Faculty of Aerospace Engineering

Differential Dynamic Programming for Aerial Robots

using an Aerodynamics Model

N.O. Abuter Grebe

June 5, 2017

TU Delft Delft University of Technology

Challenge the future

Differential Dynamic Programming for Aerial Robots

using an Aerodynamics Model

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering
at Delft University of Technology

N.O. Abuter Grebe

June 5, 2017

Faculty of Aerospace Engineering · Delft University of Technology

Delft University of Technology

Delft University Of Technology
Department Of
Control and Simulation

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance a thesis entitled **"Differential Dynamic Programming for Aerial Robots "** by **N.O. Abuter Grebe** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: June 5, 2017

Readers:

S. Stoneman, DLR

Ir. C. de Wagter, TU Delft

Ir. J.A. Melkert, TU Delft

Prof. dr. ir. M. Mulder, TU Delft

Acknowledgment

At this point, I want to thank everyone who directly or indirectly supported me in my thesis work.

Most importantly I want to thank both my daily supervisors Samantha Stoneman and Teodor Tomić from the German Aerospace Center (DLR) for all their guidance, support and energy throughout these months. I have great respect for their experience and knowledge and I am very grateful for having had me the opportunity to learn from them.

I started the search for a thesis topic with two criterias in mind. Firstly, I was driven to work and gain experience with flying robots in a research environment. Secondly, as I completed both my B.Sc. and M.Sc. studies at TU Delft, I wanted use this opportunity to set a foot in my hometown Munich for my future professional development. This led me to doing my thesis work at the DLR in Oberpfaffenhofen with Christophe de Wagter being my mentor at TU Delft. I want to thank again Christophe de Wagter for enabling this collaboration with DLR and for his guidance.

I also want to thank all my friends at DLR with whom I had many interesting conversations and who made my time DLR a unique experience, most importantly Pascal Möller, Wim van Ekeren, Jongseok Lee, Stefan Büttner, Tim Wunderlich, Ulrike Leipscher und Arne Sachtler.

This work concludes my 6 years of study at TU Delft. Therefore I feel this is the moment to express my gratitude to my parents and to my three sisters Maria, Francisca and Sofia for their love and unconditional support. I also want to express my appreciation to my girlfriend Simone for her love and patience.

Summary

State of the art trajectory generation schemes for quadrotors assume a simple dynamic model. They neglect aerodynamic effects such as induced drag and blade flapping and assume that no wind is present. In order to overcome this limitation, this thesis investigates a trajectory optimization scheme based upon Differential Dynamic Programming (DDP). There are various software-implementations of the DDP scheme. For future deployment on robotic hardware the software is required to be computationally efficient and to be open-source. The C++ template-based optimization library named GCOP developed at JHU was deemed suitable so it was selected for this purpose. Before implementing the solver, a full model of the Crazyflie Nano Quadcopter is identified experimentally. The model considers a first order term for the aerodynamic forces in each axis of the body frame. The solver is validated, normalized and the performance is benchmarked. This method yields reliable minimum control-effort trajectories. The computation time required to reach the optimum solution is studied for different dicretizations, and for different choices of solver parameters. A control scheme is proposed and studied in Monte-Carlo simulations. It is robust and able to handle large modelling errors in mass and moment of inertia while ensuring minimal error on the final state.

Acronyms

ACADO	Automatic Control and Dynamic Optimization Toolkit
DDP	Differential Dynamic Programming
DOF	Degrees of Freedom
DP	Dynamic Programming
EOM	Equations of Motion
GCOP	Geometric Control Optimization and Planning Library
GPM	Gauss Pseudospectral Method
GPOPS	General Purpose Optimal Control Software
MEMS	Microelectromechanical System
NLP	Nonlinear Program
SNOPT	Automatic Control and Dynamic Optimization Toolkit
SOS	Sparse Optimization Suite

List of Symbols

Greek Symbols

$\boldsymbol{\Gamma}_\mathrm{c}$	contact torque
$\boldsymbol{\Gamma}_\mathrm{d}$	aerodynamic torque
$\boldsymbol{\Gamma}_\mathrm{m}$	modelling error torque
$\boldsymbol{\Gamma}_\mathrm{p}$	propulsion torque
λ_i	rotation direction of i-th rotor
μ	advance ratio
$\boldsymbol{\omega}$	rotational speed of quadcopter
$\overline{\omega}_i$	rotational speed of i-th rotor
ϕ	roll angle
ψ	yaw angle
ρ	air density
$\boldsymbol{\tau}_\mathrm{c}$	contact wrench
$\boldsymbol{\tau}_\mathrm{d}$	aerodynamic wrench
$\boldsymbol{\tau}_{\mathrm{d},i}$	aerodynamic wrench for i-th propeller
$\boldsymbol{\tau}_\mathrm{m}$	modelling error wrench
$\boldsymbol{\tau}_\mathrm{p}$	propulsion wrench
θ	pitch angle

Roman Symbols

$\mathbf{0}_{m\times n}$	m × n null matrix
A	rotor disk surface area

$\mathbf{A}_{\mathrm{d},i}$	drag matrix
B	body frame
$c_{\mathrm{f,lat}}$	lateral flapping drag coefficient
$c_{\mathrm{f,lon}}$	longitudinal flapping drag coefficient
$c_{\mathrm{i},1}$	induced drag coefficient
$c_{\mathrm{i},2}$	induced drag coefficient
C_{p}	power coefficient
P	propeller power
C_{Q}	torque coefficient
C_t	thrust coefficient
C_x^B	linear drag constants in body frame along x-axis
C_y^B	linear drag constants in body frame along y-axis
C_z^B	linear drag constants in body frame along z-axis
D	rotor diameter
$\mathbf{d}_i$	location of i-th rotor with respect to origin of the body frame
$\mathbf{f}_{\mathrm{c}}$	contact force
$\mathbf{f}_{\mathrm{d}}$	aerodynamic force
$\mathbf{f}_{\mathrm{m}}$	modeling error force
$\mathbf{f}_{\mathrm{p}}$	propulsion force
g	standard acceleration due to gravity
h	timestep
I	inertial reference frame
$\mathbf{I}_{m\times n}$	$m \times n$ identity matrix
$\mathcal{I}$	moment of inertia matrix
$\mathcal{I}_y$	moment of inertia around y-axis of quadcopter
J	objective function
k	number of DDP iterations
L	rotor arm of quadcopter
$\mathbf{M}$	generalized inertia matrix
m	quadcopter mass
N	discretization size
O_B	origin of body reference frame
O_I	origin of inertial reference frame

Q_i	torque of i-th rotor
$\mathbf{Q}_\mathrm{f}$	final state weighing matrix
R	rotor radius
$\mathbf{r}$	displacement of O_B with respect to O_I
$\mathbf{R}$	input weighing matrix
$\mathbf{R}_\mathrm{bi}$	rotation matrix from inertial to body frame of reference
T	total thrust force
t_0	trajectory start time
t_f	trajectory end time
T_h	hover thrust
T_i	thrust force of i-th rotor
U	slipstream velocity
$\mathbf{u}(t)$	control input
$\mathbf{u}^*$	optimal control input
$\hat{\mathbf{i}},\ \hat{\mathbf{j}},\ \hat{\mathbf{k}}$	unit vectors in right-handed frame of reference
v_h	induced velocity in hover
v_i	induced velocity
$\mathbf{v}_\infty$	freestream airspeed
$\mathbf{v}_{\infty,i}$	freestream velocity at i-th rotor
$\mathbf{v}_\mathrm{r}$	true airspeed
$\mathbf{v}_\mathrm{w}$	windspeed
$v_{\mathrm{r},x}$	true airspeed in x-direction of inertial frame
$V(\mathbf{x}, i)$	cost-to-go function at discrete timestep $t = i$
v_xy	freestream velocity in rotor plane
$v_{\mathrm{r},z}$	true airspeed in z-direction of inertial frame
v_z	freestream velocity perpendicular to rotor plane
$\mathbf{x}(t)$	state vector

List of Figures

List of Tables

Contents

Chapter 1

Introduction

In this work a method for optimizing quadrotor trajectories is developed. This method includes a drag model and takes advantage of local wind information, reducing the error between the generated trajectory and the real world. The generated trajectory is then exploited in a feedforward manner. To conclude, this system is tested in a simulation in order to benchmark and study its performance.

This chapter will first give a general introduction to the problem. In section 1-2, the research question is then presented. Finally, the thesis outline is given in section 1-3.

1-1 General Introduction

Multicopters and especially quadrotors have received a lot of public attention since the early 2000s due to the increased availability of Microelectromechanical System (MEMS) components as well as advances in low-cost hardware. A multicopter is a rotorcraft consisting of multiple rotors mounted to electric motors on a frame. This frame typically consists of rods which are coupled in the centre. There the electronic hardware such as batteries, electronic speed controllers, and sensing units are attached.

Multicopters are mechanically simple and robust since typically stiff propeller blades are the only moving parts. They have a high thrust-to-weight ratio and are capable of large angular accelerations, thus they are highly maneuverable. Their ability to hover and to takeoff and land vertically creates many use-cases. Some current and future applications of multicopters include inspection of power grids in the energy industry (Luque-Vega, Castillo-Toledo, Loukianov, & Gonzalez-Jimenez, 2014), autonomous planting and harvesting in the agricultural industry (Zairi, Hazry, et al., 2010), (IPATE, VOICU, & DINU, 2015), and mail delivery (Mathew, Smith, & Waslander, 2015). In the consumer electronics market they are increasingly used for aerial photography (Luque-Vega et al., 2014), (Markwalter, 2015). Furthermore, multicopters are widely regarded as interesting and relatively cost-effective platforms for research in the fields of mechatronics, robotics and especially control, since they

are underactuated and characterized by nonlinear dynamics. In this work quadrotors are considered. Quadrotors have four rotors and typically weigh less than three kilograms.

A typical control architecture for a quadrotor has several layers and begins with a trajectory generator, which provides a reference trajectory to a tracking controller. A reference trajectory consists of the quadrotor's state as a continuous function of time for a predefined number of discrete time steps. The lower-level trajectory tracking controller receives the reference trajectory as an input and generates the corresponding control outputs to the motors.

The aim of this thesis is to develop a method for generation of trajectories for a quadrotor. It should be possible to implement this method in the future on a embedded quadcopter system. As a rough approximation, let's assume the feedback control runs at 500 Hz. It can be assumed that the trajectory generation should run not more than one order of magnitude slower, which leads to a maximum computation time of 0.02 s. This is only an approximation, but as computation time will be critical for a successful future hardware integration, the solver should be implemented in a machine-oriented language, such as C++. The method should show an improvement to the methods currently available, which will be achieved through feed-forward consideration of aerodynamic effects, including them in the model. This model should incorporate propeller aerodynamics, since the quadrotor aerodynamics are dominated by them. Furthermore, it should exploit knowledge of the local wind velocity, such that in the future a wind estimation method can be integrated, such as presented by *Tomic et al.* in (Tomic & Haddadin, 2015).

The problem is nonlinear and includes constraints. We will express it as an optimal control problem, and then require a solver which can handle nonlinear problems. The open source Geometric Control Optimization and Planning Library (GCOP) framework, containing a Differential Dynamic Programming (DDP) implementation in C++ (Kobilarov, 2016), will be used for this.

1-2 Research Question

The following main research question is posed:

Is it possible, and if so how, to exploit the aerodynamics model in a feed-forward manner to improve the flight performance of a quadrotor under wind influence ?

A number of technical and scientific sub-questions can be formulated, which help to answer the main question.

These subquestions are listed below:

Trajectory Generation

1. Which method should be used to generate the trajectories?
2. Is it necessary to initialize the trajectories in order for the solver to find the solution? If yes, which method should be used to initialize the trajectories ?
3. How should the objective function be defined?

4. How sensible is the solver to solver parameters?

5. What is the solver performance with and without drag model?

   Trajectory Tracking

6. Can the flight performance be improved by exploiting the aerodynamics model in a feed-forward manner ? Which control structure is suitable for this purpose?

1-3 Thesis outline

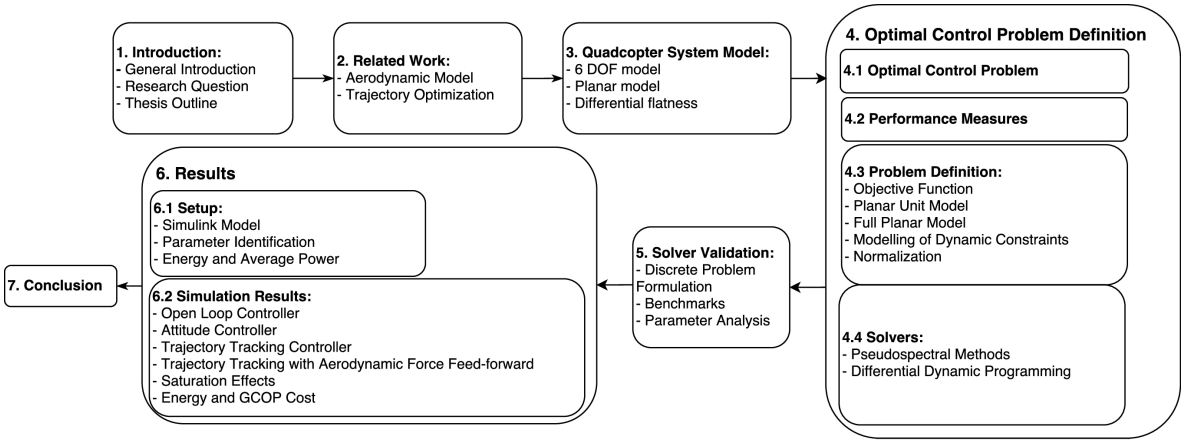


Figure 1-1: Thesis structure in flowchart.

This thesis contains 7 chapters, starting with this chapter, and 2 appendices. Chapter 2 elaborates on related work found in literature. In chapter 3, the 6 DOF model and the simplified, planar model are presented. In chapter 4, the problem is formulated as an optimal control problem and the solvers are introduced. In chapter 5, the problem is defined in discrete time, the solver is validated and it's sensibility to solver parameters is studied. In chapter 6, first the simulation model and the method for the experimental identification of the aerodynamic parameters is presented. Then the simulation results are presented, for different types of controllers. In chapter 7 the simulation results are linked to the research questions and recommendations are given.

Chapter 2

Related Work

In this chapter, the literature review is presented. Firstly, the aerodynamic model of a quadrotor will be presented, followed by the different optimization methods available.

2-1 Aerodynamic Model

In the first efforts to model small scale rotorcrafts, dynamic models excluded drag and propeller aerodynamics (T. Hamel, Mahony, Lozano, & Ostrowski, 2002), (Omari, Hua, Ducard, & Hamel, 2013). A large amount of control studies have been done ignoring these effects, as controllers are designed about hover conditions where aerodynamic effects can be neglected. The reason is that feedback control usually has sufficient bandwidth to compensate the unmodeled aerodynamic effects, which are furthermore typically of dissipative form.

More recently, aerodynamic models have been exploited for control purposes in (Omari et al., 2013), (Pierre-Jean, Callou, Vissiere, & Petit, 2011), (Mahony, Kumar, & Corke, 2012), (Martin & Salaun, 2010), (Pounds, Mahony, & Corke, 2010).

Omari et al. (Omari et al., 2013) implemented the aerodynamic model with blade flapping and induced drag terms into an online feedback controller. This was done under the motivation that the quadrotor model ceases to be differentially flat when incorporating aerodynamic forces. By using the aerodynamic model they introduced a term to compensate the expected aerodynamic force. This was done in a feedforward manner due to computational complexity.

The results in (Omari et al., 2013) illustrate that when a trajectory controller with no aerodynamic model is used, the effects from blade flapping and induced drag cause disturbances, affecting the flight performance. They furthermore show that for a high reference velocity, a controller with only first order effects in the model shows poor trajectory tracking.

As these disturbances are typically of a high frequency, they require a high level of integral control action from the controller, which leads to poor control performance. *Omari et al.* compare the response of the system given a ramp input in position, between a controller

with no aerodynamic model and one with a first-order aerodynamic model. With no aerodynamic model, the flight trajectory shows a significant offset and the integral control action is high. The first-order aerodynamics augmented controller achieved the tracking objective and showed a much lower integral action for a low velocity reference $v_{\text{ref}} = 1\text{m}\,\text{s}^{-1}$.

Nevertheless, second order aerodynamics effects are often neglected, since they are proportional to the square of the quadrotor linear velocity and therefore are small near hovering, (Martin & Salaun, 2010), (Benallegue, Mokhtari, & Fridman, 2006). They are generally seen as unmodeled disturbances, which should be compensated for by the controller.

P. Martin et al. show in (Martin & Salaun, 2010) that aerodynamic effects proportional to the sum of the propeller angular velocities $\sum \varpi_i$ times the quadrotor linear velocity $\dot{\boldsymbol{r}}$ and the angular velocity $\boldsymbol{\omega}$ appear and should be considered as a necessary addition to the simple propeller model acting exclusively in the direction of the propeller axis. This aerodynamic force $\mathbf{F}_{\text{H}}$ is called H-force in helicopter literature (Watkinson, 2004), (A. R. S. Bramwell, 2001), and is defined for the i-th rotor as $\mathbf{F}_{\text{H},i} = -\varpi_i \lambda_{\text{h},i} \dot{\boldsymbol{r}}$, with $\lambda_{\text{h},i}$ a propeller specific constant (Martin & Salaun, 2010).

Finally, in a wide range of recent publications, i. e. (Guillaume Allibert, Abeywardena, Bangura, & Mahony, 2014), (Omari et al., 2013), (Leishman, Macdonald, Beard, & McLain, 2014), (Waslander & Wang, 2009), (Tomic & Haddadin, 2015), only aerodynamic forces proportional to the square of the propeller angular velocity times the linear velocity or even only proportional to the linear velocity are considered.

These choices are a matter of model fidelity and are a trade off between model complexity and model error.

2-2 Trajectory Optimization

Trajectory optimization is the process of developing trajectories that minimize a performance measure.

It can be done in the state space or the control space, solving either an inverse or forward dynamics problem. In an inverse dynamics problem the state is represented explicitly and an optimization problem is solved (Mueller, Hehn, & D'Andrea, 2015). In a forward dynamics problem the control inputs are parametrized and the states are computed using forward integration (Tassa, Mansard, & Todorov, 2014). The forward dynamics problem has the advantage that state-control trajectories are dynamically feasible. Therefore dynamic constraints on the inputs are not needed.

When trajectories are generated by superposing a speed profile on a previously generated path, trajectory tracking involves a trade off between the complexity of the planned path and the input update rate. *G.M. Hoffmann et al.* present in (Hoffmann, Waslander, & Tomlin, 2008) a trajectory tracking controller for quadrotors, which decouples these two components. The path-planning is not required to be dynamically feasible, as the presented algorithm modifies the speed profile of the input path to be dynamically feasible. It does this in two steps. Firstly, the cross track acceleration constraint is satisfied by computing at every waypoint a maximum allowable speed. Then it computes the optimal speed profile that satisfy track acceleration constraints, minimum desired speed and the previously computed maximum allowable speed.

J. H. Gillula et al. model in (Gillula, Huang, Vitus, & Tomlin, 2010) the quadrotor system as a collection of hybrid modes, one for each operating regime. Aerobatic maneuvers are then decomposed into series of discrete maneuvers. Guaranteed safe transitions between modes are obtained from the Hamilton-Jacobi differential game formulation, resulting in provably safe switching conditions on altitude, altitude rate, attitude and attitude rate. This method was implemented for a backflip maneuver, a maneuver split up into three main modes. This method is especially suitable for complex maneuvers on nonlinear systems

Mellinger et al. state in (Mellinger & Kumar, 2011) that LQR-tree based searches, as presented in (Tedrake, 2009) are impractical for a system with six degrees of freedom. They present a flatness-based minimum snap trajectory generation method, which has the ability to run through a series of specified waypoints and to take into account constraints on positions, velocities, accelerations and inputs.

A method which has been tested successfully on a quadrotor system is the two-step method proposed by *M. W. Mueller et al.* in (Mueller et al., 2015). The authors first use a flatness-based state-to-state planner to generate the trajectory in closed form, while ignoring feasibility constraints. The position trajectories are fifth-order time-dependent polynomials. By formulating an optimal control problem, the constants of the polynomials are solved for. The objective function minimizes the upper bound of the product of the quadrotor inputs. In the second step the feasibility is checked recursively. Advantage is taken from the property that specific polynomials can be rapidly verified for constraints on the system inputs, position, velocity and/or acceleration. This algorithm can be applied when a search over a large space of trajectories is required for a nonconvex problem.

