *American Journal of Mechanical Engineering*, 1(1):1–5, 2013. 5 pages. [cited at p. 28, 87]

Appendices

Appendix A

Terminology, Abbreviations and Symbols

This appendix provides complete lists of terminology, abbreviations and symbols. The symbols are grouped into *scalars*, *vectors* and *matrices*.

A.1. Terminology

Table A.1: Terminology

multi-domain	Consisting of components from multiple domains
model fidelity	How accurately a computer-based model exhibits the behavior of the modeled system
modeling goal	Overall purpose of the modeling effort, common for all model present in the method used
controller	A controller steers a physical system by regulating the actuators based on reading from the sensors and a control algorithm.
state	The state of a system refers to the condition of relevant system properties, such as position, velocity etc.
co-model	A collaborative model consisting of a DE model of the digital controller and a CT model of the physical system dynamics
co-simulation	Simulation of a co-model. Information, shared design parameters and events can be exchanged between the DE and CT models.
contract	The contract specifies the information shared between the DE and CT models of a co-model, including monitored and controlled parameters, shared design parameters and events.

block	Element of a model
flight mode	The fight mode provides additional assistance to the pilot. Several flight modes are defined in the APM:Copter autopilot, including Stabilize, Altitude Hold, Position Hold etc.
script	A script is, in this settings, an input file to a co-simulation. The script can contain input to the co-simulation, such as pilot input, or modify model parameters and values during a co-simulation.
telemetry control	The pilot of an UAV often uses a hand-held remote control to steer the UAV. Telemetry control refers to the wireless communication hardware and protocols used between the remote control and the UAV.

A.2. Abbreviations

Table A.3: List of abbreviations

CPS	Cyber-Physical System
CT	Continuous Time
DE	Discrete Event
UAV	Unmanned Aerial Vehicle
MPC	Model Predictive Control
PID	Proportional Integral Derivative
DESTECS	Design Support and Tooling for Embedded Control Software
DAS	Danish Aviation Systems
LBMPC	Learning Based Model Predictive Control
ABS	Anti-lock Braking System
MIMO	Multiple-Input Multiple-Output
QP	Quadratic Programming
DLL	Dynamic Link Library
ESC	Electronic Speed Controller
PWM	Pulse Width Modulation

SysML	Systems Modeling Language
BDD	Block Definition Diagram
IBD	Internal Block definition Diagram
BLDC	Brushless Direct Current
AHRS	Attitude Heading Reference Systems
JNI	Java Native Interface
LOC	Lines Of Code
DSE	Design Space Exploration
CG	Center of Gravity
ACA	Automated Co-model Analysis
EC	Evaluation Categories
SMPC	Switching Model Predictive Control
SIDUR	Safety Innovations for Drones/UAVs enhancing the Reliability

A.3. Symbols

Table A.5: List of scalars

$\psi_{\mathbb{b}}$	Angular position, describing the rotation about the $\mathbf{z}_{\mathbb{b}}$ axis
$\theta_{\mathbb{b}}$	Angular position, describing the rotation about the $\mathbf{y}_{\mathbb{b}}$ axis
$\phi_{\mathbb{b}}$	Angular position, describing the rotation about the $\mathbf{x}_{\mathbb{b}}$ axis
$x_{\mathbb{w}}$	Position
$y_{\mathbb{w}}$	Position
$z_{\mathbb{w}}$	Position
R_d	Rotor aerodynamic drag coefficient
d	Arm length of the Quadrotor (Half wingspan)
m	Mass of the Quadrotor

g	Gravitational constant
I_{xx}	Quadrotor inertia about the $\mathbf{x}_{\mathbb{b}}$ axis
I_{yy}	Quadrotor inertia about the $\mathbf{y}_{\mathbb{b}}$ axis
I_{zz}	Quadrotor inertia about the $\mathbf{z}_{\mathbb{b}}$ axis
N	Prediction horizon
M	Control horizon
p	Number of control inputs
q	Number of plants outputs
F_s	Sample rate
T_s	Sample time
ω_m	Motor angular speed
i_m	Motor current
u_m	Motor voltage
A_m	Motor coefficient
B_m	Motor coefficient
s_m	Motor speed setpoint
T_m	Motor torque
R_m	Motor phase resistance
L_m	Motor phase inductance
I_m	Motor rotor inertia
M_{lo}	Motor losses coefficient
R_l	Rotor lift coefficient
R_t	Reaction torque coefficient
I_r	Rotor inertia

F_{lift}	Rotor lift force
ω_r	Rotor angular speed
T_r	Rotor reaction torque
D_r	Rotor aerodynamic drag
A_{esc}	ESC transfer function parameter
B_{esc}	ESC transfer function parameter
C_{esc}	ESC transfer function parameter
D_{esc}	ESC transfer function parameter