Nonlinear optimal control problems generally cannot be solved analytically and therefore numerical methods have to be used to solve these optimal control problems. The numerical methods generally raise problems related to the computation cost, stability, robustness and the understanding of the nature of the obtained solution, (Geoffroy, Mansard, Raison, Achiche, & Todorov, 2014). Differential dynamic programming is an algorithm in the area of numerical optimal control, which is nearly equivalent to the Newton descent algorithm (DE O. PANTOJA, 1988). It's strength is that it is easy to implement in an efficient way. For example, it has been implemented for effective real-time control of a 25 DOF Robot by (Tassa et al., 2014). The algorithm starts with an initial trajectory for the states and controls. The candidate solution is iteratively modified in two-steps. Firstly, a quadratic model of the variation of the candidate trajectory is computed in a backward loop, together with the corresponding linear-quadratic regulator. Then the candidate trajectory is modified in a forward loop.

In this thesis, trajectory optimization using Differential Dynamic Programming will be studied, with and without a linear aerodynamic model. Furthermore a controller will be developed, which is able to exploit the aerodynamic model.

Chapter 3

Quadcopter System Model

The Equations of Motion (EOM) of a quadrotor will be presented in this chapter, both for a full six degrees of freedom model and a planar model. These models will be used throughout the work.

3-1 Six degrees of freedom equations of motions

3-1-1 Reference Frames

The inertial frame is $I = \{O_I; \hat{\mathbf{i}}_I, \hat{\mathbf{j}}_I, \hat{\mathbf{k}}_I\}$ and the body frame is $B = \{O_B; \hat{\mathbf{i}}_B, \hat{\mathbf{j}}_B, \hat{\mathbf{k}}_B\}$, with the unit vectors $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$, $\hat{\mathbf{k}}$. Following aerospace convention, the i-axis points forward, the j-axis to the right and the k-axis points downwards. Considering the inertial frame with origin O_I, $\hat{\mathbf{i}}_I$ points to the front and is parallel to the ground, $\hat{\mathbf{j}}_I$ points to the right side and is parallel to the ground. The unit vector $\hat{\mathbf{k}}_I$ is orientated downwards, perpendicular to the ground.

The displacement of the origin of the body frame O_B with respect to the inertial frame origin O_I is defined by $\mathbf{r} \in \mathbb{R}^3$. The body frame B is rotated with respect to the inertial frame I by the euler angles for roll ϕ, pitch θ and yaw ψ, through the rotation matrix $\mathbf{R}_{\text{bi}} \in \mathbb{R}^{3\times3}$. Furthermore, $\mathbf{d}_i$ denotes the location of the i-th rotor with respect to the center of the body frame. For simplicity, the horizontal plane of the rotors and the quadcopter will be assumed to be aligned.

3-1-2 Rigid body equations of motion

The full 6 Degrees of Freedom (DOF) equations of motion are the well-known equations of motion of a rigid body plus all external forces,

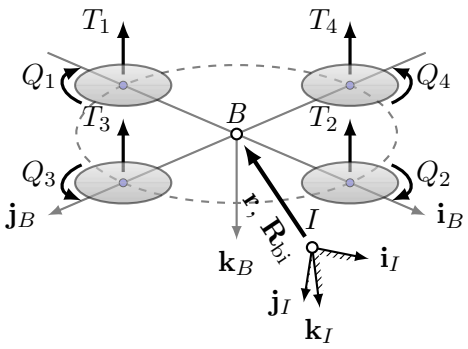


Figure 3-1: Sketch of quadrotor system (Martin & Salaun, 2010). The rotational speed of the i-th propeller is $\overline{\omega}_i$.

$$m\ddot{\boldsymbol{r}} = mg\hat{\mathbf{k}} + \mathbf{f}_\mathrm{p} + \mathbf{f}_\mathrm{d} + \mathbf{f}_\mathrm{c} + \mathbf{f}_\mathrm{m} \ , \tag{3-1}$$

$$\mathcal{I}\dot{\boldsymbol{\omega}} = [\boldsymbol{\omega}]_x\mathcal{I}\boldsymbol{\omega} + \boldsymbol{\Gamma}_\mathrm{p} + \boldsymbol{\Gamma}_\mathrm{d} + \boldsymbol{\Gamma}_\mathrm{c} + \boldsymbol{\Gamma}_\mathrm{m} \ , \tag{3-2}$$

$$\dot{\mathbf{R}}_\mathrm{bi}^\mathrm{T} = \mathbf{R}_\mathrm{bi}^\mathrm{T}[\boldsymbol{\omega}]_\times \ , \tag{3-3}$$

with the moment of inertia matrix $\mathcal{I} \in \mathbb{R}^{3\times3}$ and the quadcopter mass m. The force $\mathbf{f} \in \mathbb{R}^3$ is defined in the inertial frame and the torque $\boldsymbol{\Gamma} \in \mathbb{R}^3$ in the body frame. The propulsion, aerodynamic, contact and modeling error forces/torques are denoted as $\mathbf{f}_\mathrm{p}/\boldsymbol{\Gamma}_\mathrm{p}$, $\mathbf{f}_\mathrm{d}/\boldsymbol{\Gamma}_\mathrm{d}$, $\mathbf{f}_\mathrm{c}/\boldsymbol{\Gamma}_\mathrm{c}$, $\mathbf{f}_\mathrm{m}/\boldsymbol{\Gamma}_\mathrm{m}$, respectively. $[\boldsymbol{\omega}]_\times$ denotes the skew symmetric matrix of $\boldsymbol{\omega}$.

Now we condense these relations into the generalized formulation,

$$\mathbf{M}\dot{\mathbf{v}} + \mathbf{C}(\mathbf{v})\mathbf{v} + \mathbf{g} = \boldsymbol{\tau}_\mathrm{p} + \boldsymbol{\tau}_\mathrm{d} + \boldsymbol{\tau}_\mathrm{c} + \boldsymbol{\tau}_\mathrm{m} \ , \tag{3-4}$$

with the generalized velocity defined as $\mathbf{v} = [\dot{\mathbf{r}}^\mathrm{T};\ \boldsymbol{\omega}^\mathrm{T}]^\mathrm{T} \in \mathbb{R}^6$ and the wrench defined as $\boldsymbol{\tau} = [\mathbf{f}^\mathrm{T};\ \boldsymbol{\Gamma}^\mathrm{T}]^\mathrm{T} \in \mathbb{R}^6$.

The generalized inertia matrix $\mathbf{M}$ is defined as

$$\mathbf{M} = \begin{bmatrix} m\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathcal{I} \end{bmatrix} \in \mathbb{R}^{6\times6} \ , \tag{3-5}$$

whereby the identity and null matrix are defined as $\mathbf{I}_{m\times n} \in \mathbb{R}^{m\times n}$ and $\mathbf{0}_{m\times n} \in \mathbb{R}^{m\times n}$, respectively.

The other matrices used in equation (3-4) are the coriolis and centripetal terms

$$\mathbf{C}(\mathbf{v}) = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & -[\mathcal{I}\boldsymbol{\omega}]_\times \end{bmatrix} \in \mathbb{R}^{6\times6}, \tag{3-6}$$

and the weight vector

$$\mathbf{g} = -[mg\hat{\mathbf{k}}^\mathrm{T}\ \mathbf{0}_{1\times3}]^\mathrm{T} \in \mathbb{R}^{6\times1} \ , \tag{3-7}$$

with g being the gravitational acceleration.

We assume the vehicle is flying in an environment free of obstacles and no external wrenches due to collision, $\boldsymbol{\tau}_\mathrm{c} = 0$.

In the following sections, the drag and propulsion model will be defined.

3-1-3 Propulsion Model

The propeller axes are introduced as $\hat{\mathbf{i}}_P$, $\hat{\mathbf{j}}_P$, $\hat{\mathbf{k}}_P$. Assuming that the propellers are aligned with the body frame, the body axes can be used instead.

The propulsion wrench $\boldsymbol{\tau}_\mathrm{p}$ is the sum of the propulsion wrenches $\boldsymbol{\tau}_{\mathrm{p},i}$ of all the rotors. The propulsion of the i-th propeller is defined as

$$\boldsymbol{\tau}_{\mathrm{p},i} = \begin{bmatrix} \mathbf{R}_\mathrm{bi}^\mathrm{T} T_i(-\hat{\mathbf{k}}) \\ \lambda_i Q_i \hat{\mathbf{k}} + \mathbf{d}_i \times \big(T_i(-\hat{\mathbf{k}})\big) \end{bmatrix} , \tag{3-8}$$

with the thrust force from the i-th rotor $T_i \in \mathbb{R}$ pointing upwards in the direction of $-\hat{\mathbf{k}}_B$. The drag torque from the i-th rotor is represented by $\lambda_i Q_i$, where $\lambda_i \in \{-1, 1\}$ encodes the rotation direction (positive around $+\hat{\mathbf{k}}_B$) and Q_i the magnitude of the torque.

The true airspeed $\mathbf{v}_\mathrm{r}$ is the difference between the ground speed $\dot{\mathbf{r}}$ and the windspeed $\mathbf{v}_\mathrm{w}$,

$$\mathbf{v}_\mathrm{r} = \dot{\mathbf{r}} - \mathbf{v}_\mathrm{w} . \tag{3-9}$$

The freestream velocity $\mathbf{v}_\infty$, is then computed from the true airspeed $\mathbf{v}_\mathrm{r}$ and the body angular velocity $\boldsymbol{\omega}$

$$\mathbf{v}_\infty = (\mathbf{R}_\mathrm{bi}\mathbf{v}_\mathrm{r} + \boldsymbol{\omega} \times \mathbf{d}) . \tag{3-10}$$

For notational simplicity, we will now derive the thrust T, torque Q and power P for a single propeller in isolation. Based on momentum theory (Leisham, 2006), the thrust force of a propeller depends on the slipstream velocity U and induced velocity v_i. The thrust T is defined as

$$T = 2\rho A v_\mathrm{i} U , \tag{3-11}$$

where ρ is the air density and A is the rotor disk surface area.

The propeller slipstream velocity U is the sum of the freestream velocity $\mathbf{v}_\infty$ and the induced velocity v_i pointing downwards in the propeller frame along $\mathbf{k}_B$, defined as

$$U = ||v_\mathrm{i}\mathbf{k}_B + \mathbf{v}_\infty|| . \tag{3-12}$$

The rotor induced velocity v_i can be obtained from the implicit formulation

$$v_\mathrm{i} = \frac{v_\mathrm{h}}{\sqrt{v_\mathrm{xy}^2 + (v_\mathrm{i} + v_\mathrm{z})^2}} , \tag{3-13}$$

which can be solved by several Newton-Raphson iterations (Tomic & Haddadin, 2015). The freestream velocity in the rotor plane v_xy is

$$v_\mathrm{xy} = ||\mathbf{v}_\mathrm{xy}|| = ||\mathbf{v}_\infty - (\mathbf{v}_\infty \cdot \hat{\mathbf{k}}_B)\hat{\mathbf{k}}_B|| , \tag{3-14}$$

while the freestream perpendicular to the rotor plane is defined as

$$v_\mathrm{z} = ||(\mathbf{v}_\infty \cdot \hat{\mathbf{k}}_B)\hat{\mathbf{k}}_B|| . \tag{3-15}$$

The induced velocity in hover v_h is obtained by substituting the hover thrust T_h

$$v_\mathrm{h} = \sqrt{\frac{T_\mathrm{h}}{2\rho A}} \,. \tag{3-16}$$

Additionally, the thrust T_i can be modeled with

$$T = \rho D^4 C_\mathrm{T} \overline{\omega}^2 \,, \tag{3-17}$$

where C_T is the thrust coefficient.

The torque Q_i generated by the rotation of the i-th propeller is modeled as

$$Q = \rho D^5 C_\mathrm{Q} \overline{\omega}^2 \,, \tag{3-18}$$

with the propeller diameter D, the air density ρ, torque coefficient C_Q and propeller rotational speed $\overline{\omega}$ in rad s^{-1}. Through substitution of $C_\mathrm{Q} = \frac{C_\mathrm{p}}{2\pi\overline{\omega}}$ we get

$$Q = \rho D^5 \frac{C_\mathrm{p}}{2\pi} \overline{\omega} \,, \tag{3-19}$$

with the power coefficient C_p.

The power P is modeled with

$$P = \rho D^5 C_\mathrm{p} \overline{\omega}^3 \,. \tag{3-20}$$

where C_P is the power coefficient.

3-1-4 Blade flapping and induced drag

Now the drag model is introduced, based upon blade flapping and induced drag. When a quadrotor is in forward flight, the advancing rotor blade has a higher tip velocity and therefore generates more lift than the retreating blade, which causes them to deflect up and down, as they are not completely stiff. This movement is called blade flapping, causing the thrust vector to deflect back, causing an induced drag component, as depicted in Figure 3-2.

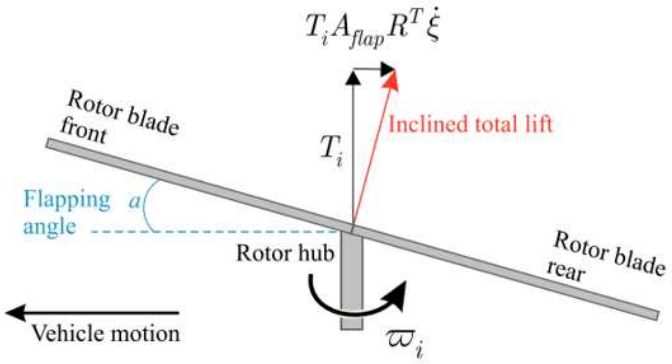


Figure 3-2: Sketch for blade-flapping (Omari et al., 2013)

From (Omari et al., 2013), the aerodynamic effects can be described by a lumped expression with a term proportional to the thrust force T_i and the linear velocity $\dot{\boldsymbol{r}}$ transformed to the body frame B. The drag wrench $\boldsymbol{\tau}_{\mathrm{d},i} \in \mathbb{R}^6$ from the i-th propeller is defined proportional to the thrust T_i and to the freestream velocity $\mathbf{v}_{\infty,i} \in \mathbb{R}^3$ as:

$$\boldsymbol{\tau}_{\mathrm{d},i} = -\mathbf{R}_{\mathrm{bi}}^T \mathbf{D}_i \mathbf{R}_{\mathrm{bi}} \begin{bmatrix} \mathbf{v}_{\infty,i} \\ \mathbf{0}_{3\times 1} \end{bmatrix} . \tag{3-21}$$

Finally, $\mathbf{D}_i$ is defined as

$$\mathbf{D}_i = \begin{bmatrix} T_i \mathbf{A}_{\mathrm{d},i} & \mathbf{0}_{3\times 3} \\ \left(\mathbf{d}_n \times T_i \mathbf{A}_{\mathrm{d},i} \right) & \mathbf{0}_{3\times 3} \end{bmatrix} , \tag{3-22}$$

where the matrix $\mathbf{A}_{\mathrm{d},i}$ contains the lumped drag parameters as

$$\mathbf{A}_{\mathrm{d},i} = \begin{bmatrix} c_{\mathrm{f,lon}} + c_{\mathrm{i},1} & -c_{\mathrm{f,lat}} & 0 \\ c_{\mathrm{f,lat}} & c_{\mathrm{f,lon}} + c_{\mathrm{i},2} & 0 \\ 0 & 0 & 0 \end{bmatrix} , \tag{3-23}$$

where $c_{\mathrm{f,lon}}$, $c_{\mathrm{f,lat}}$ are the longitudinal and lateral flapping drag coefficients and $c_{\mathrm{i},1}$, $c_{\mathrm{i},2}$ are induced drag coefficients (Tomic & Haddadin, 2015).

Higher-order terms in linear and angular velocities, which can be derived from classical blade theory, are neglected, as we assume a small advance ratio μ (Martin & Salaun, 2010). The advance ratio μ is a non-dimensional measure for rotor velocity, characterizing the angle of attack on the blade sections, regardless of the actual true airspeed. It is defined as the ratio of the freestream speed in the propeller plane to the rotor tip speed,

$$\mu = \frac{v_{\mathrm{xy}}}{\overline{\omega} R} , \tag{3-24}$$

with v_{xy} the freestream velocity in the propeller plane, $\overline{\omega}$ the propeller rotational speed and R the rotor radius (Leisham, 2006).

3-2 Planar Model

In order to simplify the analysis of the system a planar model of a quadrotor with motion only in the (x, z) plane is derived (i.e. $\phi = \psi = 0$), as is done in (Sreenath, Michael, & Kumar, 2013), (Cabecinhas, Naldi, Marconi, Silvestre, & Cunha, 2012), (Sharifi, Zhang, & Gordon, 2011).

The thrust force T is pointing upwards in the body frame, as shown in Figure 3-3. The gravitational acceleration points in the positive z direction. The specific model is defined as

$$\ddot{x} = -\frac{T}{m} \sin\theta = -u_1 \sin\theta , \tag{3-25}$$

$$\ddot{z} = g - \frac{T}{m} \cos\theta = g - u_1 \cos\theta , \tag{3-26}$$

$$\ddot{\theta} = \frac{\tau}{\mathcal{I}_y} = u_2 . \tag{3-27}$$

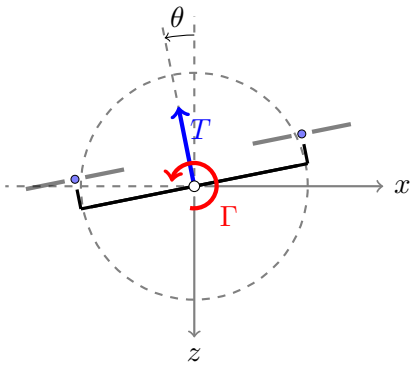


Figure 3-3: Planar model of the quadrotor (Tomic et al., 2014) .

The first control input u_1 is the thrust normalized by the mass and the second control input u_2 is the torque normalized by the moment of inertia .

The planar equations of motion are augmented by only the linear terms of the aerodynamics model. The drag forces in the body frame are modeled with the linear drag constants C_x^B and C_z^B as

$$\begin{bmatrix}\ddot{x}\\ \ddot{z}\end{bmatrix} = -u_1 \begin{bmatrix}\sin\theta\\ \cos\theta\end{bmatrix} + \begin{bmatrix}0\\ g\end{bmatrix} - \mathbf{R}_{\text{bi}}^{\text{T}} \begin{bmatrix}C_x^B & 0\\ 0 & C_z^B\end{bmatrix} \mathbf{R}_{\text{bi}} \begin{bmatrix}v_{\text{r},x}\\ v_{\text{r},z}\end{bmatrix} , \tag{3-28}$$

$$\ddot{\theta} = u_2 . \tag{3-29}$$

where $v_{\text{r},x}$ and $v_{\text{r},z}$ are the true windspeeds in the inertial frame. The rotation matrix $\mathbf{R}_{\text{bi}} \in \mathbb{R}^{2\times 2}$ is defined as

$$\mathbf{R}_{\text{bi}} = \begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix} . \tag{3-30}$$

3-3 Differential Flatness

In this section, a general definition of flatness will be presented (Josua Braun, 2015) and applied to obtain the input-output linearization of the planar model.