Table A.6: List of vectors

$\boldsymbol{\xi}_{\mathbb{W}} \in \mathbb{R}^3$	Position of the quadrotor
$\dot{\boldsymbol{\xi}}_{\mathbb{W}} \in \mathbb{R}^3$	Velocity of the quadrotor
$\ddot{\boldsymbol{\xi}}_{\mathbb{W}} \in \mathbb{R}^3$	Acceleration of the quadrotor
$\boldsymbol{\eta}_{\mathbb{b}} = [\phi_{\mathbb{b}}, \theta_{\mathbb{b}}, \psi_{\mathbb{b}}]^T \in \mathbb{R}^3$	Angular position of the quadrotor
$\boldsymbol{\omega}_{\mathbb{b}} = \dot{\boldsymbol{\eta}}_{\mathbb{b}} \in \mathbb{R}^3$	Angular velocity of the quadrotor
$\dot{\boldsymbol{\omega}}_{\mathbb{b}} \in \mathbb{R}^3$	Angular acceleration of the quadrotor
$\boldsymbol{\tau}_{\mathbb{b}} \in \mathbb{R}^3$	Angular moments of the quadrotor
$\boldsymbol{\omega}_r \in \mathbb{R}^4$	Angular velocity of the rotors
$\mathbf{z}_{\mathbb{W}} \in \mathbb{R}^3$	World reference frame, down axis
$\mathbf{y}_{\mathbb{W}} \in \mathbb{R}^3$	World reference frame, east axis
$\mathbf{x}_{\mathbb{W}} \in \mathbb{R}^3$	World reference frame, north axis
$\mathbf{z}_{\mathbb{b}} \in \mathbb{R}^3$	Body fixed frame, down axis
$\mathbf{y}_{\mathbb{b}} \in \mathbb{R}^3$	Body fixed frame, right axis
$\mathbf{x}_{\mathbb{b}} \in \mathbb{R}^3$	Body fixed frame, forward axis
$\mathbf{f}_{\mathbb{b}} \in \mathbb{R}^3$	Total forces affecting the quadrotor, including lift forces

$\boldsymbol{\alpha} \in \mathbb{R}^4$	Angle between $\mathbf{x}_{\mathbb{b}}$ axis and rotors
$\mathbf{x}_s \in \mathbb{R}^6$	State vector including orientation and derivatives of these
$\dot{\mathbf{x}}_s \in \mathbb{R}^6$	Derivative of the state vector $\mathbf{x}_s$
$\mathbf{x}_d \in \mathbb{R}^6$	Discrete state vector including orientation and derivatives of these
$\mathbf{u} \in \mathbb{R}^p$	Control inputs
$\hat{\mathbf{u}} \in \mathbb{R}^p$	Predicted control inputs
$\mathbf{u}^* \in \mathbb{R}^p$	Optimal control inputs
$\hat{\mathbf{U}} \in \mathbb{R}^{p*M}$	Sequence of M predicted control inputs $\hat{\mathbf{u}}$
$\mathbf{y}_s \in \mathbb{R}^q$	Plant output
$\mathbf{y}_d \in \mathbb{R}^q$	Discrete plant output
$\hat{\mathbf{y}}_d \in \mathbb{R}^q$	Predicted plant output
$\mathbf{r} \in \mathbb{R}^q$	Reference trajectory
$\boldsymbol{\mu} \in \mathbb{R}^q$	Weights on plant tracking error
$\boldsymbol{\rho} \in \mathbb{R}^p$	Weights on control variations

Table A.7: List of matrices

$\mathbf{R} \in \mathbb{R}^{3\times3}$	Rotation matrix
$\mathbf{I} \in \mathbb{R}^{3\times3}$	The inertia matrix of the quadrotor
$\mathbf{A} \in \mathbb{R}^{6\times6}$	General state space matrix
$\mathbf{B} \in \mathbb{R}^{6\times3}$	General state space matrix
$\mathbf{C} \in \mathbb{R}^{3\times6}$	General state space matrix
$\mathbf{A}_d \in \mathbb{R}^{6\times6}$	Discrete-time state space matrix
$\mathbf{B}_d \in \mathbb{R}^{6\times3}$	Discrete-time state space matrix
$\mathbf{C}_d \in \mathbb{R}^{3\times6}$	Discrete-time state space matrix

Appendix B

MPC Calculations

This appendix describes the linearization and discretization of the attitude state space model of the quadrotor, as presented in section 2.4.1. It is assumed that the reader has already read chapter 2 and is familiar with the notation. A complete list of symbols is available in appendix A.

B.1. Linearization

The linearization of the state space model is performed with a first-order Taylor expansion, under the assumption that the inertia matrix $\mathbf{I} \in \mathbb{R}^{3\times3}$ is diagonal.
Linearization using a first order Taylor expansion for a n-variable function has the general expression:

$$f(x_1, \cdots, x_n) \approx f(x_1^0, \cdots, x_n^0) + \frac{\partial f(x_1^0, \cdots, x_n^0)}{\partial x_1}(x_1 - x_1^0) + \cdots + \frac{\partial f(x_1^0, \cdots, x_n^0)}{\partial x_n}(x_n - x_n^0) \tag{B.1}$$

where x^0 denotes the linearization point for x.
In the linearization of the attitude state space system the following variables are defined:

$$\dot{\phi_{\mathbb{b}}} = \dot{\phi_{\mathbb{b}}}^0 + \delta\dot{\phi_{\mathbb{b}}} \tag{B.2}$$

$$\dot{\theta_{\mathbb{b}}} = \dot{\theta_{\mathbb{b}}}^0 + \delta\dot{\theta_{\mathbb{b}}} \tag{B.3}$$

$$\dot{\psi_{\mathbb{b}}} = \dot{\psi_{\mathbb{b}}}^0 + \delta\dot{\psi_{\mathbb{b}}} \tag{B.4}$$

$$\dot{\mathbf{u}} = \dot{\mathbf{u}}^0 + \delta\dot{\mathbf{u}} \in \mathbb{R}^3 \tag{B.5}$$

The nonlinear model of the roll motion can be described by equation B.6:

$$\ddot{\phi_{\mathbb{b}}}(\dot{\theta_{\mathbb{b}}}, \dot{\psi_{\mathbb{b}}}, \mathbf{u}_1) = \dot{\theta_{\mathbb{b}}}\dot{\psi_{\mathbb{b}}}\frac{I_{yy} - I_{zz}}{I_{xx}} + \frac{d}{I_{xx}}\mathbf{u}_1 \tag{B.6}$$

Using the Taylor expansion defined in equation B.1 and the defined variables in equations B.2 to B.5 gives the following linear expression for the roll motion:

$$\ddot{\phi_{\mathbb{b}}} = \dot{\psi_{\mathbb{b}}}^0\frac{I_{yy} - I_{zz}}{I_{xx}}\delta\dot{\theta_{\mathbb{b}}} + \dot{\theta_{\mathbb{b}}}^0\frac{I_{yy} - I_{zz}}{I_{xx}}\delta\dot{\psi_{\mathbb{b}}} + \frac{d}{I_{xx}}\delta\mathbf{u}_1 \tag{B.7}$$

The system will be linearized about the main operating point, a hovering state, given by the following points:

$$\boldsymbol{\omega}_{\mathbb{b}}{}^{0} = \begin{bmatrix} \dot{\phi}_{\mathbb{b}}{}^{0} \\ \dot{\theta}_{\mathbb{b}}{}^{0} \\ \dot{\psi}_{\mathbb{b}}{}^{0} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \mathbf{u}^{0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{B.8}$$

Inserting the linearization point from B.8 into equation B.7 gives the following:

$$\ddot{\phi}_{\mathbb{b}} = \frac{d}{I_{xx}} \delta \mathbf{u}_1 \tag{B.9}$$

The same method can be applied to pitch and yaw motion, resulting in the following linearized equations:

$$\ddot{\theta}_{\mathbb{b}} = \frac{d}{I_{yy}} \delta \mathbf{u}_2 \tag{B.10}$$

$$\ddot{\psi}_{\mathbb{b}} = \frac{1}{I_{zz}} \delta \mathbf{u}_3 \tag{B.11}$$

The linearized equations for the angular accelerations can be written in the general continuous time state space model:

$$\dot{\mathbf{x}}_s = \begin{bmatrix} \delta\dot{\phi}_{\mathbb{b}} \\ \delta\ddot{\phi}_{\mathbb{b}} \\ \delta\dot{\theta}_{\mathbb{b}} \\ \delta\ddot{\theta}_{\mathbb{b}} \\ \delta\dot{\psi}_{\mathbb{b}} \\ \delta\ddot{\psi}_{\mathbb{b}} \end{bmatrix} = \mathbf{A}\mathbf{x}_s + \mathbf{B}\delta\mathbf{u} \tag{B.12}$$

$$\mathbf{y}_s = \mathbf{C}\mathbf{x}_s \tag{B.13}$$

where $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ is given by

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & \\ \frac{d}{I_{xx}} & 0 & 0 & \\ 0 & 0 & 0 & 0 \\ 0 & \frac{d}{I_{yy}} & 0 & \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{I_{zz}} & \end{bmatrix} \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{B.14}$$

B.2. Discretization

Using a zero-order hold on the inputs and a sampling rate $F_s = 1/T_s$, the continuous time state space model can be discretized into the general discrete state space model:

$$\mathbf{x}_d(k+1) = \mathbf{A}_d\mathbf{x}_d(k) + \mathbf{B}_d\delta\mathbf{u}(k) \tag{B.15}$$

$$\mathbf{y}_d(k) = \mathbf{C}_d\mathbf{x}_d(k) \tag{B.16}$$

where $\mathbf{A}_d$ and $\mathbf{B}_d$ are given by

$$\mathbf{A}_d = e^{\mathbf{A}T_s} = \mathcal{L}^{-1}\{(s\mathbf{I}^{6\times 6} - \mathbf{A})^{-1}\}_{t=T_s} \tag{B.17}$$

$$\mathbf{B}_d = (\int_0^{T_s} e^{\mathbf{A}\tau} d\tau)\mathbf{B} \tag{B.18}$$

Note that $\mathbf{I}^{6\times 6}$ is an identity matrix and not the inertia matrix. $\mathcal{L}^{-1}$ is the inverse Laplace transformation and s is the associated complex variable.
Assuming a sampling rate of 100 Hz, an arm length of 0.226 m and using the moments of inertia presented in table 3.4 gives the following state space matrices:

$$\mathbf{A}_d = \begin{bmatrix} 1 & \frac{1}{T_s} & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \frac{1}{T_s} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \frac{1}{T_s} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0.01 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0.01 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.19}$$

$$\mathbf{B}_d = \begin{bmatrix} 0.0001 & 0 & 0 \\ 0.1963 & 0 & 0 \\ 0 & 0.0001 & 0 \\ 0 & 0.0618 & 0 \\ 0 & 0 & 0.0002 \\ 0 & 0 & 0.3439 \end{bmatrix} \tag{B.20}$$

$$\mathbf{C}_d = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{B.21}$$

Appendix C

Propulsion System Refinement Fittings

This appendix presents the mathematical derivations of the refined propulsion system parameters, as described and presented in table 3.3. The propulsion system parameters include a transfer function for the Electronic Speed Controller (ESC) and various motor and rotor parameters as shown in figure C.1. The ESC, motor and rotor parameters are described in the three following subsections.