Definition 1

(Differential Flatness)
The nonlinear system

$$\dot{\mathbf{x}} = \mathbf{f}\Big(\mathbf{x}(t), \mathbf{u}(t)\Big), \quad \mathbf{x}(0) = \mathbf{x}_0 , \tag{3-31}$$

$$\mathbf{x} \in \mathbb{R}^n, \quad \mathbf{f} \in \mathbb{R}^n, \quad \mathbf{u} \in \mathbb{R}^p, \tag{3-32}$$

is *differentially flat*, if one can define p outputs y_i which are dependent on the state vector $\mathbf{x}(t)$, input vector $\mathbf{u}(t)$ and a finite number of derivatives of the inputs $\dot{\mathbf{u}}(t), \ldots, \overset{(s)}{\mathbf{u}}(t)$

$$\begin{bmatrix}y_1\\ \vdots\\ y_p\end{bmatrix} = \mathbf{y} = \begin{bmatrix}c_1\Big(\mathbf{x}, \mathbf{u}, \dot{\mathbf{u}}, \ldots, \overset{(\alpha)}{\mathbf{u}}\Big)\\ \vdots\\ c_p\Big(\mathbf{x}, \mathbf{u}, \dot{\mathbf{u}}, \ldots, \overset{(\alpha)}{\mathbf{u}}\Big)\end{bmatrix} = \mathbf{c}\Big(\mathbf{x}, \mathbf{u}, \dot{\mathbf{u}}, \ldots, \overset{(\alpha)}{\mathbf{u}}\Big) , \tag{3-33}$$

such that the input and state variables can be represented by functions which are dependent on the flat output vector $\mathbf{y} \in \mathbb{R}^p$ and its derivatives

$$\begin{aligned} \mathbf{x} &= \mathbf{a}\left(\mathbf{y}, \dot{\mathbf{y}}, \ldots, \overset{(\beta)}{\mathbf{y}}\right) , \\ \mathbf{u} &= \mathbf{b}\left(\mathbf{y}, \dot{\mathbf{y}}, \ldots, \overset{(\beta+1)}{\mathbf{y}}\right) . \end{aligned} \tag{3-34}$$

The planar model without aerodynamics is differentially flat (Mellinger & Kumar, 2011). Therefore all states (θ, $\dot{\theta}$, $\ddot{\theta}$, x, $\dot{x}$, $\ddot{x}$, z $\dot{z}$, $\ddot{z}$) and inputs (u_1, u_2) can be computed from the desired position trajectories (and derivatives thereof). The pitch angle θ and its derivatives ($\dot{\theta}$, $\ddot{\theta}$) can be obtained from equations (3-25), (3-26), (3-27) through algebraic manipulation. Solving equation (3-25) for u_1 we get

$$u_1 = -\frac{\ddot{x}}{\sin\theta} . \tag{3-35}$$

Now we can substitute equation (3-35) into equation (3-26) and obtain

$$\tan(\theta) = \frac{\ddot{x}}{\ddot{z} - g} . \tag{3-36}$$

Solving for θ,

$$\theta = \operatorname{atan2}\left(-\ddot{x}, g - \ddot{z}\right) . \tag{3-37}$$

The first and the second derivatives of the angle θ are obtained through differentiation of eq. (3-37),

$$\dot{\theta} = \frac{\dddot{x}(\ddot{z} - g) - \dddot{z}\ddot{x}}{(g - \ddot{z})^2 + \ddot{x}^2} , \tag{3-38}$$

$$\begin{aligned} \ddot{\theta} = \frac{1}{((g - \ddot{z})^2 + \ddot{x}^2)^2}[&((g - \ddot{z})^2 + \ddot{x}^2)(\ddddot{x}(\ddot{z} - g) - \ddddot{z}\ddot{x}) \\ &-(\dddot{x}(\ddot{z} - g) - \dddot{z}\ddot{x})(2\dddot{z}(\ddot{z} - g) + 2\dddot{x}\ddot{x})] . \end{aligned} \tag{3-39}$$

Substituting equation (3-36) into (3-35), input u_1 is computed,

$$u_1 = \sqrt{(g - \ddot{z})^2 + \ddot{x}} . \tag{3-40}$$

The torque input u_2 is obtained through the system model in equation (3-27) and equation (3-39),

$$u_2 = \ddot{\theta} . \tag{3-41}$$

This derivation also shows that the planar system (3-25)-(3-27) is differentially flat.

As equation (3-28) shows, the model with drag is made up of a first-order as well as a second-order derivative of position $\mathbf{x}$. Therefore, the relation between $\mathbf{u}$ and $\dot{\mathbf{x}}$ is implicit. The conclusion is that we can't prove the system to be differentially flat or input-output linearizable (Omari et al., 2013).

Chapter 4

Optimal Control Problem Definition

In optimal control, the general aim is to find "the control signals that will cause a process to satisfy the physical constraints and at the same time minimize (or maximize) some performance criterion" (Kirk, 2006). Defining the problem as an optimal control problem (OCP) allows inclusion of the full nonlinear system dynamics as the mathematical model of the system to be optimized.

4-1 Optimal Control Problem

An OCP is defined by a set of differential equations $\mathbf{f}$ describing the dynamics of the system, a performance measure J as the function of state variables $\mathbf{x}(t)$ and of control variables $\mathbf{u}(t)$, constraints and boundary conditions. Based on (P. Gill, Murray, & Saunders, 2005) and (Maier, 2013), the general optimal control problem is defined as :

Definition 2

(Optimal Control Problem)
Find a control $\mathbf{u}^*$ which causes the system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \quad \mathbf{x} \in \mathbb{R}^N, \quad \mathbf{u} \in \mathbb{R}^M, \quad f : \mathbb{R}^N \times \mathbb{R}^M \to \mathbb{R}^N, \quad t \in [t_0, \mathrm{f}] \,, \tag{4-1}$$

subject to the inequality constraints

$$\mathbf{x}_{\min} \leq \mathbf{x}(t) \leq \mathbf{x}_{\max}, \quad \mathbf{u}_{\min} \leq \mathbf{u}(t) \leq \mathbf{u}_{\max}, \tag{4-2}$$

and the boundary conditions

$$\mathbf{x}(t_0) = \mathbf{x}_0, \qquad \mathbf{x}(t_\mathrm{f}) = \mathbf{x}_\mathrm{f}, \tag{4-3}$$

to follow a trajectory $\mathbf{x}^*$ that minimizes the optimality criterion

$$J(\mathbf{x}(t), \mathbf{u}(t), t) \,. \tag{4-4}$$

4-2 Performance Measures

One can consider various performance measures, depending on the constraints and the nature of the problem. This section will present three commonly used physically motivated performance measures, which will be used in my thesis work.

Definition 3

(Minimum-Time Problem)
If the goal is to minimize the time to transfer the system from the initial state $\mathbf{x}(t_0) = \mathbf{x}_0$ to the final state $\mathbf{x}(t_f) = \mathbf{x}_f$, we are dealing with a minimum-time problem.
The performance measure to be minimized is

$$J = t_\mathrm{f} - t_0 = \int_{t_0}^{t_\mathrm{f}} \mathrm{d}t \ , \tag{4-5}$$

with t_f the first instant of time when $\mathbf{x}$(t) and $\mathbf{x}_f$ intersect.
In General Purpose Optimal Control Software (GPOPS), the solver can be defined as a minimum-time problem, using GCOP that is not possible.

Definition 4

(Terminal Control Problem)
If the final state is not constrained and the goal is minimizing the deviation of the final state from its' desired value $\mathbf{r}(t_\mathrm{f})$, a terminal control problem is defined.

The performance measure is

$$J = \sum_{i=1}^{n} [x_i(t_\mathrm{f}) - r_i(t_\mathrm{f})]^2 \ , \tag{4-6}$$

where n is the number of states. Since positive and negative deviations are equally undesirable, the error is squared. Absolute values could also be used, but the quadratic form in Eq. (4-6) is easier to handle mathematically. Using matrix notation, we have

$$J = [\mathbf{x}(t_\mathrm{f}) - \mathbf{r}(t_\mathrm{f})]^\mathrm{T} [\mathbf{x}(t_\mathrm{f}) - \mathbf{r}(t_\mathrm{f})] \ , \tag{4-7}$$

or this can be written as

$$J = ||\mathbf{x}(t_\mathrm{f}) - \mathbf{r}(t_\mathrm{f})||^2 \ . \tag{4-8}$$

Definition 5

(Minimum Control-Effort Problem)
The goal in a minimum control effort problem is to transfer a system from an arbitrary initial state $\mathbf{x}(t_0) = \mathbf{x}_0$ to a final state, with a minimum expenditure of control effort.

The meaning of the term "minimum control effort" depends upon the particular physical application; therefore, the performance measure may assume various forms. The general performance measure for multiple inputs is

$$J = \int_{t_0}^{t_\mathrm{f}} \mathbf{u}^T(t) \mathbf{R} \mathbf{u}(t) \mathrm{d}t \ , \tag{4-9}$$

where only the diagonal entries of $\mathbf{R}$ are non-zero. These entries weigh the control-effort for each input variable.

4-3 Problem Definition

4-3-1 Objective Function

The objective function must improve the flight performance and ensure that the final state is reached.

Therefore, our objective function J is a combination of the terminal control problem and the minimum control effort problem.

$$J = [\mathbf{x}(t_\mathrm{f}) - \mathbf{r}(t_\mathrm{f})]^T \mathbf{Q}_f [\mathbf{x}(t_\mathrm{f}) - r(t_\mathrm{f})] + \int_{t_0}^{t_\mathrm{f}} [\mathbf{u}^T(t) \mathbf{R} \mathbf{u}(t)] \mathrm{d}t \, . \tag{4-10}$$

This objective function has diagonal cost matrices on the input $\mathbf{R} \in \mathbb{R}^{6\times6}$ and on the final state $\mathbf{Q}_f \in \mathbb{R}^{6\times6}$

4-3-2 Planar Unit-Model

In the first implementation, which was used for the validation in chapter 5, the GCOP solver is configured to use the total torque τ and thrust T acting on the COM of the rigid body as the control variables u_1, u_2. Furthermore, the mass m and the moment of inertia $\mathcal{I}_y$ are unit and therefore do not appear in the equations.

Without Drag

For the planar system without aerodynamic drag, the flat model is defined as

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{z} \\ \dot{\theta} \\ \ddot{x} \\ \ddot{z} \\ \ddot{\theta} \end{bmatrix} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} \dot{x} \\ \dot{z} \\ \dot{\theta} \\ -u_1 s\theta \\ g - u_1 c\theta \\ u_2 \end{bmatrix}, \quad \mathbf{x}(0) = \mathbf{x}_0, \quad \mathbf{x}(t_\mathrm{f}) = \mathbf{x}_\mathrm{f}, \tag{4-11}$$

where the constraints on the controls $\mathbf{u} = [u_1 \; u_2]^T$ are defined by

$$\frac{T_\mathrm{min}}{mg} \leq u_1 \leq \frac{T_\mathrm{max}}{mg}, \quad |u_2| \leq \ddot{\theta}_\mathrm{max}. \tag{4-12}$$

T_min is the minimum and T_max the maximum available thrust, $\ddot{\theta}_\mathrm{max}$ is the absolute limit on the rotational acceleration.

With Drag

For the planar system with aerodynamic drag, true airspeeds $v_{\mathrm{r},x}$, $v_{\mathrm{r},z}$ and drag constants in the body frame C_x^B, C_z^B, we have the following model:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{z} \\ \dot{\theta} \\ \ddot{x} \\ \ddot{z} \\ \ddot{\theta} \end{bmatrix} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} \dot{x} \\ \dot{z} \\ \dot{\theta} \\ -u_1 s\theta \\ g - u_1 c\theta \\ u_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \mathbf{R}_{\mathrm{bi}}^T \begin{pmatrix} C_x^B & 0 \\ 0 & C_z^B \end{pmatrix} \mathbf{R}_{\mathrm{bi}} \begin{pmatrix} v_{\mathrm{r},x} \\ v_{\mathrm{r},z} \end{pmatrix} \\ 0 \end{bmatrix}, \tag{4-13}$$

with the boundary conditions

$$\mathbf{x}(0) = \mathbf{x}_0, \quad \mathbf{x}(t_\mathrm{f}) = \mathbf{x}_\mathrm{f} \ . \tag{4-14}$$

As in the case without drag, the constraints on the controls $\mathbf{u} = [u_1 \ u_2]^T$ are defined by

$$\frac{T_{\min}}{mg} \leq u_1 \leq \frac{T_{\max}}{mg}, \quad |u_2| \leq \ddot{\theta}_{\max}. \tag{4-15}$$

4-3-3 Full Planar Model

In this section, the model definition used in chapter 6 is presented. The GCOP solver uses the rotational speeds $\overline{\omega}_1$, $\overline{\omega}_2$, $\overline{\omega}_3$ as input variables u_1, u_2, u_3. As we are looking at planar flight, zero out-of plane torque is assumed, and therefore the rotors which are not in the plane must have the same rotational speed, i.e. $\overline{\omega}_3 = \overline{\omega}_4$.

Without Drag

For the planar system without aerodynamic drag, the model is defined as

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{z} \\ \dot{\theta} \\ \ddot{x} \\ \ddot{z} \\ \ddot{\theta} \end{bmatrix} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} \dot{x} \\ \dot{z} \\ \dot{\theta} \\ -\frac{s\theta}{m} C_t (u_1^2 + u_2^2 + 2u_3^2) \\ -\frac{c\theta}{m} C_t (u_1^2 + u_2^2 + 2u_3^2) + g \\ \frac{C_t L}{J} (-u_1^2 + u_2^2) \end{bmatrix}, \quad \mathbf{x}(0) = \mathbf{x}_0, \quad \mathbf{x}(t_\mathrm{f}) = \mathbf{x}_\mathrm{f}, \tag{4-16}$$

where the constraints on the controls $\mathbf{u} = [u_1 \ u_2 \ u_3]^T$ are defined by

$$u_{\min} = 0 \, \mathrm{rad\,s^{-1}} \leq u \leq u_{\max} = 2513 \, \mathrm{rad\,s^{-1}} \, . \tag{4-17}$$

The minimum and the maximum rotational speeds are $u_{\min}$, $u_{\max}$.

With Drag

For the planar system with aerodynamic drag, the model is defined as

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{z} \\ \dot{\theta} \\ \ddot{x} \\ \ddot{z} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{z} \\ \dot{\theta} \\ -\frac{s\theta}{m} C_t(u_1^2 + u_2^2 + 2u_3^2) \\ -\frac{c\theta}{m} C_t(u_1^2 + u_2^2 + 2u_3^2) + g \\ \frac{C_t L}{J}(-u_1^2 + u_2^2) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\frac{1}{m}\mathbf{R}_{\text{bi}}^T \begin{pmatrix} C_x^B & 0 \\ 0 & C_z^B \end{pmatrix} \mathbf{R}_{\text{bi}} \begin{pmatrix} v_{\text{r},x} \\ v_{\text{r},z} \end{pmatrix} \\ 0 \end{bmatrix}, \tag{4-18}$$

with the boundary conditions

$$\mathbf{x}(0) = \mathbf{x}_0, \quad \mathbf{x}(t_\text{f}) = \mathbf{x}_\text{f}, \tag{4-19}$$

where the constraints on the controls $\mathbf{u} = [u_1\ u_2\ u_3]^T$ are defined by

$$u_{\min} = 0\,\text{rad}\,\text{s}^{-1} \leq u \leq u_{\max} = 2513\,\text{rad}\,\text{s}^{-1}\,. \tag{4-20}$$

The minimum and the maximum rotational speeds are $u_{\min}$, $u_{\max}$.

4-3-4 Modelling of Control Constraints

Electric motors, such as the DC motors used in the crazyflie quadcopter, are limited by a maximum rotational speed, after which they enter a saturated state. This behaviour is ideally modeled by static constraints on the rotor speed of each rotor.

In the first implementation, which was used for the validation in chapter 5, the GCOP solver used the total torque τ and thrust T acting on the COM of the rigid body as the control variables u_1, u_2. Therefore the constraints are dynamic,

$$\begin{bmatrix} T \\ \tau \end{bmatrix} = \begin{bmatrix} C_t & C_t & C_t & C_t \\ C_t L & -C_t L & 0 & 0 \end{bmatrix} \begin{bmatrix} \overline{\omega}_1^2 \\ \overline{\omega}_2^2 \\ \overline{\omega}_3^2 \\ \overline{\omega}_4^2 \end{bmatrix}, \text{ where } \begin{bmatrix} \overline{\omega}_{1,\min}^2 \\ \overline{\omega}_{2,\min}^2 \\ \overline{\omega}_{3,\min}^2 \\ \overline{\omega}_{4,\min}^2 \end{bmatrix} < \begin{bmatrix} \overline{\omega}_1^2 \\ \overline{\omega}_2^2 \\ \overline{\omega}_3^2 \\ \overline{\omega}_4^2 \end{bmatrix} < \begin{bmatrix} \overline{\omega}_{1,\max}^2 \\ \overline{\omega}_{2,\max}^2 \\ \overline{\omega}_{3,\max}^2 \\ \overline{\omega}_{4,\max}^2 \end{bmatrix}, \tag{4-21}$$

and cannot be modeled using the GCOP-solver. The reason is, that the GCOP-solver only allows for static constraints on the model control variables. Therefore, in chapter 5, no constraints are used.

To correctly model the dynamic constraints for chapter 6, we define the control variables u_1, u_2, u_3, u_4 as the individual rotational speeds $\overline{\omega}_1$, $\overline{\omega}_2$, $\overline{\omega}_3$ and $\overline{\omega}_4$. These control inputs are then limited to the electrical limit of the brushless motor, $\overline{\omega}_{\min} = 0\,\text{rad}\,\text{s}^{-1} \leq \overline{\omega}_\text{i} \leq \overline{\omega}_{\max} = 2513\,\text{rad}\,\text{s}^{-1}$, which was obtained from experimental data.

4-3-5 Normalization

In the current implementation, heuristic tuning is required for every change in the desired final state $\mathbf{x}_\mathrm{f}$ or the final time t_f, in order to ensure that the desired final state is reached. The reason is that the DDP algorithm (with the cost function defined in 4-3-1) makes a tradeoff between minimizing the control effort and the final state error. I. e., if the weight on a specific final state is not sufficiently high, the solver will find a trajectory which improves the control effort but deteriorates the final state error.

To avoid having to repeat this heuristic tuning of the weights, the state and control vectors are normalized. The weights are tuned once initially, and for each state variable and control input the maximum absolute value in the initial solution is used as the normalization constant. For states remaining close to zero (e.g. x in vertical flight), this would lead to a division by zero. Therefore, these states are not normalized.

For the integration of the model states the non-normalized states and inputs are required. Therefore the normalized data is firstly denormalized, the integration is performed and then the data is normalized again. In this way, the DDP algorithm works only with normalized data and all weights only have to be tuned once, see 4-1.

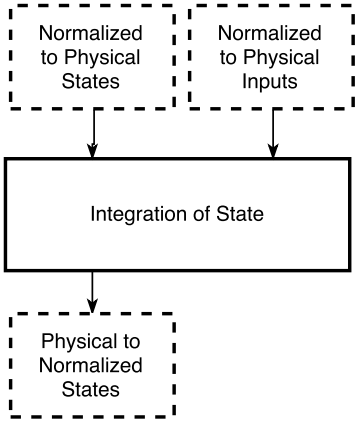


Figure 4-1: Normalization scheme for state integration, with normalization methods in dashed boxes.

In Figure 4-2 the convergence of the solver with normalized states and controls was compared to the convergence of the solver with non-normalized states and controls, for two different numbers of discrete steps $N = 20$ and $N = 50$. The solver with normalized states and controls consistently performs better with a much lower cost and faster convergence. The drawback of normalization is an increased computation time for the same number of iterations k and discretization size N, of approximately 9%. As the cost converges much faster and reaches a lower cost, this drawback can be accepted.