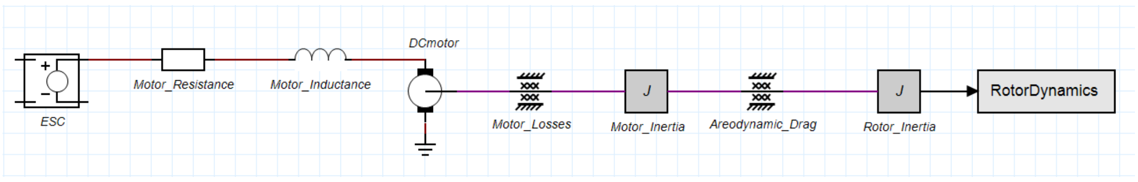


Figure C.1: Propulsion system 20-sim model

C.1. Electronic speed controller parameters

The purpose of the ESCs are to control the motor speed, based on a digital setpoint from the controller. The motor speed is controlled in an open-loop fashion, where it is assumed that the voltage applied to the motor is proportional to the motor speed.
The ESCs are modeled as a simple transfer function transforming the motor setpoint to a voltage. This simple approach is used, first of all because the hardware and software implementation of the ESCs are unidentified and secondly because a transfer function gives a good model fidelity with a limited workload. It should be noted that the transfer function only incorporates the static performance of the ESCs and no dynamic behavior e.g. how fast the ESCs can change the output voltage.
In order to determine the transfer function a test was performed, measuring the motor voltage at different motor setpoints. Figure C.2 shows the measured data along with the transfer function obtained through an exponential regression. The exponential regression was chosen based on the appearance of the test data and obtained through Matlab's Curve Fitting Tool:

$$u_m = A_{esc} e^{B_{esc} s_m} + C_{esc} e^{D_{esc} s_m} \tag{C.1}$$

where A_{esc}, B_{esc}, C_{esc} and D_{esc} are transfer function parameters, u_m is the voltage applied to the motor and s_m is the motor setpoint.

The values of the ESC transfer function parameters are presented in table C.1.

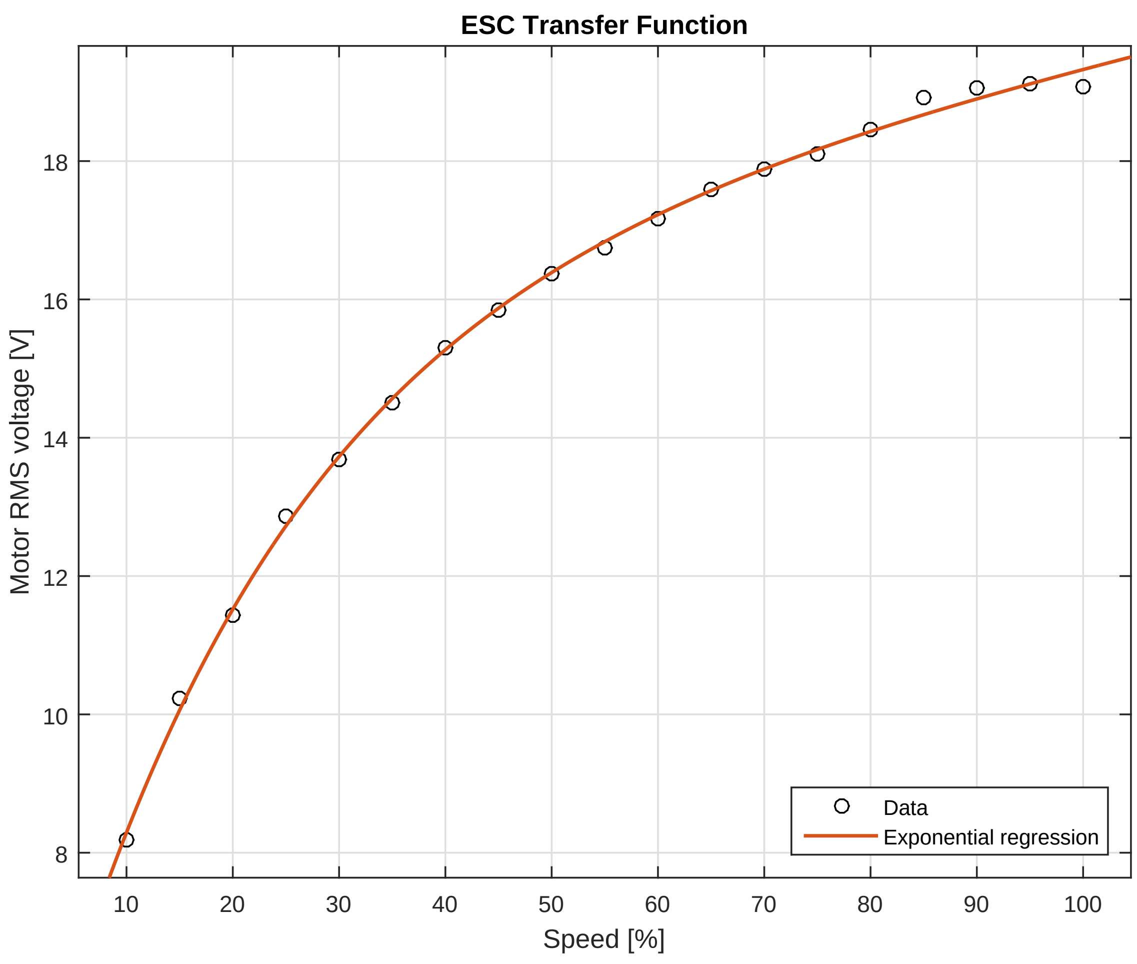


Figure C.2: ESC measurements and fitted transfer function

Table C.1: ESC transfer function parameters

Parameter	**Value**
A_{esc}	16.45
B_{esc}	0.001704
C_{esc}	-12.95
D_{esc}	-0.04282

C.2. Motor parameters

A Brushless Direct Current (BLDC) motor is a complex system, which can be described at various abstraction levels, from a complete description of magnetic fields and a three phased electric circuit to an ideal linear transfer function. An intermediate abstraction level is chosen in this thesis, in order to simplify the modeling as much as possible, while still capturing both the static and dynamic behavior of the motor. The motor is modeled as a single phased DC motor, where the original three phases and the Pulse Width Modulation (PWM) signals are all transformed into DC-equivalent values. Five properties of the motor is included in the model:

- Motor resistance
- Motor inductance
- Motor inertia
- Motor transfer function
- Motor losses

The motor resistance, inductance and inertia are found through simple measurements, as described in table 3.3.
The motor transfer function of an ideal DC motor can be considered strictly linear [Cai&10], but the measurements and [Tempo&11] suggest that an affine relationship is more accurate:

$$\omega_m = A_m u_m + B_m \tag{C.2}$$

where ω_m is the motor speed, A_m is the ideal motor constant, u_m is the voltage applied to the motor and B_m is the constant that makes the function affine.
Measurements at different speeds were performed using the setup of figure 3.7a, collecting data of both motor voltage and speed. The motor speed was determined by measuring the frequency of the cross-phase sinusoidal signal. The measurements were subsequently fitted with a linear and an affine regression, presented in figure C.3. The strictly linear regression is only presented to show that an affine function is a better fit.

Motor loss is considered equal to the torque required to turn the motor with no load [Toliyat&04] and is assumed linearly proportional to the motor speed [Virgala&13]:

$$T_m = M_{lo}\omega_m \tag{C.3}$$

where T_m is motor torque, M_{lo} is motor loss coefficient and ω_m is motor speed. Additionally, the relationship between motor torque and motor current is assumed linear:

$$T_m = A_m i_m \tag{C.4}$$

where A_m is the motor constant and i_m is the motor current.
Given the proportionality between motor torque and current, measurements of the current at different speeds provide the torque load on the motor. This is plotted and fitted to obtain the motor loss coefficient, shown in figure C.4.
The complete list of motor parameters are presented in table C.2.

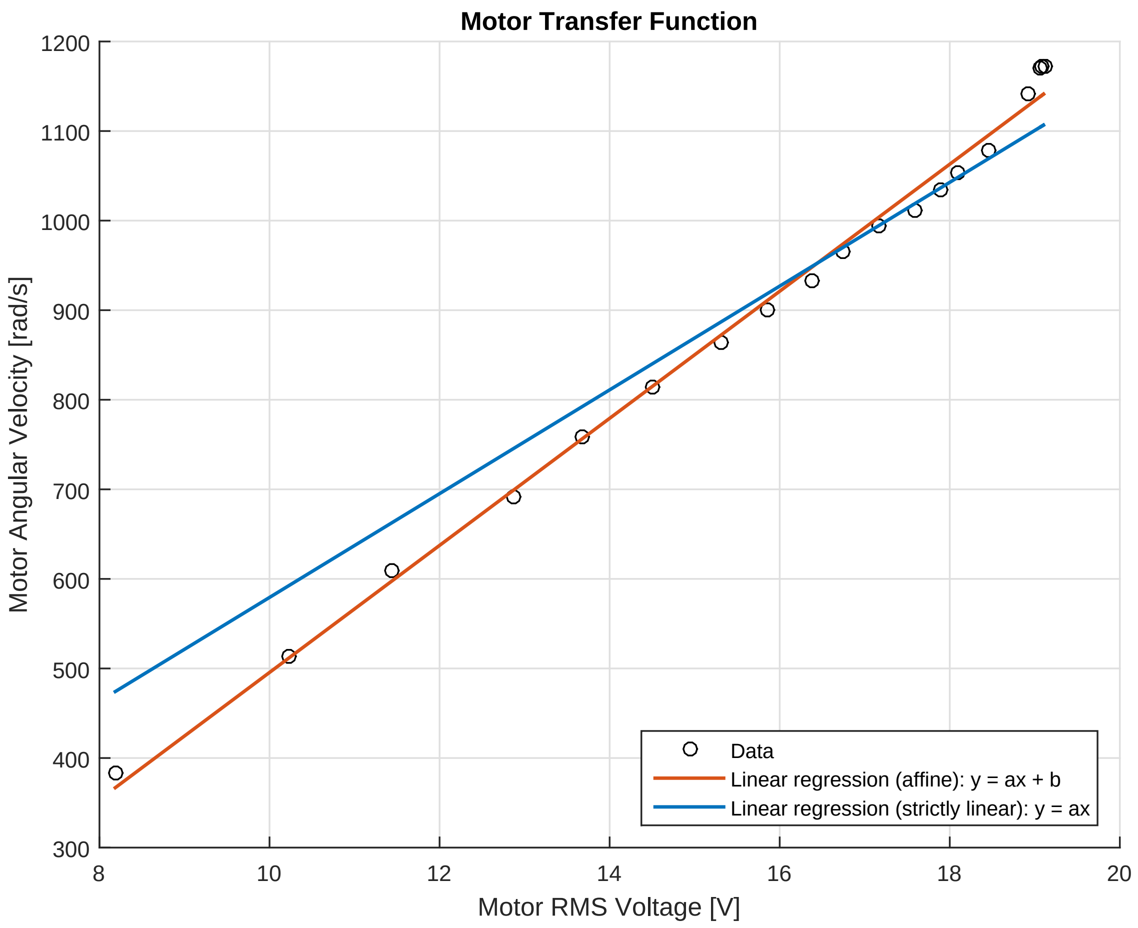


Figure C.3: Motor voltage and speed measurements and linear regressions used for obtained motor constants.