4-4 Solvers

The main solver is chosen with the intention of future deployment on robotic hardware. Therefore the solver library is required to be computationally efficient, ideally be written

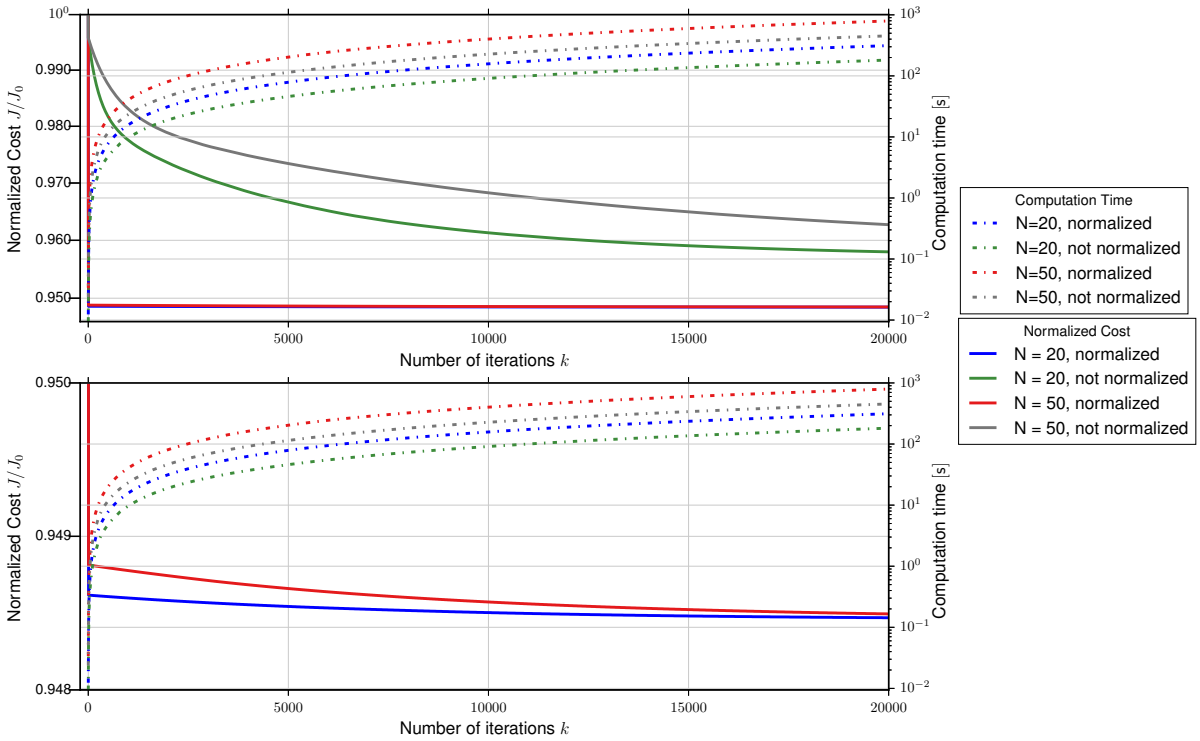


Figure 4-2: Normalized Cost J/J_0 plotted against iterations k and computation cost, with and without normalization of states and controls. The upper plot shows the entire cost-range, the bottom plot zooms into the curves for the normalized cases. When comparing the same discretization N, the costs in the normalized case are lower and converge after fewer iterations k.

in C++ and to be open-source. Furthermore, it has to be possible to define constraints on the control inputs, as control saturation should be modeled, as described in Section 4-3-4. As compiled in Table 4-1, the GCOP solver fulfills these requirements. In Section 4-4-1, an introduction to DDP and the GCOP package is given.

For validation of the main solver, a second solver is required. It should make use of a different optimization method for a meaningful validation. Ideally it is written in a language with high level of abstraction, allowing a time-efficient implementation. As this is merely the validation solver, the computation time required is irrelevant. For this purpose the GPOPS library was used. In Section 4-4-2, an introduction to Gauss Pseudospectral Method (GPM) and the GPOPS package is given. The validation is presented in

4-4-1 Differential Dynamic Programming

The Dynamic Programming (DP) method is an indirect method, i.e. the trajectory is not represented explicitly by the states, but is instead represented implicitly by the controls $\mathbf{u}(\mathbf{x}, i)$.

Table 4-1: Characterization of GCOP and GPOPS Optimization Methods

Solver Name	GCOP	GPOPS
Optimization Method	Differential Dynamic Programming	Gauss Pseudospectral Method
Programming Language	C++	Matlab
License	Open Source	Proprietary
Constraints on states	Not Possible	Possible
Constraints on input	Possible	Possible
Documentation	Not Available	Available

DP is based upon the *principle of optimality*. Introduced by Bellman, who transformed DP into a systematic tool (Bertsekas, 2001), the *principle of optimality* states the following intuitive fact.

Definition 6

(Principle of Optimality, (Bertsekas, 2001))

"Let $\pi^* = \{u_0^*, u_1^*, \ldots, u_{N-1}^*\}$ be an optimal policy for the basic problem, and assume that when using π^*, a given state x_i occurs at time i with positive probability. Consider the subproblem whereby we are at x_i at time i and wish to minimize the "cost-to-go"from time i to time N:

$$E\{g_N(x_n) + \sum_{k=i}^{N-1} g_k(x_k, u_k(x_k), w_k)\} \; . \tag{4-22}$$

Then the truncated policy $\{u_i^*, u_{i+1}^*, \ldots, u_{N-1}^*\}$ is optimal for this subproblem."

In other words, we can solve a large problem given the solutions to its smaller subproblems. This principle allows expressing the minimization over a sequence of controls as a sequence of minimizations over a single control (Tassa et al., 2014).

The DDP algorithm consists of two steps. Firstly, an optimization step backwards in time, receding from t_f to t_0 in discrete timesteps. Secondly, an integration step forwards in time. If the running cost for a certain time index and the cost on the following timestep is expressed as $l(\mathbf{x}, \mathbf{u})$ and $V'(\mathbf{f}(\mathbf{x}, \mathbf{u})))$, the cost-to-go $V(\mathbf{x})$ can be expressed as

$$V(\mathbf{x}) = \min_{\mathbf{u}}[l(\mathbf{x}, \mathbf{u}) + V'(\mathbf{f}(\mathbf{x}, \mathbf{u})))] \; . \tag{4-23}$$

DDP in the *Newton-Raphson* form is a method which recursively computes the (locally) optimal input(s) $\mathbf{u}(\mathbf{x}, i)$ resulting in the (locally) optimal cost-to-go function $V(\mathbf{x}, i)$, where $\mathbf{x}$ represents the state at discrete time t_i. If the change in cost $V(\mathbf{x})$ due to small perturbations $\delta\mathbf{u}, \delta\mathbf{x}$ is expressed as

$$Q(\delta\mathbf{x}, \delta\mathbf{u}) = l(\mathbf{x} + \delta\mathbf{x}, \mathbf{u} + \delta\mathbf{u}) + V'(\mathbf{x} + \delta\mathbf{x}, \mathbf{u} + \delta\mathbf{u}) \quad , \tag{4-24}$$

the algorithm computes the optimal control modification $\delta\mathbf{u}^*(\delta\mathbf{x})$ for some state perturbation $\delta\mathbf{x}$ with the linear feedback policy

$$\delta\mathbf{u}^*(\delta\mathbf{x}) = \min_{\delta\mathbf{u}} Q(\delta\mathbf{x}, \delta\mathbf{u}) = \mathbf{k} + \mathbf{K}\delta\mathbf{x} \quad , \quad \mathbf{k} = -Q_{\mathbf{uu}}^{-1} Q_{\mathbf{u}} \quad \text{and} \quad \mathbf{K} = -Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}} \; , \tag{4-25}$$

using a second-order Taylor series expansion of the change in cost Q (Tassa, Mansard, & Todorov, 2003).

The backward pass is followed by the forward pass, where the control policy is evaluated, with α the backtracking searching parameter

$$\hat{\mathbf{x}}_0 = \mathbf{x}_0 \quad , \tag{4-26}$$

$$\hat{\mathbf{u}}_i = \mathbf{u}_i + \alpha \mathbf{k}_i + \mathbf{K}_i(\hat{\mathbf{x}}_i - \mathbf{x}_i) \quad , \tag{4-27}$$

$$\hat{\mathbf{x}}_{i+1} = \mathbf{f}(\hat{\mathbf{x}}_i, \hat{\mathbf{u}}_i) \quad . \tag{4-28}$$

The backward and forward passes are iterated until convergence to the locally optimal trajectory.

The indirect representation employed in DDP has a large advantage in terms of computation time/complexity when compared to direct representations, as it solves N times the m-dimensional problem, instead of solving a single mN-dimensional problem, with complexities of $O(Nm^3)$ and $O(N^3m^3)$, respectively. Furthermore, this method converges quadratically (Murray & Yakowitz, 1984).

GCOP Software Package

GCOP (Kobilarov, 2016), is an open-source C++ framework for DDP, developed at the ASCO lab at John Hopkins University. It is used for optimal control, estimation and planning of dynamic systems and includes well-known optimization methods, amongst others DDP. Its implementation makes use of templates and is therefore flexible, while at the same time the structure is modular. This enables flexibility and freedom in the definition of new systems and optimization algorithms (Garimella, 2016). Its implementation in C++ makes it attractive for realtime implementations. It has been used successfully for many applications, such as ground vehicle trajectories, quadrotor trajectories, and for complex multibody dynamics, e.g. aerial manipulation (Garimella, 2016).

For the implementation of my quadrotor model, firstly a software interface to the framework had to be implemented in C++. A templated optimization loop was then written, containing amongst others a flatness-based method to compute an initial solution, a system model used for the state-integration and the normalization methods. Python was then used to create an interface to run the compiled C++ code and visualize and analyze the results.

The step-function computes from an initial state $\mathbf{x}_a$, input vector $\mathbf{u}_a$ and timestep h, the final state $\mathbf{x}_b$ and the final pitch angle θ_b.

The flatness-based method to create the full initial trajectory utilizes the relations for differential flatness (equations (3-35)-(3-41)) to compute all states and controls based on a polynomial trajectory. The generation of this initial polynomial trajectory is explained in appendix 8-1.

4-4-2 Pseudospectral Methods

Pseudospectral methods are a class of direct collocation and employ a direct transcription of a continuous-time optimal control problem to a Nonlinear Program (NLP) (Garg & Patterson, 2009). This NLP can then be solved using common software packages, such as Automatic Control and Dynamic Optimization Toolkit (ACADO) or Sparse Optimization Suite

(SOS) (Rao et al., 2010). They make use of global polynomials for the parametrization of the state and inputs, which are collocated with nodes from a Gaussian quadratures, i.e. the differential-algebraic equations are orthogonally or pseudospectrally collocated, providing spectral convergence (Rao et al., 2010).

GPOPS Software Package

Many pseudospectral methods have been described mathematically, but their algorithms are not available in the open literature in a form which can be readily implemented. Therefore most scientific users take advantage of open-source software or commercial off-the shelf (COTS) software, which are typically time-intensive to learn. Additionally, this approach has in many cases a low educational value, as one interacts with the software as a black box with no insight about the underlying methodology.

Out of this motivation GPOPS was developed (Rao et al., 2010). GPOPS is a software for solving multiple-phase optimal control problems, implemented in MATLAB. It is based on the Gauss pseudospectral method and has been developed at the Draper Laboratory, MIT. This method outputs an approximated optimal control solution. Firstly, the solver discretizes the differential equations and the objective function for each phase, then independent phases are connected using linkage conditions on state and time, as depicted in Figure 4-3.

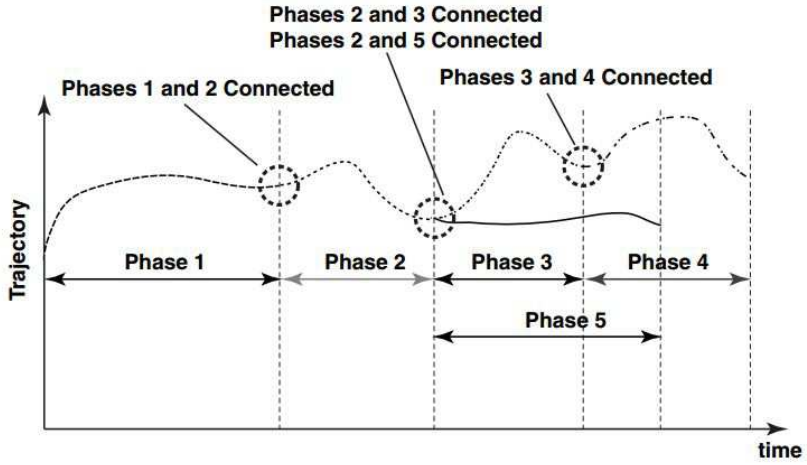


Figure 4-3: Linkages in multiple-phase pseudospectral control problem as employed in the GPOPS solver (Rao et al., 2010). In this schematic, the end of phases 1, 2 and 3 are linked to the start of the phases 2,3 and 4.

The user can choose whether the discretization is done either using finite differences or analytical derivatives. This results in a large nonlinear programming problem (NLP), which is solved using Automatic Control and Dynamic Optimization Toolkit (SNOPT) (P. Gill et al., 2005),(P. E. Gill, Murray, & Saunders, 2008). For our problem involving a point-to-point flight without external events, no linkage is needed and only one phase is solved for.

Chapter 5

Solver Validation

In chapter 4, the continous-time optimal control problem with its objective function, constraints and planar system model was defined. Also, both solvers were presented, GPOPS as the validation and GCOP as the main solver to be studied.

The next step is to test and validate the implementation of DDP in the GCOP library. In this chapter, we determine whether the trajectories generated with and without drag model are optimal and reliable. Furthermore, the performance of the solver is studied, focusing on the influence of the solver parameters on computation time needed to reach convergence. In the first section, the discrete formulation of the objective function is presented. In section 5-2, the solver is validated with the reference solver GPOPS and analytical methods. In section 5-3, the sensitivity of the required computation time to the choice of solver parameters is studied for a vertical, horizontal and diagonal test case. Results are compared for two types of discretization, equidistant and cosine discretization.

5-1 Discrete Objective Function

The objective of this work is to improve the flight performance of a quadcopter using a model-based optimal control solver. That is, the solver should compute a minimum-effort trajectory, minimizing the overall energy usage. For this purpose, the GCOP framework allows assigning a running cost on the square of the input variables. The final state should ideally be fixed. Nevertheless, in GCOP it is not possible to put a constraint on the final state $\mathbf{x}(N)$, see Table 4-1. To ensure that the solver reaches the desired final state, a cost term is implemented, which represents the deviation of the final state from the desired final state. This means that the total cost is defined, based upon the continuous-time definition in Section 4-3-1, as the sum of the final state error and the running cost on the square of the inputs. Therefore, this sum is minimized.

The objective function in discrete time is defined as

$$J(\mathbf{x}(i), \mathbf{u}_{i..N-1}, i) = [\mathbf{x}(N) - \mathbf{x}_\mathrm{f}]\mathbf{Q}_\mathrm{f}[\mathbf{x}(N) - \mathbf{x}_\mathrm{f}] + \sum_{k=i}^{N-1} \mathbf{u}(k)^T \mathbf{R}\mathbf{u}(k) \ , \tag{5-1}$$

where $\mathbf{x}(i) \in \mathbb{R}^6$ is the state at $t = i$ and $\mathbf{u}_{i..N-1} \in \mathbb{R}^2$ is the control policy from $t = i$ to $t = N - 1$. The desired final state is defined as $\mathbf{x}_\mathrm{f} \in \mathbb{R}^6$ with the corresponding weighting matrix $\mathbf{Q}_\mathrm{f} \in \mathbb{R}^{6\times6}$ and the control weighting matrix $\mathbf{R} \in \mathbb{R}^{2\times2}$.

It should be noted that the cost weights $\mathbf{Q}_\mathrm{f}$ and $\mathbf{R}$ are tuned heuristically in an iterative process, with the goal of minimizing the error on the final state and minimizing the control effort. This tuning is necessary, as if e.g. the cost weights on the squared inputs are too large with respect to the cost weights on the desired final state, the computed final position might have an offset to the desired final state. Based upon the tuning used in this chapter, the solver is later-on normalized for Chapter 6, as explained in Section 4-3-5.

The dynamic model defined in Section 4-3-2 requires definition of the drag constants C_x^B and C_z^B. At this point of time the experimental data for the identification of the drag constants was not yet available. Therefore, for the purpose of the validation it is sufficient to assume that they are different in x and z direction as

$$C_x^B = 1.0, \ C_z^B = 2.0 \ . \tag{5-2}$$

5-2 Benchmark Problems

To verify that the heuristically tuned solver is computing the minimum-effort solution, and that the model is correctly implemented, the GCOP solver is validated in this chapter with the reference solver GPOPS. This is done for the three benchmark problems of horizontal, diagonal and vertical flight for the model with and without drag. The validation is successful, if the three different solution methods (GCOP, GPOPS, analytical optimal solution) are able to reach the same global optimum.

The three test-cases are horizontal flight ($x_\mathrm{f} = 10m$, $z_\mathrm{f} = 0m$), vertical flight ($x_\mathrm{f} = 0$m, $z_\mathrm{f} = 10$m) and diagonal flight ($x_\mathrm{f} = 0$m, $z_\mathrm{f} = 0$m). The initial position is the origin ($x_0 = 0$m, $z_0 = 0$m) for all cases. All other boundary conditions are set to zero, such that e.g. the initial

and final rotational speeds $\dot{\theta}_0, \dot{\theta}_\mathrm{f}$ and rotation angles $\theta_0, \theta_\mathrm{f}$ are zero. Each test-case is assured to be feasible by ensuring the final time t_f is greater than or equal to the final time from the minimum-time solution obtained from GPOPS. For vertical flight, the solver is additionally validated with the solution from optimal control theory, which is derived analytically.

5-2-1 Optimality of Solution

In order to validate the results from the GCOP and GPOPS solver, firstly a benchmark problem for vertical flight in the z direction is analyzed on the unit-model. This problem is especially useful, as the quadcopter dynamics can be simplified to a simple linear model, for which the optimal solution can be derived analytically.

GCOP and GPOPS Solutions

First, the initial and final state in both the GCOP and GPOPS solver are set to

$$\mathbf{x}_0 = \begin{bmatrix} x_0 \\ z_0 \\ \theta_0 \\ \dot{x}_0 \\ \dot{z}_0 \\ \dot{\theta}_0 \\ \ddot{x}_0 \\ \ddot{z}_0 \\ \ddot{\theta}_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{x}_\mathrm{f} = \begin{bmatrix} x_\mathrm{f} \\ z_\mathrm{f} \\ \theta_\mathrm{f} \\ \dot{x}_\mathrm{f} \\ \dot{z}_\mathrm{f} \\ \dot{\theta}_\mathrm{f} \\ \ddot{x}_\mathrm{f} \\ \ddot{z}_\mathrm{f} \\ \ddot{\theta}_\mathrm{f} \end{bmatrix} = \begin{bmatrix} 0 \\ -10 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad . \tag{5-3}$$

The weighting matrix $\mathbf{Q}_\mathrm{f}$ on the deviation from the desired final state and the weighting matrix $\mathbf{R}$ on the control inputs are defined, whereas GPOPS only requires the latter,

$$\mathbf{Q}_\mathrm{f} = \begin{bmatrix} Q_{\mathrm{f},x} \\ Q_{\mathrm{f},\dot{x}} \\ Q_{\mathrm{f},z} \\ Q_{\mathrm{f},\dot{z}} \\ Q_{\mathrm{f},\theta} \\ Q_{\mathrm{f},\dot{\theta}} \end{bmatrix} = \mathbf{I}, \quad \mathbf{R} = \begin{bmatrix} R_{u_1} \\ R_{u_2} \end{bmatrix} = \mathbf{I} \quad . \tag{5-4}$$

The problem for a final time of $t_f = 4\,\mathrm{s}$ is discretized in GCOP with $N = 50$ equidistant steps and solved using $k = 50$ iterations. GPOPS does not require a time discretization. The GCOP and GPOPS trajectories for a simulation time of $t_\mathrm{f} = 4.0$ s are shown in Figure 5-1. The solution obtained with GCOP does not converge to the desired final position $z = -10\,\mathrm{m\,s^{-1}}$ and velocity $\dot{z} = 0\,\mathrm{m\,s^{-1}}$, whereas the GPOPS trajectory reaches the final state. The reason for this is that the objective function in GCOP is not formulated as a pure minimum control-effort problem, but is instead formulated as the sum of the control effort and the final state error. Therefore, for a meaningful validation, this final state error in GCOP must be corrected by retuning the final state weights.

Analytical Solution

The optimal solution for the vertical case is obtained analytically, since the planar model, presented in Section 3-2, can be considered to be linear, if purely vertical motion is assumed, i.e. θ, x and their derivatives are zero. Therefore, if only vertical motions are considered, the planar system dynamics defined in Equation 4-16 simplifies to the linear system

$$\ddot{z} = g - u \,, \tag{5-5}$$

where u is the input, z the vertical position and g the standard gravity. The model is subject to the boundary conditions

$$\int_0^{t_\mathrm{f}} \dot{z}(t)\mathrm{dt} = z_\mathrm{f} - z_0 \,,\ \dot{z}_0 = \dot{z}_\mathrm{f} = z_0 = 0 \,,\ z_\mathrm{f} = 10\ \mathrm{m\,s^{-1}} \,, \tag{5-6}$$

where t_f is the simulation end time. Given the performance index $\int_0^{t_\mathrm{f}} u^2(t)\mathrm{dt}$, the optimal trajectory is a cubic polynomial for the position $z(t)$, a quadratic polynomial for the velocity $\dot{z}(t)$ and a linear polynomial for the acceleration $a = \ddot{z}(t)$ (Bruno Siciliano, 2010).