Table C.2: Motor parameters

Parameter	**Description**	**Value**	**Unit**
R_m	Motor resistance	0.12700	Ω
L_m	Motor inductance	1.6968E-4	H
I_m	Motor rotor inertia	3.3800E-7	$\mathrm{kg\,m^2}$
M_{lo}	Motor loss coefficient	4.4528E-6	N m s
A_m	Motor coefficient	70.930	-
B_m	Motor coefficient	-213.76	-

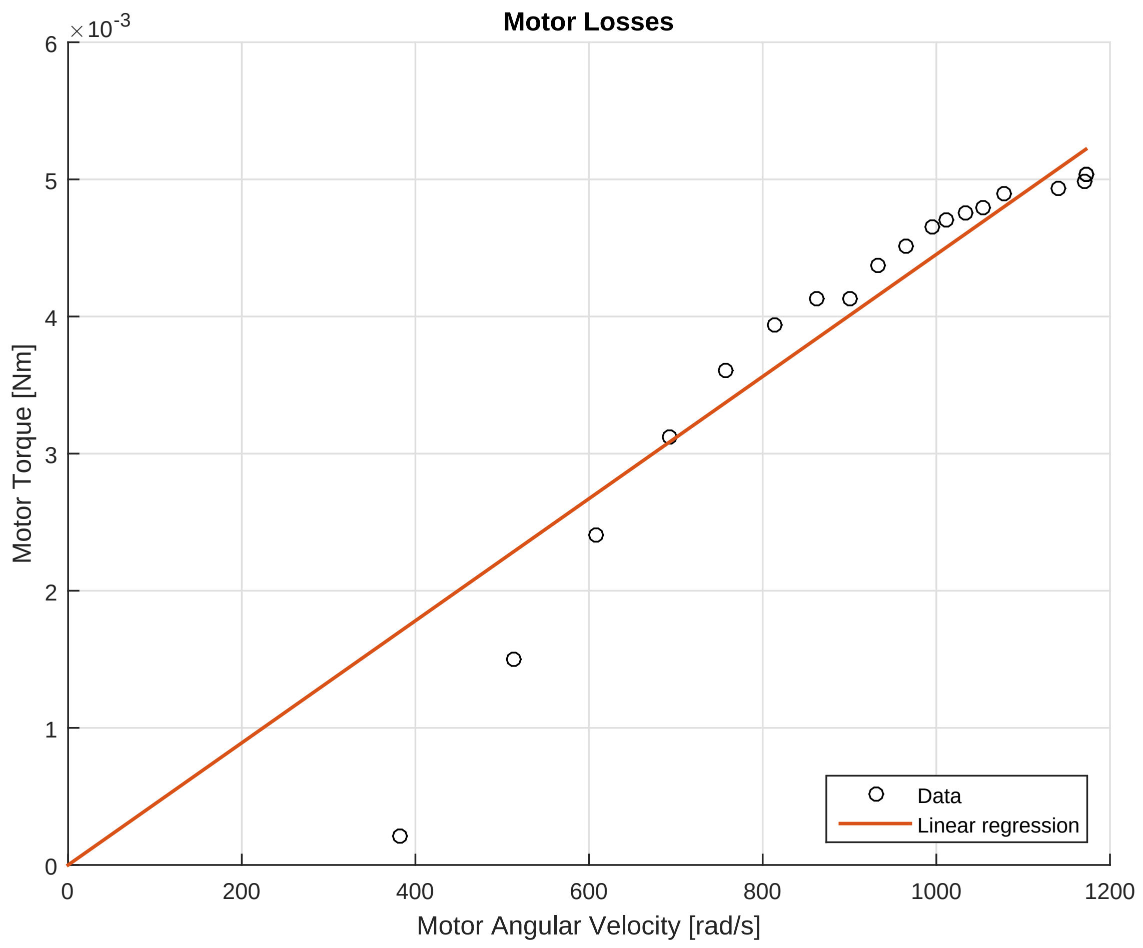


Figure C.4: Motor torque and speed measurements with no load and linear regression, providing the motor losses coefficient

C.3. Rotor parameters

The static and dynamic behavior of a rotor can be described by four distinct properties, assuming a rigid rotor:

- Lift force
- Reaction torque
- Aerodynamic drag
- Inertia

The inertia of the rotor was approximated using SolidWorks. An exact 3D model of the rotor was not available. Therefore the inertia was approximated using an similar 3D rotor model, which was modified with the correct mass, blade length and hub diameter. The curvature and thickness of the rotor were not taken into account. The rotor inertia along with the rest of the rotor coefficients are presented in table C.3.

The lift force is assumed to be proportional to the square of the rotor speed [Mahony&12, Alexis&11]:

$$F_{lift} = R_l {\omega_r}^2 \tag{C.5}$$

where F_{lift} is the lift force of the rotor, R_l is the lift coefficient and ω_r is the rotor angular speed. An experiment to measure the lift force at different rotor speeds was performed using the setup shown in figure 3.7a. The measurement data and associated quadratic regression are presented in figure C.5. The rotor lift coefficient is obtained through the quadratic regression and is presented in table C.3.

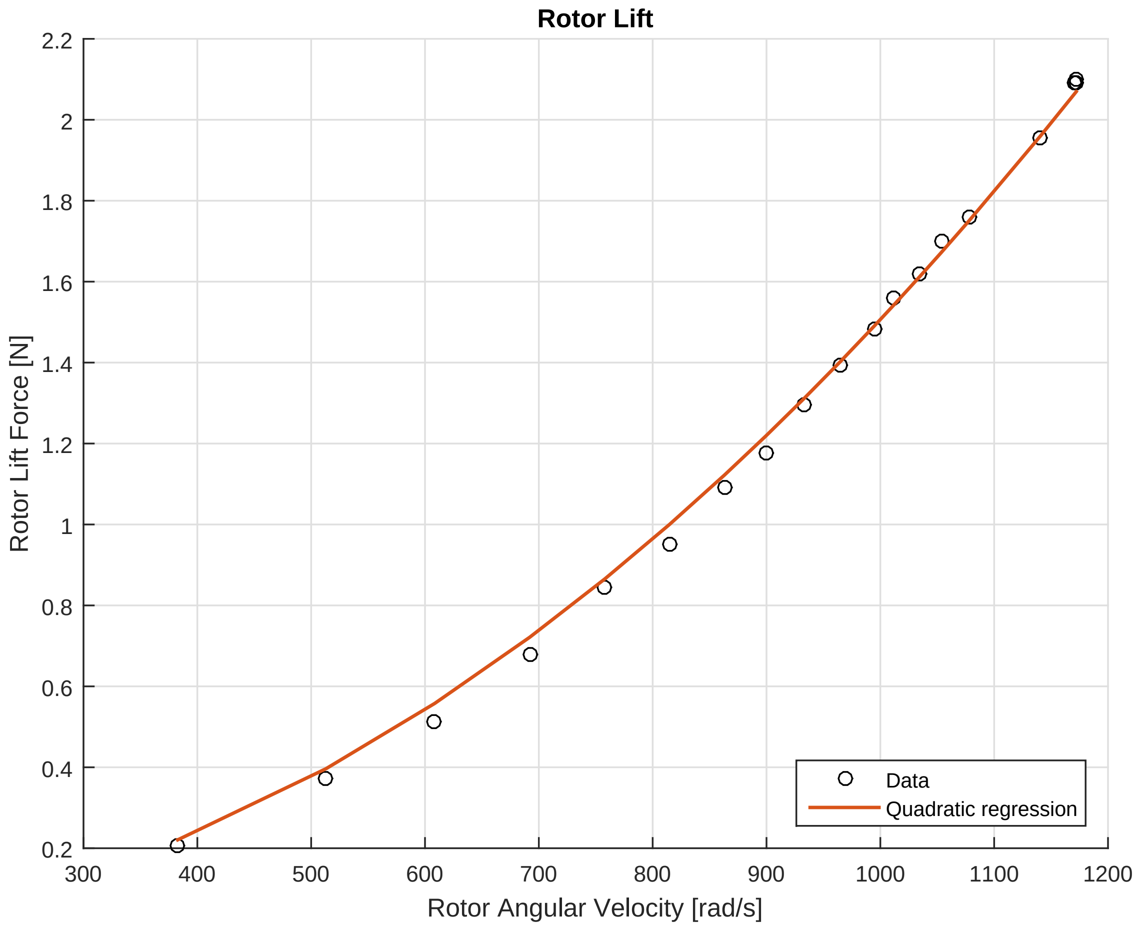


Figure C.5: Rotor measurements and quadratic regression leading to the rotor lift coefficient

The reaction torque describes the torque created in the opposite direction of the rotor rotation, as a result of Newtons second law, and is assumed to be proportional to the square of the rotor speed [Mahony&12, Alexis&11]:

$$T_r = R_t {\omega_r}^2 \tag{C.6}$$

where T_r is the reaction torque, R_t is the reaction torque coefficient and ω_r is the angular rotor speed. The combined reaction torque of all four rotors provide the quadrotor with the ability to perform a rotation in yaw. A measurement of the reaction torque at different speeds was performed using the setup shown in figure 3.7b. Figure C.6 presents the measurement data and the quadratic regression, from which the reaction torque coefficient is obtained.