The analytical solution shown in Figure 5-1 coincides with the GPOPS solution, but shows differences to the poorly tuned GCOP solution. The flatness-based polynomial initialization trajectory is also shown in Figure 5-1. For details on the trajectory initialization method, see Appendix 8-1.

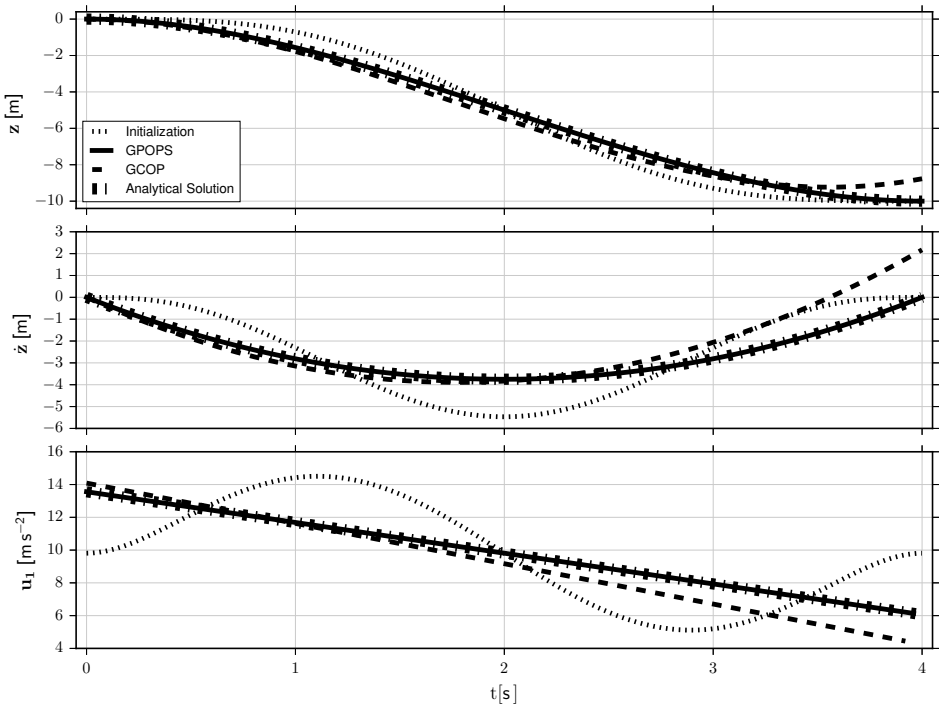


Figure 5-1: Solution for vertical flight with unit weights on $\mathbf{Q}_\mathrm{f}$ and $\mathbf{R}$. The analytical solution and the GCOP initialization are compared to the solutions from GCOP and GPOPS solver. The solution from GCOP doesn't converge to desired final state.

Retuning of GCOP and GPOPS parameters

In order to help GCOP to reach the desired final state for the vertical flight problem, the weights $\mathbf{Q}_\mathrm{f}$ and $\mathbf{R}$ are retuned, increasing $Q_{\mathrm{f},z}$ and $Q_{\mathrm{f},\dot{z}}$. This results in the weights matrices $\mathbf{Q}_\mathrm{f}$ and $\mathbf{R}$,

$$\mathbf{Q}_f = \begin{bmatrix} Q_{\mathrm{f},x} \\ Q_{\mathrm{f},\dot{x}} \\ Q_{\mathrm{f},z} \\ Q_{\mathrm{f},\dot{z}} \\ Q_{\mathrm{f},\theta} \\ Q_{\mathrm{f},\dot{\theta}} \end{bmatrix} = \begin{bmatrix} 0.02 \\ 0.02 \\ 0.2 \\ 1 \\ 0.02 \\ 0.02 \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} R_{u_1} \\ R_{u_2} \end{bmatrix} = 0.02 \times \mathbf{I} \quad . \tag{5-7}$$

After retuning the cost weights, the trajectories computed with GCOP, GPOPS and the analytical solution coincide, and the three solutions reach the desired final states, see Fig. 5-2. This validates the GCOP solver and proofs its ability to reach the minimum control effort solution.

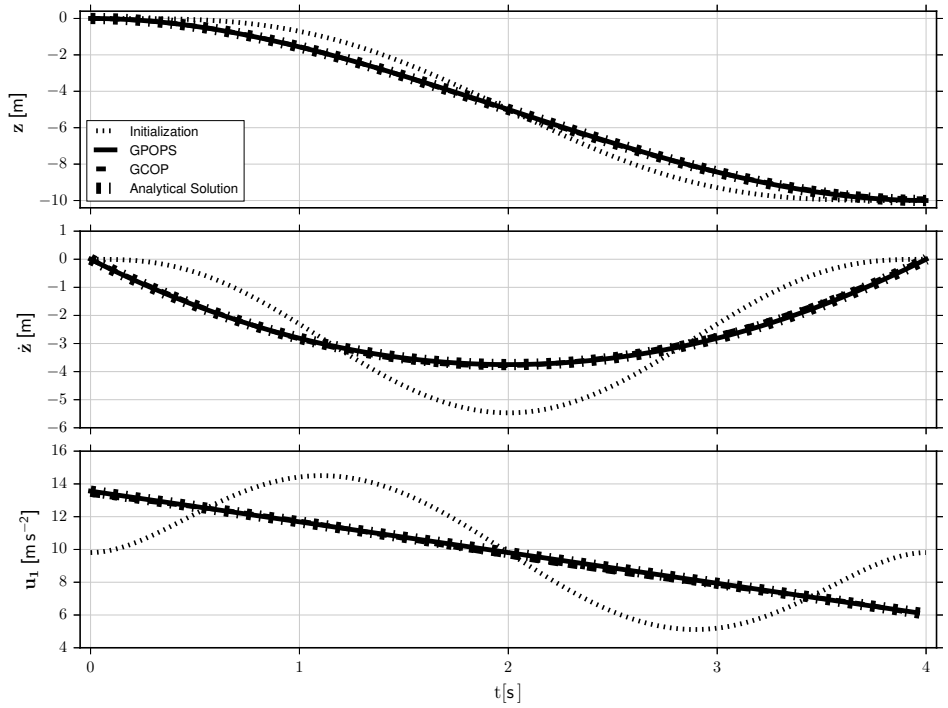


Figure 5-2: Solution for vertical flight. The analytical solution and the GCOP initialization are compared to solutions from GCOP and GPOPS solver. The weights on $\mathbf{Q}_f$ and $\mathbf{R}$ were determined heuristically by trial-and-error, such that the GCOP solution reaches the desired final state. This validates the solvers.

5-2-2 Validation of Full Model

The GCOP and GPOPS trajectories are generated for the full planar model (as defined in Section 4-3-3) with and without drag. The cost weights defined in equation 5-7 are used. The differences in the solutions from both methods are analyzed qualitatively and quantitatively.

Vertical Flight

In Figure 5-3, the resulting trajectories are shown for the vertical flight with and without drag. As expected, the horizontal position x, velocity $\dot{x}$, pitch θ and rotational speed $\dot{\theta}$ are all close to zero, both with and without drag model.

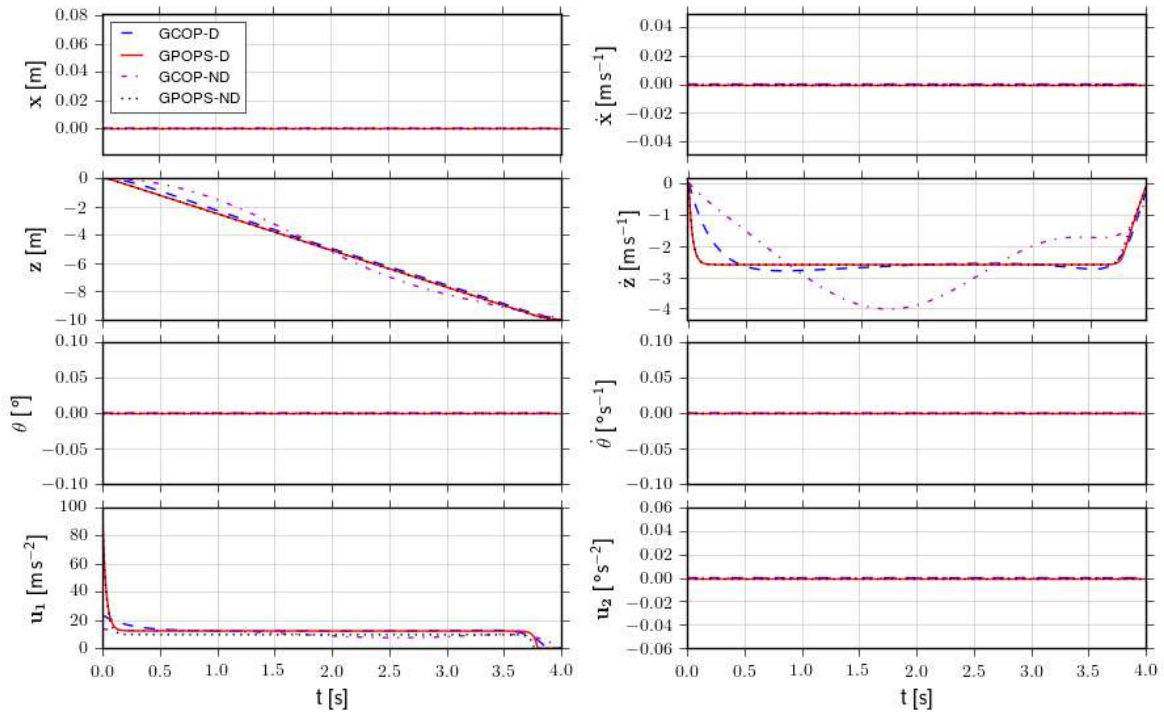


Figure 5-3: Validation trajectories for the vertical flight problem without (ND) and with (D) drag model for both solvers.

The vertical velocity $\dot{z}$ reaches for the GPOPS-solver with and without drag and the GCOP solver with drag a maximum value of $|\dot{z}| = 2.5\ \mathrm{m\,s^{-1}}$. The GCOP solution without drag contains a large oscillation with a maximum vertical speed of $|\dot{z}| = 4.0\ \mathrm{m\,s^{-1}}$. It can be concluded that including the drag model damps and stabilizes the trajectory. For the vertical position z, the solution from GPOPS with and without drag and the curve from GCOP with drag model are very similar. The result from GCOP without drag model shows an oscillation in z.

In contrast to Section 5-2-1, it becomes apparent that the GCOP and GPOPS trajectories differ. As the model changed, the weights matrices $\mathbf{Q}_f$ and $\mathbf{R}$ would require to be retuned in order to obtain the same trajectories. It can be concluded that if in GCOP the final-state error weights $\mathbf{Q}_f$ are too large with respect to the running input costs weights $\mathbf{R}$ the solutions might differ from the global optimum minimum control effort solution.

State and Control Error

In order to analyze the state and control errors between both solvers quantitatively, an error metric needs to be introduced. The Root Mean Square (RMS)

$$\text{RMS} = \sqrt{\sum(\hat{\kappa} - \kappa)^2/N} \tag{5-8}$$

and the Normalized Root Mean Square (NRMS)

$$\text{NRMS} = \frac{RMS}{|\hat{\kappa}_{\max} - \hat{\kappa}_{\min}|} \tag{5-9}$$

are introduced as a measure for the quality of fit, where $\hat{\kappa}$ and κ represent the estimated value and the reference value, respectively. They represent a discrete time series for one trajectory-variable, of length N. $\hat{\kappa}$ represents the GCOP , and κ the GPOPS time series.

The normalization is done with respect to the range, which is the absolute difference between the maximum $\hat{\kappa}_{\max}$ and minimum $\hat{\kappa}_{\min}$ from the GCOP time series. This normalization is done to make the results comparable, but it has the drawback that for values with small range (e.g. values close to zero) the division turns into a division by zero and the NRMS value explodes to a very large number. The results of the analysis are presented in table 5-1 for the three cases.

Table 5-1: NRMS in percentage for GCOP and GPOPS model, for the horizontal, vertical and diagonal flight problem

Flight Problem	Drag Model	$\mathbf{x}$	$\mathbf{z}$	$\boldsymbol{\theta}$	$\dot{\mathbf{x}}$	$\dot{\mathbf{z}}$	$\dot{\boldsymbol{\theta}}$	$\mathbf{u_1}$	$\mathbf{u_2}$
Horizontal	No	0.8	16	1.6	1.1	7.8	6.2	5.0	7.0
	Yes	1.0	21	1.5	1.4	7.2	5.2	2.6	6
Vertical	No	-	2.3	-	-	10	-	18	-
	Yes	-	2.1	-	-	9.1	-	16	-
Diagonal	No	21	37	3.0	4.3	16	6.5	16	12
	Yes	1.1	1.2	2.2	3.4	12	5.5	16	5.9

For the horizontal flight, the NRMS values of the errors for the states are small in the range between 0.8% and 21%. The largest values are found for the vertical position z, as this variable has a small range and a relatively high RMS. For the vertical flight, the NRMS values of the errors lie between 2.1% and 18%. As the fit is very good, the horizontal position x, velocity $\dot{x}$, the pitch angle θ, the rotational velocity $\dot{\theta}$ and the torque u_2 yield an "inf" value, designated in Table 5-1 as "-". The NRMS values of the errors for the diagonal flight case lie between 1.1% and 37%, where the highest value can be observed for z.

Cost Error

The difference in the total cost of the GCOP and the GPOPS solution can be used as a measure for the similarity of the obtained solutions. The values for the three test cases are collected in table 5-2, showing that the cost errors are very low for the cases with the drag model. For the case without drag model, they are slightly higher, with the highest error of approx. seven percent for the diagonal flight without the drag model.

Table 5-2: Error in Cost between GCOP and GPOPS for the horizontal, vertical and diagonal flight problem, expressed as percentage.

Flight Problem	Without drag model	With drag model
Horizontal	6.7 %	0.6 %
Vertical	0.3 %	0.5 %
Diagonal	7.2 %	3.0 %

5-3 Parameter Analysis

In a future implementation of the algorithm on the quadrotor hardware, the discretization size N and the number of iterations k will need to be specified. To make a reasonable choice of these solver parameters, it is fundamental to study the sensitivity of the required computation time to the choice of solver parameters. Furthermore, it is important to understand the behavior of the solver, e.g. which choice of parameters might prevent the solver reaching the global optimum.

Initially, the idea is to fix the discretization N and stop the solver iterations k, once a certain cost gradient is obtained. One disadvantage of this method is that the computation time is difficult to predict, as the gradient and therefore the threshold gradient are sensitive to the cost weights. The second disadvantage is that normalized cost contains steps and therefore its gradient is not continuous, as highlighted in Fig. 5-4. These disadvantages make a practical implementation of this method difficult and therefore this idea is discarded.

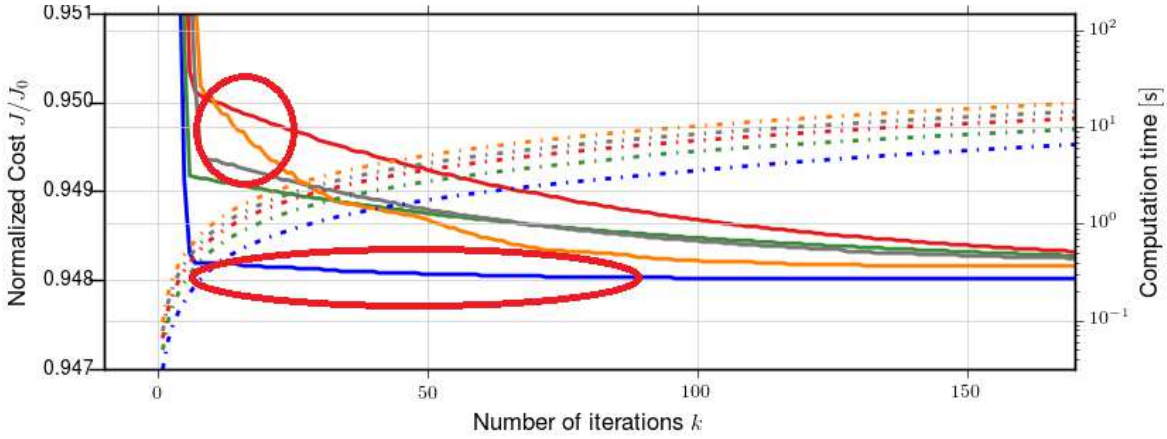


Figure 5-4: Example cost function showing step phenomena in normalized cost.

Instead it is decided to define the discretization size N and the number of iterations k upfront and keep them constant. Nevertheless a fixed discretization size and number of iterations will only yield optimal results for a specific problem formulation. These parameters might need to be redefined, if the problem definition changes, e.g. simulation end time or the quadcopter model. To fix these parameters, it is needed to know how they influence the computation time and the trajectory cost; if any configuration leads to a local minimum it should be avoided. Nevertheless, the choice is typically a tradeoff between quality of the solution and computation time.

The solver performance was analyzed for the three test cases and for two discretization methods. The solver is run for discretizations from $N = 5, \ldots, 400$ and $k = 1, \ldots, 700$ iterations. For the sake of brevity, in this section only the analysis of the solver performance for the vertical case is presented, as its results are representative for the three test cases. The algorithm is executed on a Unix-machine with an Intel-Xeon W3530, 2.8 GHz central processing unit (CPU) and 5GB RAM memory.

In section 5-3-1, an initial guess of the optimal parameter is made based upon the normalized cost and computation times for different N and k. The trajectory corresponding to the initial

guess is then verified. In section 5-3-2, the sensitivity to k is studied, whereas in section 5-3-3 the sensitivity to discretization N is studied. Finally, in section 5-3-4 the performance of an alternative discretization method is studied.

5-3-1 Normalized Cost Function

In Fig. 5-5-(a) the normalized costs and computation times are presented for all discretizations N and number of iterations k which were tested. Fig. 5-5-(b) shows the normalized costs for a smaller range of iterations $k = 1, ..., 120$. The discretizations $N = 10$ and $N = 5$ result in a larger cost J/J_0 than $N = 20$. This is caused by discrete timesteps Δt which are too large for an accurate integration of the states by the solver, making the trajectories inaccurate. For $N = 20$, the minimum cost is obtained at $k = 50$, which corresponds to a computation time of $T = 0.41$ s. For a larger discretization, the required computation time T becomes larger, e.g. at $N = 50$, the cost converges at $k = 100$ to the minimum cost, which corresponds to a computation time of $T = 1.12$ s, which is 2.7 times the computation time required for the discretization $N = 20$.