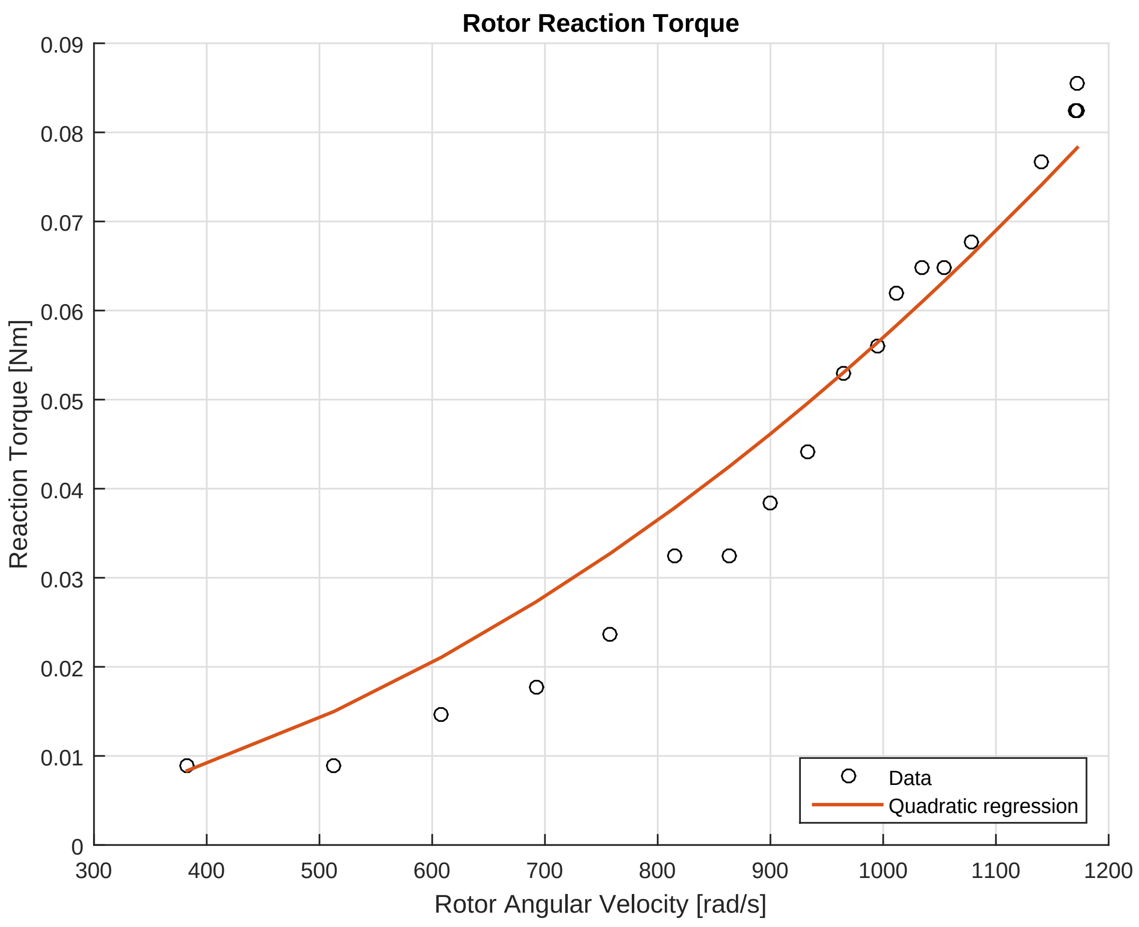


Figure C.6: Rotor measurements and quadratic regression leading to the rotor reaction torque coefficient

The aerodynamic drag is a measure of the torque counteracting the motor rotation due to displacement of air produced by the rotor. The aerodynamic drag is assumed to be proportional to the square of the rotor speed [Czyba&12]:

$$D_r = R_d {\omega_r}^2 \tag{C.7}$$

where D_r is the aerodynamic drag and R_d is the aerodynamic drag coefficient. The drag is measured as the load difference on the motor with and without the rotor attached, at different speeds, using the setup in figure 3.7a. The drag measurements and a quadratic regression are presented in figure C.7

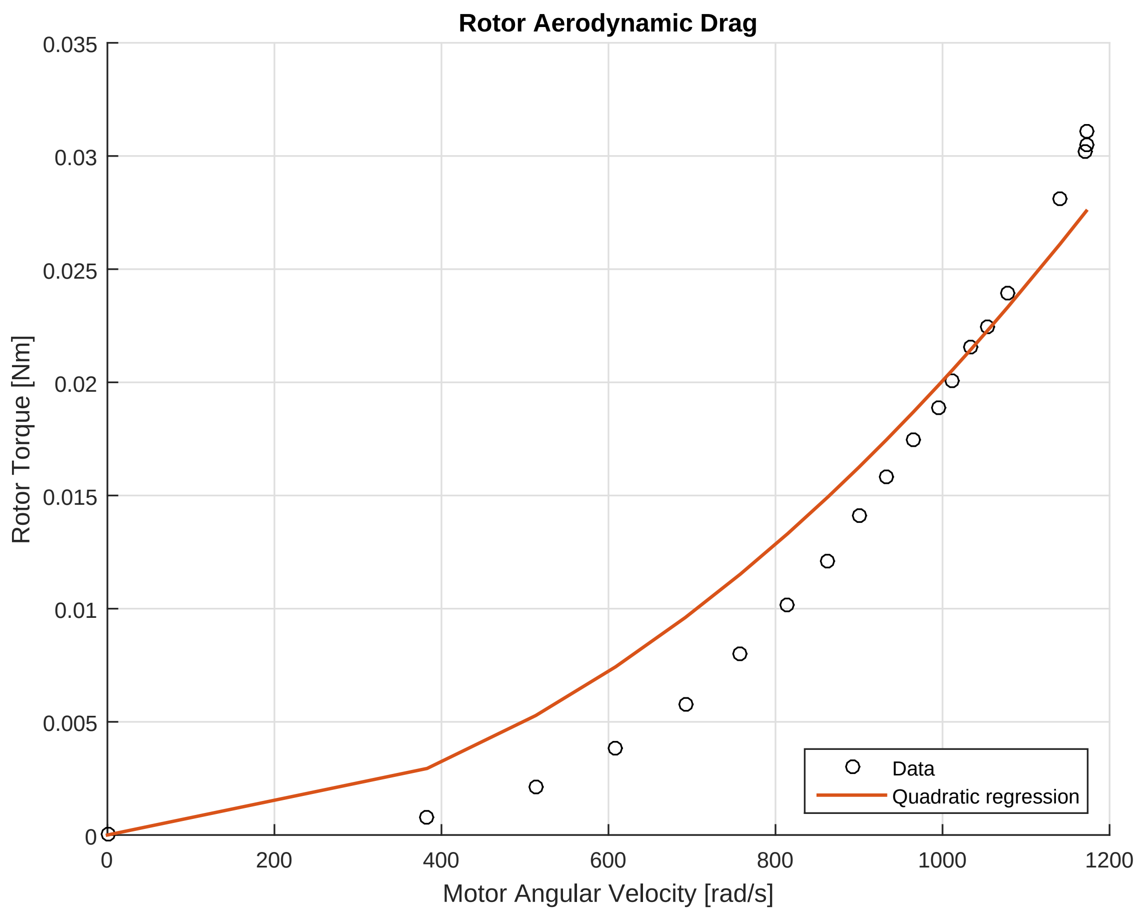


Figure C.7: Rotor measurements and quadratic regression leading to the rotor aerodynamic drag coefficient

Table C.3: Rotor parameters

Parameter	**Description**	**Value**	**Unit**
R_l	Rotor lift coefficient	1.5064E-6	$\mathrm{N\,s^2}$
R_t	Rotor reaction torque coefficient	5.6977E-8	$\mathrm{N\,m\,s^2}$
R_d	Rotor aerodynamic drag coefficient	2.0066E-8	$\mathrm{N\,m\,s^2}$
I_r	Rotor inertia	2.800E-6	$\mathrm{kg\,m^2}$

Ivan Grujic and René Nilsson, Model-Based Development and Evaluation of Control for Complex Multi-Domain Systems: Attitude Control for a Quadrotor UAV, 2016

Department of Engineering
Aarhus University
Inge Lehmanns Gade 10
8000 Aarhus
Denmark

Tel.: +45 8715 0000