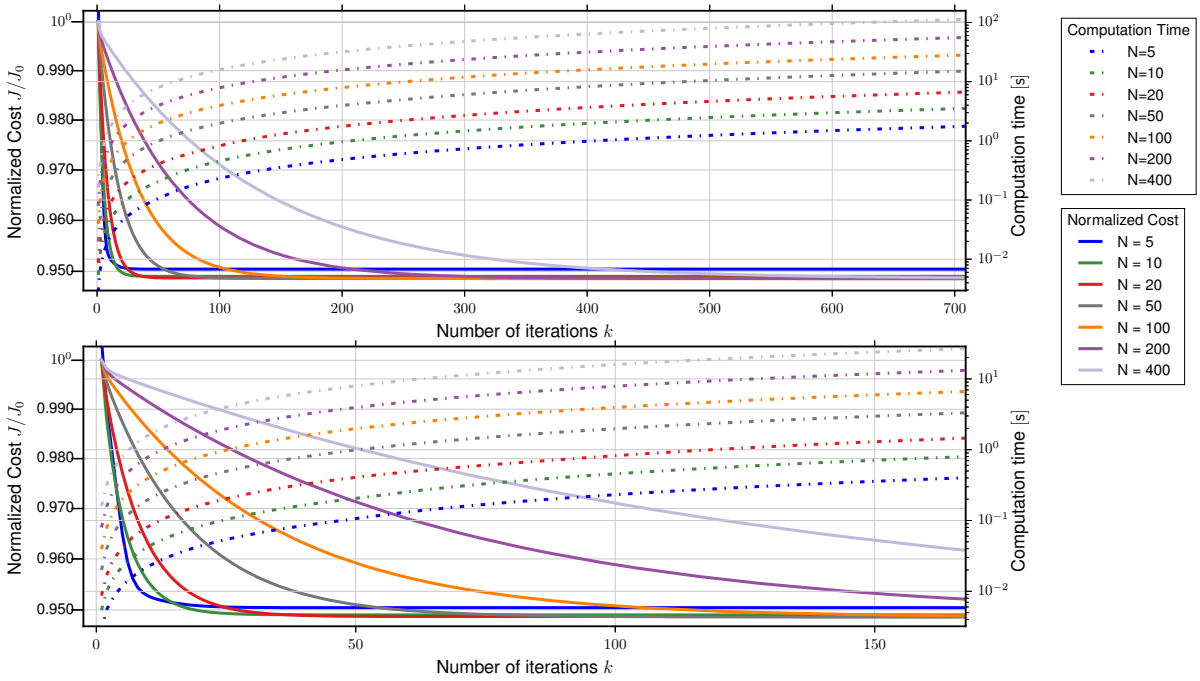


Figure 5-5: Normalized Cost J/J_0 and computation time vs number of iterations k for vertical Case with non-equal weights $\mathbf{Q}_f$ and $\mathbf{R}$ and uniform dicretizations N, (a) up to $k = 700$ iterations, (b) up to $k = 170$ iterations. The discretization $N = 5$ reaches a local minimum. The selected configuration, discretization $N = 20$ using $k = 50$ iterations has a normalized cost of $J/J_0 = 0.9487$.

In order to verify the solution, the trajectory for the chosen parameters $N = 20$ and $k = 50$ is plotted in Fig. 5-6, together with other combinations of N and k which result in the

same computation time. The chosen parameters result in the lowest cost. If N is increased (and k decreased), the solver has not yet reached the optimum solution and shows therefore oscillations. If N is decreased (and k increased) the solution is still optimal, but has a timestep $\Delta t < 0.2\,\text{s}$, which is to large to accurately integrate the dynamics, resulting in inaccurate trajectories.

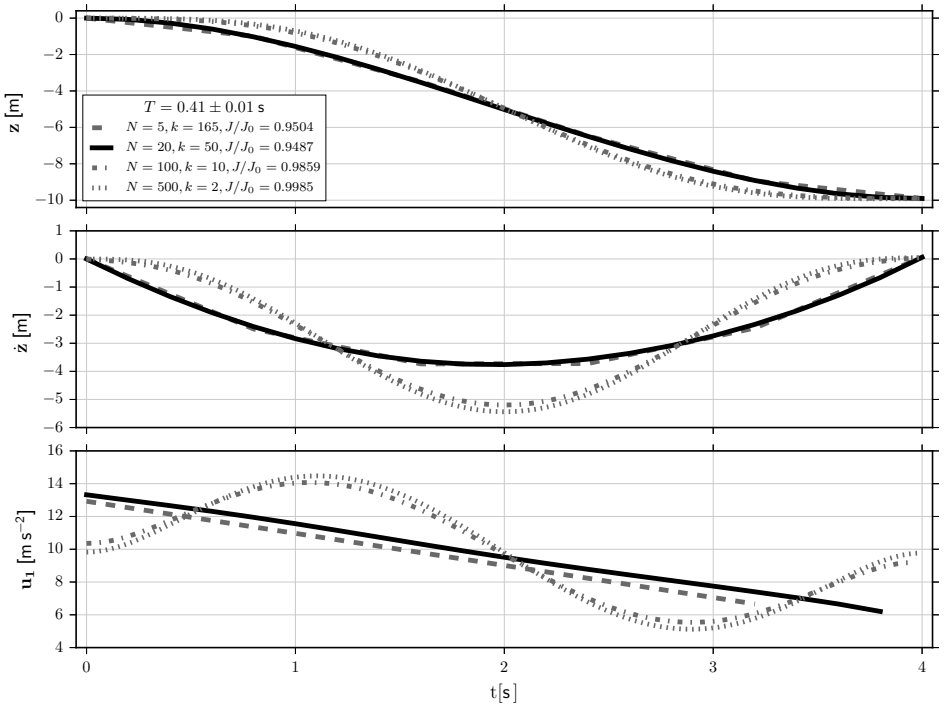


Figure 5-6: Solutions to vertical case with non-equal weights for a computation time $T = 0.41 \pm 0.01$ s. Discretization $N = 5$ to $N = 500$, iterations k and normalized cost J/J_0. The discretization $N = 20$ has the lowest cost, and is therefore the best configuration for a configuration of $T = 0.41 \pm 0.01$ s.

The chosen parameters, a discretization of $N = 20$ and $k = 50$ number of iterations result in the lowest cost J/J_0 and yield a trajectory, which is close to the analytical solution.

5-3-2 Iterations

In Fig. 5-7, the sensitivity to k is studied.$N = 20$ is fixed and the number of iterations is decreased in steps of 15 from $k = 65$ to $k = 20$.

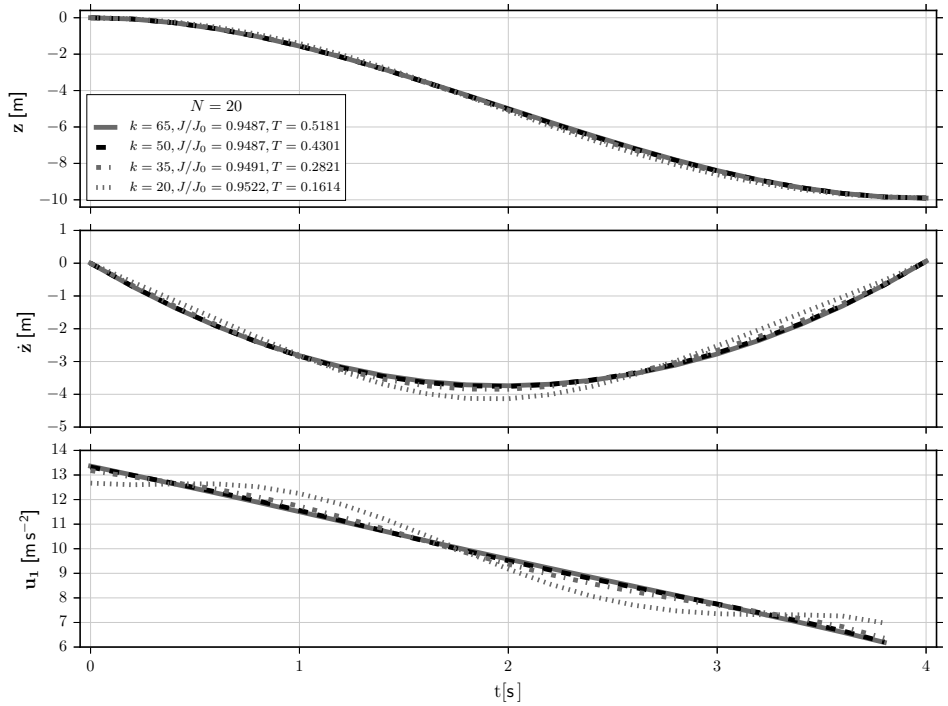


Figure 5-7: Solutions for vertical case, for a discretization of $N = 20$ and numbers of iterations $k = 20$ to $k = 65$. In terms of normalized cost J/J_0, $k = 50$ and $k = 65$ are the best configurations, whereby $k = 50$ has a lower computation time.

For iterations $k < 50$, the quality of the solution decreases, i.e. the solution is less optimal and oscillations occur. The computation time T is proportional to k. If we increase the iterations to $k = 65$, the quality and cost stays the same, while the computation time increases. Therefore, $k = 50$ iterations are confirmed as the best solution for the discretization $N = 20$.

5-3-3 Discretization Size

The sensitivity to the discretization N is studied in Fig.5-8. The iterations is fixed as $k = 50$ and the discretization N is varied.

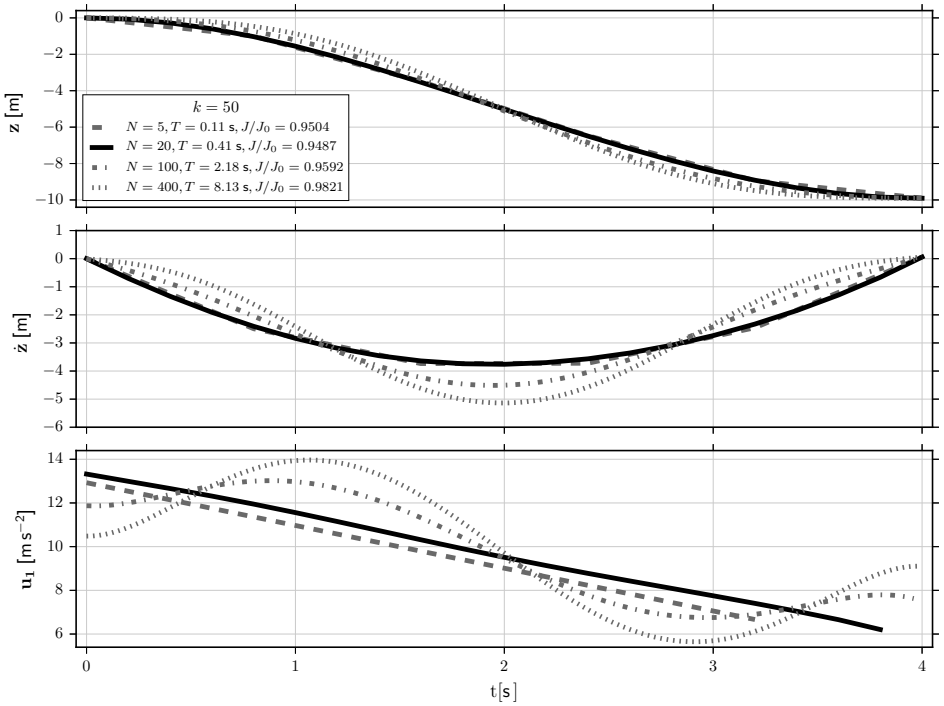


Figure 5-8: Solutions to vertical case with non-equal weights on $\mathbf{Q}_\mathrm{f}$ and R for $k = 50$ iterations. Discretization $N = 5$ to $N = 400$, computation time T and normalized cost J/J_0. The discretization $N = 20$ has the lowest cost.

For discretizations $N > 20$, e.g. $N = 100$, the states oscillate with respect to the analytical solution shown in 5-2. These curves would require more iterations k to reach the analytical solution. As previously mentioned, for discretization $N < 20$ with corresponding timestep $\Delta t > 0.2\,\mathrm{s}$ the normalized cost decreases slightly due to inaccurate state integration by the solver. Therefore, solutions with $N < 20$ should be discarded.

5-3-4 Discretization Type

For minimum-control effort problems, fast dynamics are typically required close to $t = t_0$ and $t = t_f$. In an effort to improve the performance of GCOP, the density of discretization points is increased at the edges of the timeframe, while mantaining the same discretization size N, using the simple relation

$$t(i) = \frac{1}{2} t_\mathrm{f} \big(\cos(\pi - \frac{i}{N}\pi) + 1 \big), \quad i = [1, ..., N] \quad . \tag{5-10}$$

In Fig. 5-9, the normalized cost J/J_0 and computation time T is shown for the cosine-discretization, in analog to the results presented in Figure 5-5 for the equidistant discretiza-

tion. Fig. 5-9-(a) shows the results for all combinations of solver parameters, whereas Fig. 5-9-(b) zooms into the range up to $k = 170$ iterations.

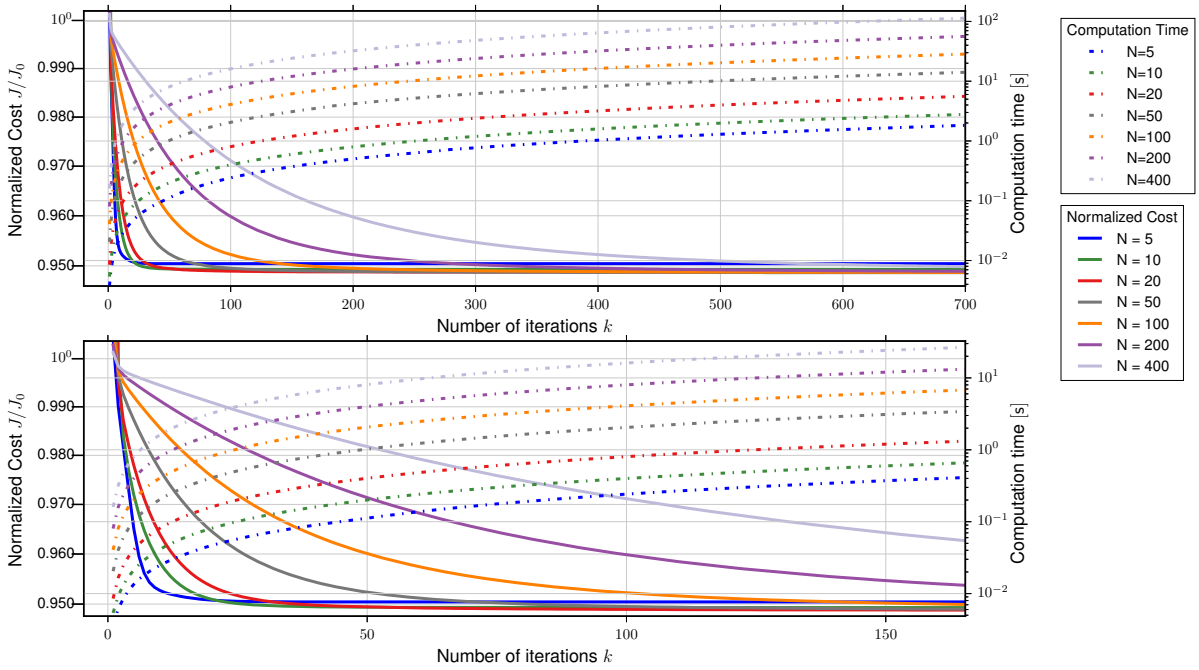


Figure 5-9: Normalized Cost J/J_0 and computation time T vs number of iterations k for vertical case with cosine discretization and non-equal weights on $\mathbf{Q}_\mathrm{f}$ and $\mathbf{R}$. The iterations are shown up to (a) $k = 700$ and (b) $k = 170$. The discretization $N = 5$ reaches a local minimum.

Using this alternative discretization scheme, the normalized cost is larger for the discretization $N < 20$ then for $N = 20$, similar than with the equidistant discretization. In order to verify the solution, the trajectory for the chosen parameters $N = 20$ and $k = 50$ is plotted in Fig. 5-10, together with the solutions for other k.

Using the cosine-discretization leads to a solution, where the input u_1 doesn't coincide with the analytical solution, it has a "step" at the beginning and at the end, Fig. 5-10. The reason for the change in slope at the edges is that the algorithm has difficulties finding the optimal solution which is not only locally optimal. If we decrease the number of discretization points at the same number of iterations k we have less points at the edges, which fixes this problem, it approaches the analytical solution and the "step"-phenomenon disappears. For a higher number of discretization points N, the solution deteriorates. It can be generally concluded that increasing the density of the discretization points locally does not improve solver performance, as the number of iterations k required to reach the optimal solution always depends on the smallest timestep in the complete time-vector. Furthermore, a minimal cost is not necessarily an indicator for reaching the optimal solution.

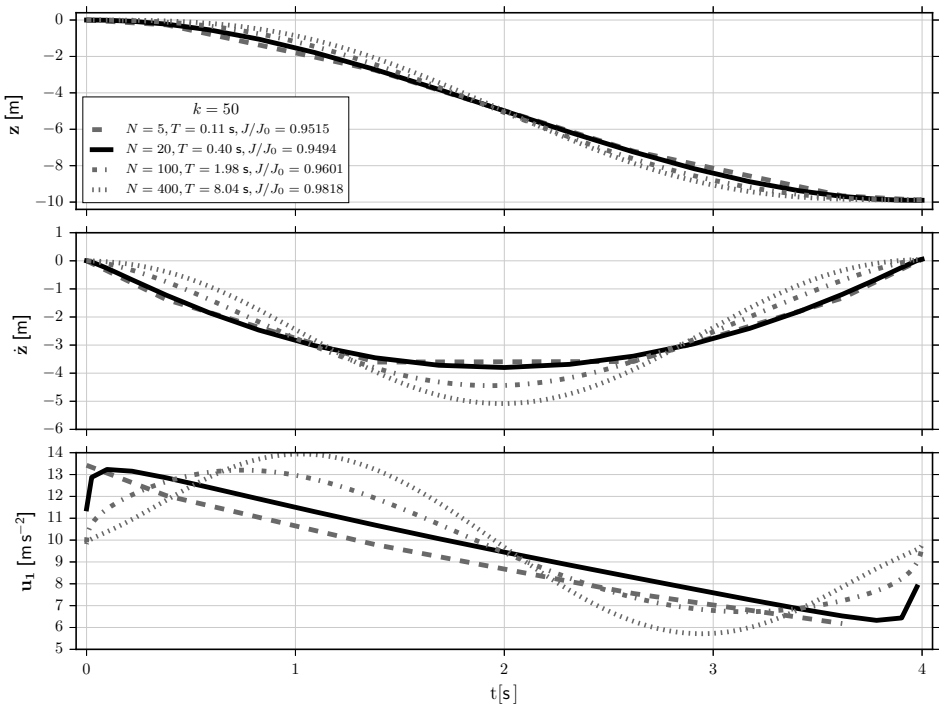


Figure 5-10: Solutions to vertical case with cosine discretization and non-equal weights on $\mathbf{Q}_{\mathrm{f}}$ and $\mathbf{R}$ using $k = 50$ iterations. Discretization $N = 5$ to $N = 400$, computation time T and normalized cost J/J_0. The discretization $N = 20$ is here the best discretization in terms of normalized cost, nevertheless it hasn't converged to the optimal solution.

Chapter 6

Results

In chapter 5 it was shown that the implementation of DDP in the GCOP library provides reliable, optimal trajectories for a planar quadcopter model with and without aerodynamic drag model. The next step is to study in simulation how the flight performance of a quadrotor can be improved in a feedforward manner by exploiting a DDP trajectory generated with an aerodynamics model. For this purpose, 4 different controller schemes are proposed and studied.

In this chapter, firstly the GCOP solver setup is defined. Then the simulation experiment is presented in section 6-1-2. In section 6-1-3, the experimentally obtained thrust and drag models are introduced. In Section 6-2 a controller is developed, which exploits the aerodynamic model and wind information while being robust to modelling errors.

The simulation experiments were performed for a horizontal, vertical and diagonal test case. In order to account for real world model error and noise, a Monte Carlo analysis was done, introducing random perturbations on the quadcopter mass and moment of inertia constant.

6-1 Setup

For the following analysis, the crazyflie nano quadcopter platform is taken as the quadrotor model, see Figure 6-1. It is a miniature quadrotor with a total mass of 29.6g (incl. batteries, which account for 7.3g), and a motor-to-motor distance of 9cm. This quadcopter is powered by 4 brushless engines. Three test cases are analyzed, the horizontal ($x_f = 4\,\mathrm{m}$, $z_f = 0\,\mathrm{m}$), vertical ($x_f = 0\,\mathrm{m}$, $z_f = -4\,\mathrm{m}$) and diagonal test case ($x_f = 4\,\mathrm{m}$, $z_f = -4\,\mathrm{m}$).

Section 6-1-1 explains the GCOP solver setup, section 6-1-2 the Simulink model and the Monte-Carlo Simulation procedure. This is followed by the parameter identification of the aerodynamic thrust coefficient and drag constants in section 6-1-3. In section 6-1-4, the equations to estimate the trajectory energy and power are introduced.

Figure 6-1: Crazyflie nano quadcopter used in this work. The gray reflective pellets are used for infrared tracking with the Vicon-system.

6-1-1 Solver Setup

The trajectories generated in GCOP are the reference trajectories for the controller. They are generated using the cost-function presented in section 5-1. The normalization of the states and control inputs has been implemented in the GCOP solver (as explained in section 4-3-5), making it possible to define only once the cost weights for a specific problem and not having to redefine the cost weights when the problem changes. Therefore, after the normalization is implemented, heuristic corrections are not necessary anymore in order to obtain convergence to the desired final state, whenever the problem is redefined. The three different test cases use therefore the same heuristically found control cost weight matrix $\mathbf{R} = \text{diag}(50.0)$ and cost weight matrix on the final state deviation $\mathbf{Q}_\text{f} = \text{diag}(Q_{\text{f},x}\ Q_{\text{f},\dot{x}}\ Q_{\text{f},\ddot{x}}\ Q_{\text{f},z}\ Q_{\text{f},\dot{z}}\ Q_{\text{f},\ddot{z}}\ Q_{\text{f},\theta}\ Q_{\text{f},\dot{\theta}}\ Q_{\text{f},\ddot{\theta}})$, see Table 6-1.

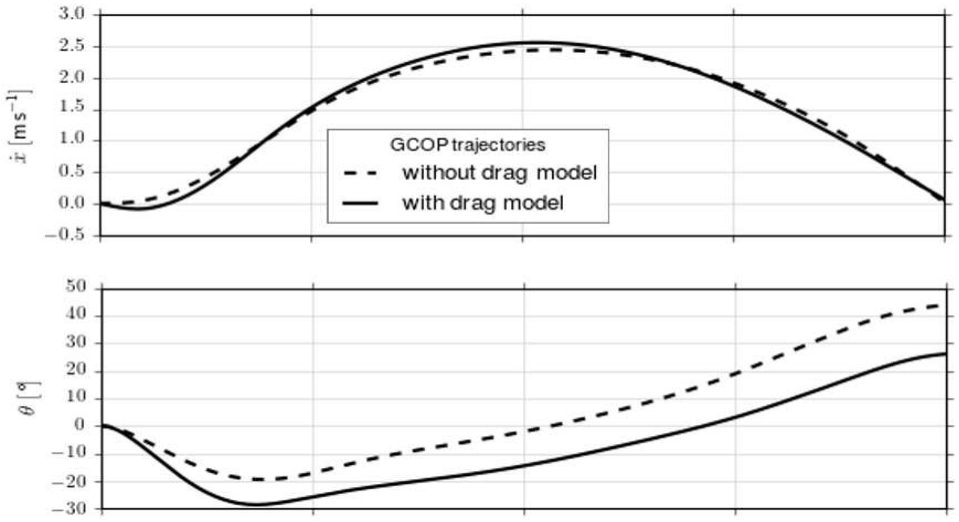
Table 6-1: GCOP cost weights used for the results in chapter 6

$Q_{\text{f},x}$	$Q_{\text{f},\dot{x}}$	$Q_{\text{f},\ddot{x}}$	$Q_{\text{f},z}$	$Q_{\text{f},\dot{z}}$	$Q_{\text{f},\ddot{z}}$	$Q_{\text{f},\theta}$	$Q_{\text{f},\dot{\theta}}$	$Q_{\text{f},\ddot{\theta}}$
1200	500	10	1800	900	10	0	0.6	0.1

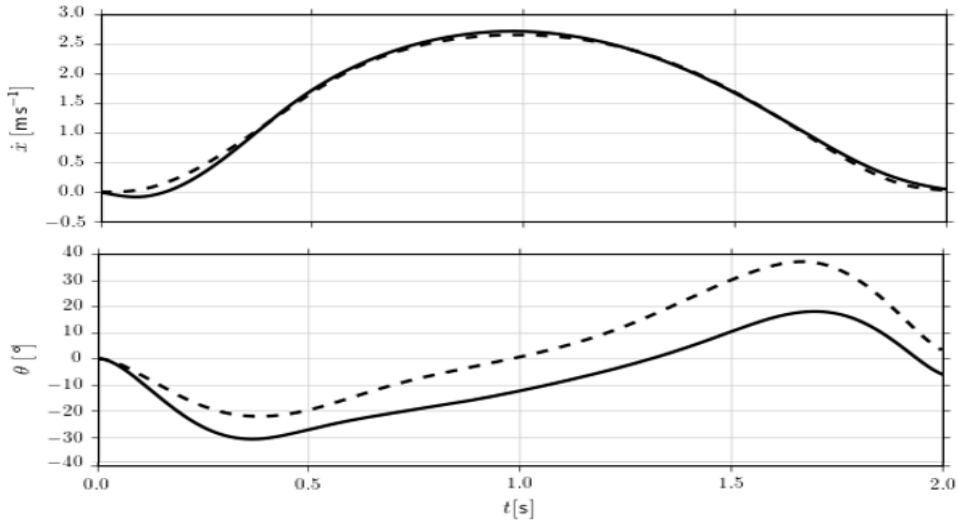
Analysis of the GCOP results showed, that if the flight time is not significantly larger then the minimum time which is feasible for reaching the desired final state (minimum time obtained by solving minimum-time problem with GPOPS), then setting the cost weights on the second derivatives $Q_{\text{f},\ddot{x}}, Q_{\text{f},\ddot{z}}, Q_{\text{f},\ddot{\theta}}$ to zero will result in non-stationary final states. The corresponding trajectory is shown in Fig. 6-2-(a). In other words, if a stationary final state cannot be achieved in the defined t_f and $Q_{\text{f},\ddot{x}}, Q_{\text{f},\ddot{z}}, Q_{\text{f},\ddot{\theta}}$ are zero, the solver will end up in a non-stationary final state with a nonzero pitch angle θ.

This problem is solved by setting a nonzero cost weight on the second-order derivatives. On Figure 6-2-(b) it can be seen that the solution then improves. Both the pitch angle θ and

acceleration in x, variables related through $\ddot{x} = \frac{T}{m}\sin(-\theta) + f_{d,x}$, are then at the final state close to zero.



(a) GCOP trajectories for zero cost weights on the final state error of the second derivatives.



(b) GCOP trajectories for nonzero cost weights on the final state error of the second derivatives.

Figure 6-2: Stationary final state is obtained by setting weights on final state deviation error for second derivatives to nonzero.

6-1-2 Simulink Model

The structure of the continuous-time Simulink Model is shown in Figure 6-3.

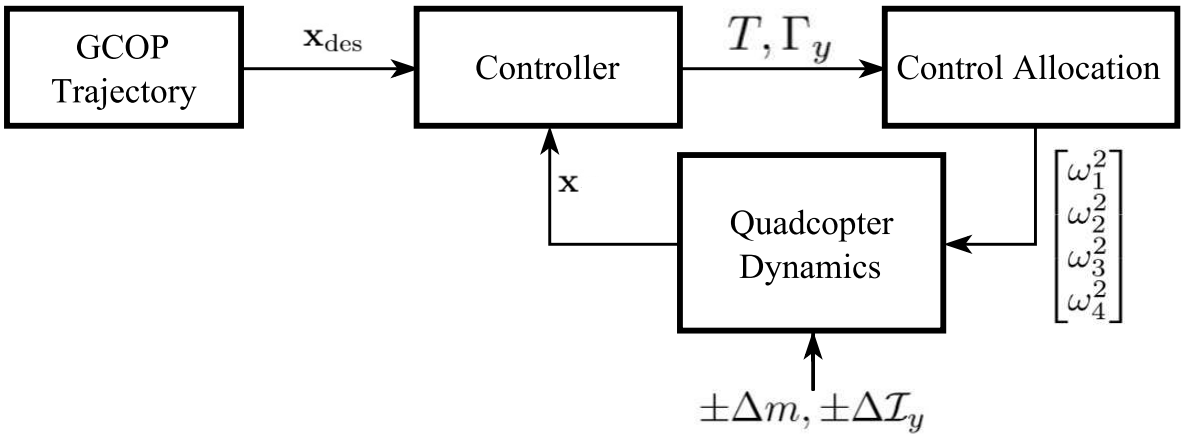


Figure 6-3: Schematic of Simulation Experiment

The GCOP reference trajectory $\mathbf{x}_{\text{des}}$ with all its states is read by the controller, as well as the state feedback $\mathbf{x}$. Based on this information, the controller computes the control thrust T and torque Γ_y. Torques Γ_x and Γ_z are collocated to zero as they don't act out-of-plane. As quadrotors have a redundant set of actuators and are underactuated, the control inputs (i.e. rotational speeds $\overline{\omega}_i^2$ for four rotors) are then allocated by solving the following expression through inversion:

$$\begin{bmatrix} T \\ \Gamma_x \\ \Gamma_y \\ \Gamma_z \end{bmatrix} = \begin{bmatrix} C_t & C_t & C_t & C_t \\ 0 & C_tL & 0 & -C_tL \\ -C_tL & 0 & C_tL & 0 \\ -C_q & C_q & -C_q & C_q \end{bmatrix} \begin{bmatrix} \overline{\omega}_1^2 \\ \overline{\omega}_2^2 \\ \overline{\omega}_3^2 \\ \overline{\omega}_4^2 \end{bmatrix} \tag{6-1}$$

The thrust coefficient C_t is identified in section 6-1-3, $L = 0.045\,\text{m}$ is the moment arm of the rotors with respect to the center of gravity, and Γ_x, Γ_y, Γ_z are the torques around the x, y and z-axis, respectively. There is no necessity to know the torque coefficient C_q, as we are allocating zero torque around the z axis, and it is sufficient to define it as any nonzero constant.

The planar quadcopter dynamics model from section 3-2 is implemented. The aerodynamic parameters were obtained experimentally in section 6-1-3. Also, the motor rotational speed limit is modelled as a saturation, which occurs at a rotational speed of $\bar{\omega} = 2513\,\text{rad}\,\text{s}^{-1}$. The gains used in the controller are shown in Table 6-2.

Table 6-2: Gains used in controller

$K_{d,x}$	$K_{p,x}$	$K_{d,z}$	$K_{p,z}$	$K_{d,\Gamma}$	$K_{p,\Gamma}$
4	4	6	9	20	100

To make the results more realistic, errors in the model are considered through a Monte-Carlo analysis by randomly perturbing the vehicle mass m and inertia $\mathcal{I}_y$ in the simulation of the

quadcopter dynamics in a range of $\pm 10\%$. The resulting trajectories give an indication of the robustness of the controller to a modeling error. They also illustrate qualitatively the sensibility to external disturbances.

6-1-3 Parameter Identification

In this analysis, the dynamic model will contain aerodynamics, i.e. a model for the rotor thrust force and for the drag forces is required. The thrust model acts in the body frame, in the upwards vertical direction , i.e. $-z_B$, and is assumed to be proportional to the squared rotational speed of each rotor. The drag model acts as well in the body frame in all three axis. Both models are created from experimental data. The thrust model is fitted to static thrust data, obtained from the manufacturer. The drag model is estimated from flight data recorded in the flight lab at DLR.

Thrust Model

In an experiment done by the developers of the crazyflie, the thrust force was measured over the entire range of rpm's (Bitcraze, 2016), see Figure 6-4. The experiment shows saturation of the brushed engines at a rotational speed of $\omega = 400\,\mathrm{s}^{-1} = 2513\,\mathrm{rad\,s}^{-1}$.

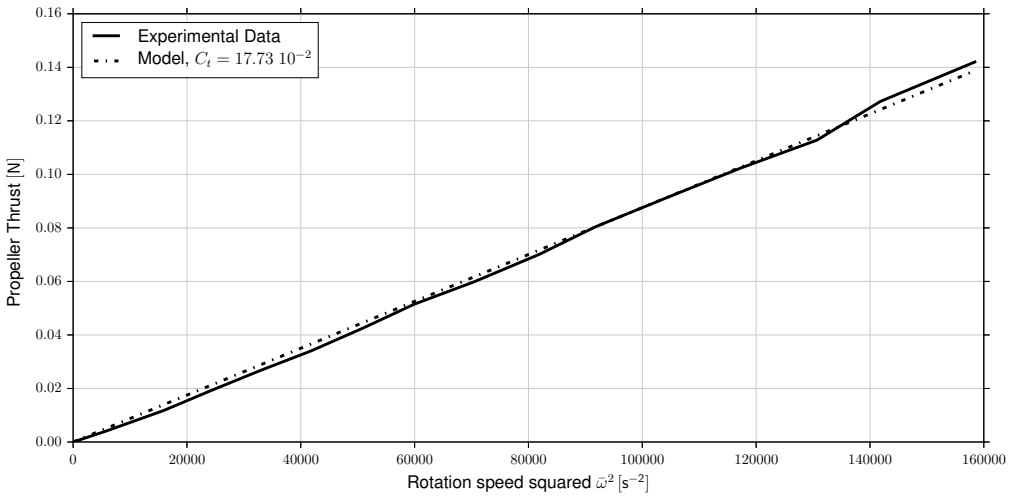


Figure 6-4: Estimation of Thrust coefficient C_t, using least squares minimization on a linear model. Mean square error of $\mathrm{MSE} = 3.4e-6$,thrust coefficient $C_t = 17.73\ 10^{-2}$.

The thrust T of a rotor is assumed to be proportional to square of the rotational speed $\overline{\omega}^2$

$$T \propto \overline{\omega}^2\,, \tag{6-2}$$

based upon theory, see Equation 3-17. By least-square fitting the data $T(\overline{\omega}^2)$ to a linear model we define the value of C_t to be $C_t = 17.73\ 10^{-2}$, with a model mean square error of $\mathrm{MSE} = 3.4e-6$.

Drag Model

The aerodynamic drag forces in the body frame $f_{\mathrm{d},x}$, $f_{\mathrm{d},y}$, $f_{\mathrm{d},z}$ are assumed to be proportional to the true airspeeds, $v_{B,x}$, $v_{B,y}$, $v_{B,z}$, respectively. This can be modeled as

$$\mathbf{f}_{\mathrm{d}} = \begin{bmatrix} f_{\mathrm{d},x} \\ f_{\mathrm{d},y} \\ f_{\mathrm{d},z} \end{bmatrix} = - \begin{pmatrix} C_x^B v_{B,x} \\ C_y^B v_{B,y} \\ C_z^B v_{B,z} \end{pmatrix} , \tag{6-3}$$

where C_x^B, C_y^B and C_z^B are the drag constants, which need to be estimated through a flight experiment in the flight laboratory at DLR.

The flight laboratory is equipped with a Vicon tracking system, from which the position and orientation of the crazyflie are logged. The data measured onboard, such as accelerations, orientation and the PWM (Pulse Width Modulated Signal) duty cycle were logged as well. From this data, the aerodynamic force in the body frame $\mathbf{f}_{\mathrm{d}}$ was then computed and used to identify the drag constants.

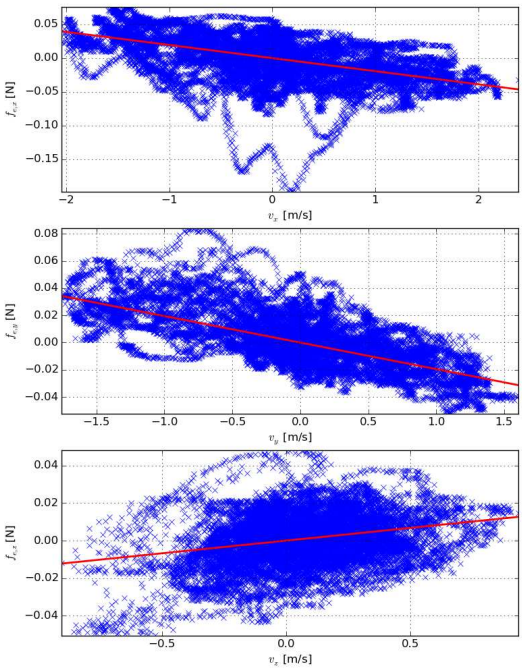


Figure 6-5: Drag forces $f_{\mathrm{d},x}$, $f_{\mathrm{d},y}$, $f_{\mathrm{d},z}$ measured experimentally for crazyflie quadrotor at different velocities v_x, v_y, v_z, respectively (blue dots). A linear model is fitted using iterative least squares (red) and the drag constants C_x^B, C_y^B and C_z^B are identified.

The resulting drag constants are $C_x^B = C_y^B = 0.0195\,\mathrm{N\,m\,s^{-1}}$ and $C_z^B = -0.0135\,\mathrm{N\,m\,s^{-1}}$. The resulting model is shown in Figure 6-5 together with the measurement points. As expected, the constants C_x^B and C_y^B are equal due to symmetry. The vertical drag constant C_z^B is negative, which is an interesting result. To validate this result, the same analysis was repeated with

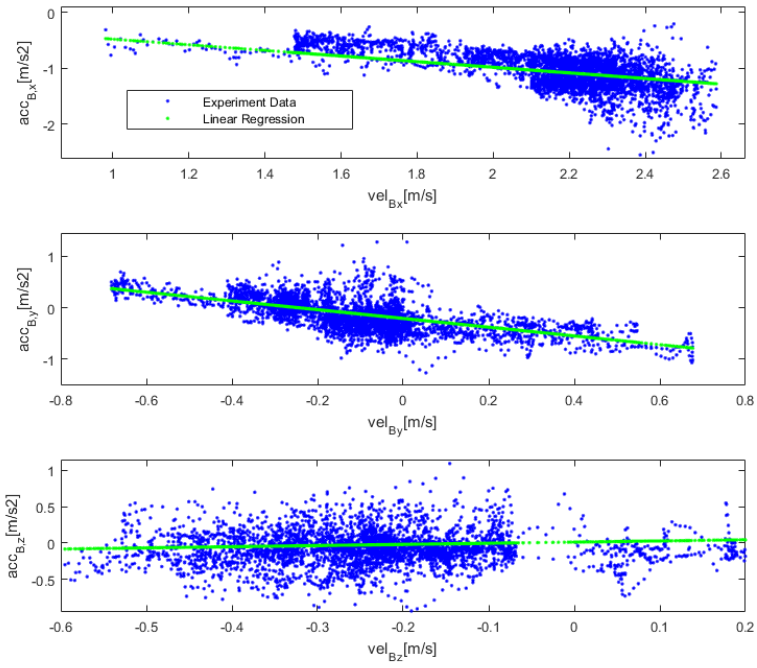


Figure 6-6: Accelerations measured experimentally at different velocities v_x, v_y, v_z, respectively (blue dots) for bebop quadrotor. A linear model is fitted using iterative least squares (green) and the drag constants C_x^B, C_y^B and C_z^B are identified.

experimental data obtained for the Bebop quadcopter at TU Delft, see Fig. 6-6. Analysis of the Bebop data shows that the vertical drag constant for this quadrotor is also negative, as the slope of the linear model in z direction is $+0.16\,\mathrm{s}^{-1}$. Therefore it can be concluded that in the z-dir of the bodyframe, the aerodynamic drag force on the frame is less dominant than the efficiency increase due to the inflow, which was described in section 3-1-3.

6-1-4 Performance Metrics

For comparison of the performance between different trajectories, corresponding metrics need to be introduced. The cost metric introduced in section 5-1 was used as the cost function for DDP, but this cost is not useful in order to make a quantitative conclusion about the energy efficiency of each trajectory, as its units are non-physical and it contains a term on the final state error.

As our solver is optimizing trajectories with respect to the control effort, we need to quantify the energy consumption for different runs. Therefore two physical performance metrics are introduced, the total energy and the average power.

Summing the power equation from equation 3-20 over the 4 rotors and integrating it with respect to time gives the energy

$$E = \int_{t=0}^{t=t_f} \sum_{i=1}^{4} \rho D^5 C_p \overline{\omega}_i^3 \mathrm{dt} \quad , \tag{6-4}$$

where $\rho = 1.205\ \mathrm{kg\,m^{-3}}$ is the air density, $D = 0.045$ m the rotor diameter and $C_p = 2\pi C_q$ is the power coefficient. The torque coefficient C_q of the rotors is not available, but it can be roughly estimated by $C_q = 0.1\ C_t$.

The average power is then the total energy divided by the simulation time t_f

$$P = \frac{1}{t_f} E = \frac{1}{t_f} \int_{t=0}^{t=t_f} \sum_{i=1}^{4} \rho D^5 C_p \overline{\omega}_i^3 \mathrm{dt} \quad . \tag{6-5}$$

These measures are only an ideal estimation of the consumed energy and average power. They do not take into account any loss due to mechanical and electrical efficiency factors.

6-2 Simulation Results

The solver has been validated in the previous chapter. The next step is to investigate how the reference trajectories for the horizontal, vertical and diagonal test cases, generated using the DDP solver, perform when used in a Simulation experiment in Simulink. The goal is to develop a controller, which can exploit in a feedforward manner the aerodynamic model of the quadrotor and information about the windspeed.

Initially it was not known, which controller would perform best in term of tracking performance, energy consumption and robustness with respect to modelling error. Therefore I came up with 4 different controllers in an iterative process, going from simple to more complex controller structure. Each flight case was tested with each controller, reference DDP trajectories with and without drag model and a Monte Carlo statistical analysis was performed for all cases by varying the mass and inertia values using a pseudo random number generator, as described in section 6-1-2. The controller tested are

- Openloop Controller,
- Attitude Controller,
- Trajectory Tracking Controller and
- Trajectory Tracking Controller with drag force feedforward.

The Trajectory Tracking Controller with drag force feedforward is then selected as the best-performing controller. In section 6-2-7 it is tested for 5 different simulation times, for the three flight cases and the 2 reference DDP trajectories and a Monte Carlos statistical analysis is performed for all these cases. The tested simulation times are $t_f = [1.5\text{ s}, 2\text{ s}, 3\text{ s}, 4\text{ s}, 5\text{ s}]$. In section 6-2-8, further simulations are run for backwind condition.

6-2-1 Open Loop Controller

As a first experiment, the reference trajectories are tested with an openloop controller, where the rotational speeds of the engines are directly fed-through as inputs to the dynamics model in the simulation.

Using the reference trajectory with drag model, the simulation trajectories (solid red, Figure 6-7) match the reference trajectories (solid black) for the three test cases in the nominal case, i.e. when no model error in m or $\mathcal{I}_y$ is introduced. This result is a consistency check of the implementation of the model in GCOP and in the simulation. More importantly it shows, that GCOP correctly feedforwards the drag model and could be used as a Model Predictive Controller (MPC). The trajectories do not match for the reference trajectory without drag model, as there is a model mismatch and no feedback terms are in place to correct for the drag forces.

Figure 6-13 shows the energy consumed by simulation trajectories, for all test cases and controllers, for a horizontal headwind of $v_x = -3\,\text{m}\,\text{s}^{-1}$.

When the Monte-Carlo Simulation is run for any reference trajectory, the error in m and $\mathcal{I}_y$ introduced by the Monte Carlo simulation causes errors in the states, but it should be

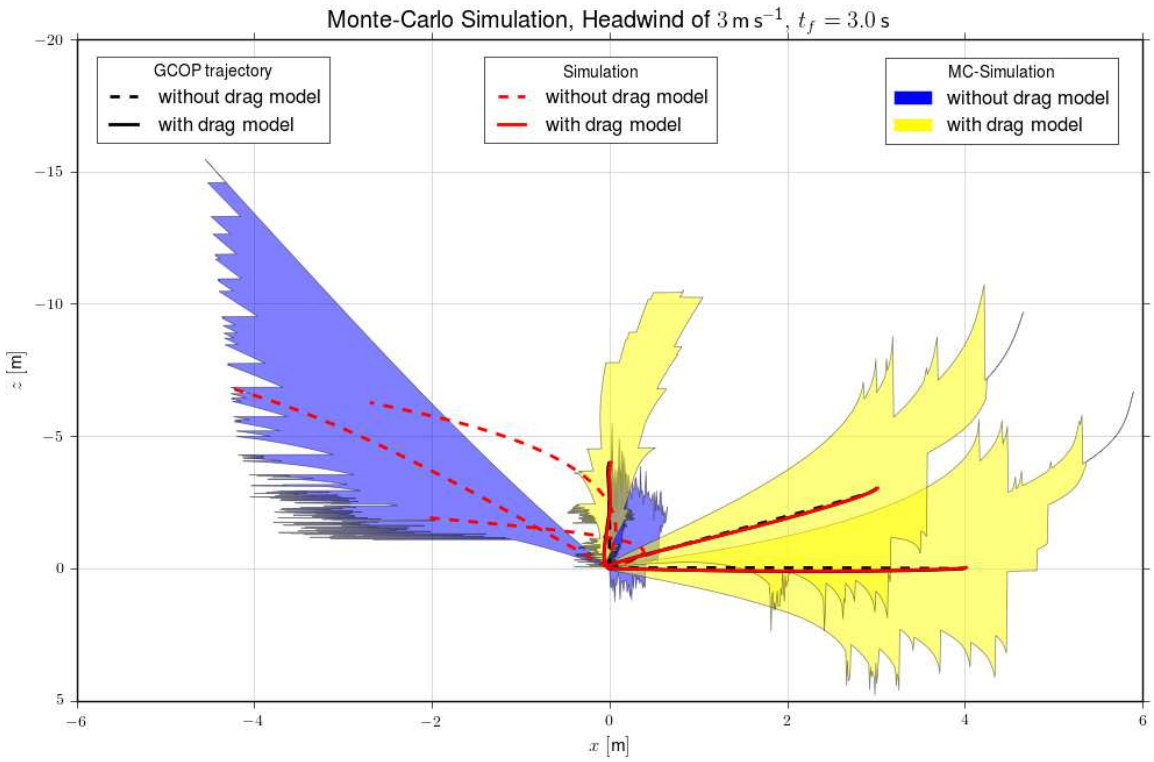


Figure 6-7: Reference trajectories (black) with and without drag, together with resulting Monte-Carlo simulation trajectories (shaded) for the horizontal, diagonal and vertical test case and horizontal wind-speed of $v_x = -3\mathrm{m\,s^{-1}}$. The open-loop controller feeds-through the rotational speeds to the dynamic model of the simulation. For the reference trajectories with drag the nominal simulation trajectories with no model error in m and $\mathcal{I}_y$ (red) match the reference trajectories.

noted that the consumed energy is not influenced by these errors, as there is no feedback loop and the rotational speeds are feedforwarded in a openloop manner. Therefore the consumed energies for the openloop controller, shown in Figure 6-13 are constant, i.e. have zero standard deviation.

The consumed energy is lower for the reference trajectories with drag model, which shows that GCOP is able to generate a more energy-efficient trajectory by including the drag model. This average decrease in control energy is significant with 9%, see Figure 6-13, where the energy costs are shown for all 4 controllers and the three test cases. It can be expected, that using DDP as a MPC with drag model would cause a similar decrease in control energy. The energy costs for the nominal trajectory with drag are

$$E_{\mathrm{diag}} = 10.13\ \mathrm{J}, \quad E_{\mathrm{ver}} = 10.12\ \mathrm{J}, \quad E_{\mathrm{hor}} = 10.56\ \mathrm{J} \quad . \tag{6-6}$$

6-2-2 Attitude Controller

Feedback terms are necessary to stabilize the quadrotor, correcting disturbances and modelling errors. In these simulation experiments, an attitude controller was implemented, which feedforwards the linear and angular accelerations $\ddot{x}_{\text{des}}$, $\ddot{z}_{\text{des}}$, $\ddot{\theta}_{\text{des}}$ from the reference trajectory and controls in close-loop the angular position θ_{des} and velocity $\dot{\theta}_{\text{des}}$.

From the linear accelerations $\ddot{x}_{\text{des}}$, $\ddot{z}_{\text{des}}$, the horizontal and vertical control force f_x and f_z are computed,

$$f_x = m\ddot{x}_{\text{des}} \ , \quad f_z = m(\ddot{z}_{\text{des}} - 9.81) \ , \tag{6-7}$$

which are then used to compute the thrust force T

$$T = \sqrt{{f_x}^2 + {f_z}^2} \ . \tag{6-8}$$

The torque is computed from the angular accelerations feedforward and the angular velocity and angular position feedback

$$\Gamma = \mathcal{I}_y(\ddot{\theta}_{\text{des}} + K_{d,\Gamma}(\dot{\theta}_{\text{des}} - \dot{\theta}) + K_{p,\Gamma}(\theta_{\text{des}} - \theta)) \ . \tag{6-9}$$

The position trajectories in the simulation have a large position error, because the control forces f_x and f_z do not take into account the drag forces. In the nominal simulation (Figure 6-8, red), the pitch angle from GCOP is well tracked, as the torque does not depend on accelerations $\ddot{x}$ and $\ddot{z}$. The nominal solution from the reference trajectory without drag is very similar for the openloop and attitude controller (see Figure 6-7), as both feedforward the same trajectory.

Now the energy consumption in Figure 6-13 (and the GCOP cost in Figure **??**) for the attitude controller is compared with energy consumption for the openloop controller. It is always lower for the openloop solution when considering the reference trajectory with drag, as the control forces of the attitude controller do not take advantage of the drag model with the negative vertical drag constant $C_y^B < 0$ and therefore overshoot the desired final states. For the reference trajectory without drag, the openloop and attitude controller solutions are very similar, as neither of them take into account the drag forces. The only difference is that there is a nonzero standard deviation in the energy and GCOP cost for the attitude controller. This happens because in order to track θ_{des} and $\dot{\theta}_{\text{des}}$ with an random simulation error in $\mathcal{I}_y$, the feedback term become active and control torques are generated, which differ from the DDP solution.

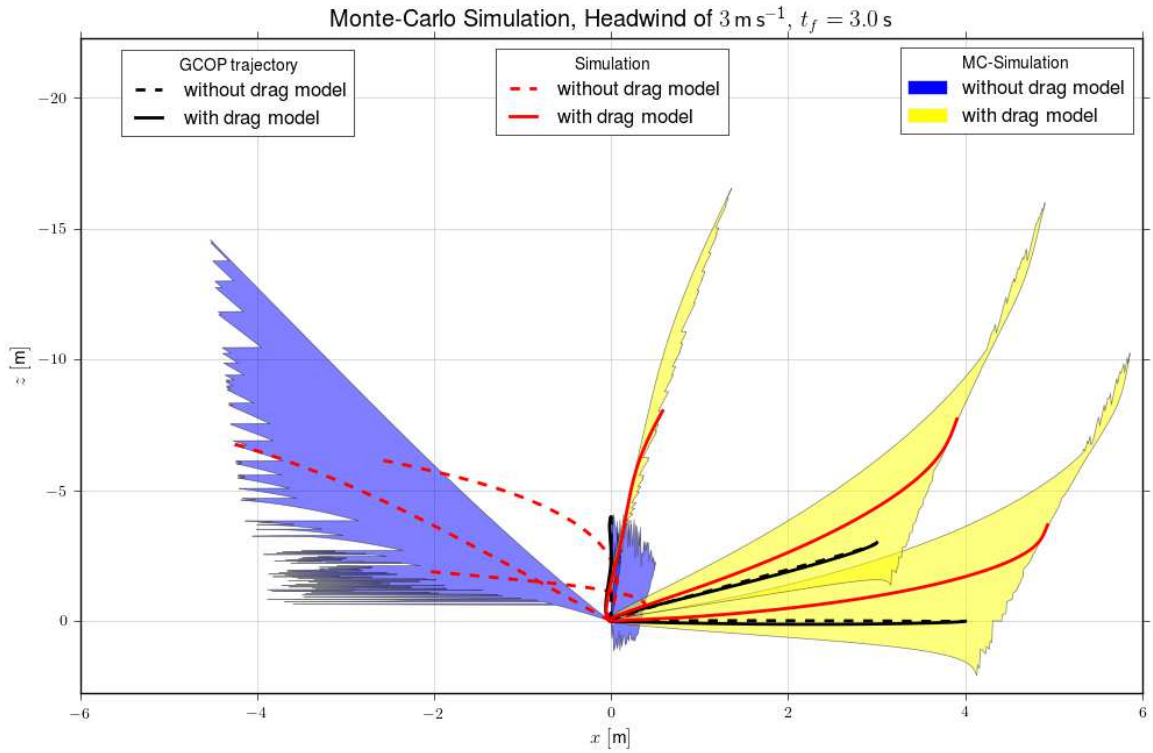


Figure 6-8: Reference trajectories (black) with and without drag, together with resulting Monte-Carlo simulation trajectories (shaded) for the horizontal, diagonal and vertical test case and horizontal windspeed of $v_x = -3\mathrm{m\,s^{-1}}$. The attitude controller feedforwards the linear and angular accelerations. The simulation trajectories have an offset in position due to the drag force generated by the horizontal headwind.

6-2-3 Trajectory Tracking Controller

In this experiment, the quadcopter is controlled by a trajectory tracking controller, where the positions x_{des}, z_{des}, the velocities $\dot{x}_{\text{des}}$, $\dot{z}_{\text{des}}$, the pitch angle θ_{ff} from the computed thrust forces and the pitch rate $\dot{\theta}_{\text{des}}$ from the GCOP trajectory are tracked in close-loop. The linear accelerations $\ddot{x}_{\text{des}}$, $\ddot{z}_{\text{des}}$ and the angular acceleration $\ddot{\theta}_{\text{des}}$ are feedforward terms and are there to improve the tracking. The control forces f_x and f_z are then computed with

$$f_x = m(\ddot{x}_{\text{des}} + K_{d,x}(\dot{x}_{\text{des}} - \dot{x}) + K_{p,x}(x_{\text{des}} - x)) \,, \tag{6-10}$$

$$f_z = m(\ddot{z}_{\text{des}} - 9.81 + K_{d,z}(\dot{z}_{\text{des}} - \dot{z}) + K_{p,z}(z_{\text{des}} - z)) \,, \tag{6-11}$$

where the thrust force is their euclidian norm

$$T = \sqrt{{f_x}^2 + {f_z}^2} \,. \tag{6-12}$$

The desired pitch angle is computed from the control forces

$$\theta_{\text{ff}} = -\text{atan2}(f_x, -f_z) \,, \tag{6-13}$$

and is tracked in the attitude controller

$$\Gamma = \mathcal{I}_y(\ddot{\theta}_{\text{des}} + K_{d,\Gamma}(\dot{\theta}_{\text{des}} - \dot{\theta}) + K_{p,\Gamma}(\theta_{\text{ff}} - \theta)) \,. \tag{6-14}$$

As the positions and velocities from the reference trajectory are now tracked in close-loop, the position tracking is better than with the attitude controller and the errors in final positions are now smaller, see Figure 6-9. The reason that there is still an error in the final positions for all nominal cases (Figure 6-9, red), is that the torque is tracking θ_{ff}, which is computed from the control forces f_x, f_z. These control forces do not take into account the drag force. At the same time, the spread of the energies in Figure 6-13 is larger compared to the attitude controller. This is expected, as this controller tracks positions and velocities in closed-loop, while different model errors in m and $\mathcal{I}_y$ are present.

The energies consumed with the open-loop controller are for the horizontal, diagonal and vertical case with drag inside the IQR of the trajectory tracking controller, which is an indication that the tracking is good, see Fig. 6-13. The median energy consumption with the trajectory tracking controller with drag is lower by an average of 9.3% then for the attitude controller. This is due to better tracking of positions and velocities. The final state error is considerably lower for the trajectory tracker, when compared to the attitude controller, see Figure 6-14.

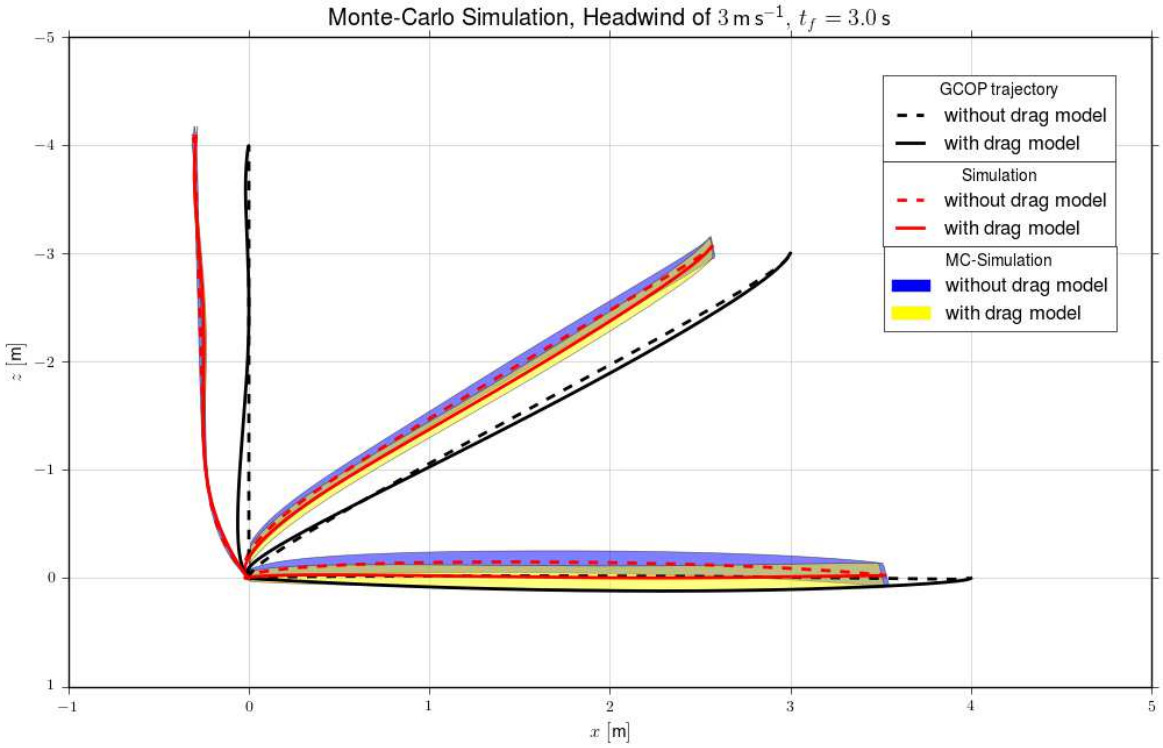


Figure 6-9: Reference trajectories (black) with and without drag, together with Monte-Carlo simulation trajectories (shaded) for the horizontal, diagonal and vertical test case and horizontal windspeed of $v_x = -3\mathrm{m\,s^{-1}}$. The trajectory tracking controller tracks the GCOP trajectories.

6-2-4 Trajectory Tracking with Aerodynamic Force Feed-forward

In order to improve the tracking performance of the trajectory tracking controller, it is extended by a feedforward term for the drag forces $f_{d,x}$ and $f_{d,z}$. They are computed from the state feedback, using the drag model and constants presented in section 6-1-3. The control forces are defined as

$$f_x = m(\ddot{x}_{\text{des}} + K_{d,x}(\dot{x}_{\text{des}} - \dot{x}) + K_{p,x}(x_{\text{des}} - x)) - f_{d,x} \ , \tag{6-15}$$

$$f_z = m(\ddot{z}_{\text{des}} - 9.81 + K_{d,z}(\dot{z}_{\text{des}} - \dot{z}) + K_{p,z}(z_{\text{des}} - z)) - f_{d,z} \ . \tag{6-16}$$

From the control forces f_x and f_z, the control pitch angle θ_{ff} and the thrust T are computed

$$\theta_{\text{ff}} = -\text{atan2}(f_x, -f_z) \ , \tag{6-17}$$

$$T = \sqrt{{f_x}^2 + {f_z}^2} \ . \tag{6-18}$$

As with the trajectory tracking controller, the torque is computed with the rotational acceleration $\ddot{\theta}_{\text{des}}$ as a feedforward term and the pitch rate $\dot{\theta}_{\text{des}}$ and angle θ_{ff} as feedback terms

$$\Gamma = \mathcal{I}_y(\ddot{\theta}_{\text{des}} + K_{d,\Gamma}(\dot{\theta}_{\text{des}} - \dot{\theta}) + K_{p,\Gamma}(\theta_{\text{ff}} - \theta)) \ . \tag{6-19}$$

The complete controller schematic is shown in Figure 6-11.

The corresponding simulation trajectories in Figure 6-10 show the best position tracking performance so far, when compared to the attitude controller and the trajectory tracking controller without drag force feedforward. For the nominal case with no model error, the simulation trajectory with drag matches the reference trajectory. Also, in the Monte Carlo simulations (shaded) good position tracking is achieved. In figure 6-13, the energy consumption for the openloop controller with drag lies within the IQR of the energy consumption for this controller with drag. The median simulation energies for the reference trajectories with drag are

$$E_{\text{diag}} = 10.23\text{J}, \quad E_{\text{ver}} = 9.74\text{J}, \quad E_{\text{hor}} = 10.61\text{J} \quad . \tag{6-20}$$

Comparing the energy consumption of this controller for a reference trajectory both with and without drag model to the open loop consumption with drag model (equation 6-6) there is only a small difference of $< 3.8\%$ for the diagonal, vertical and horizontal simulation experiments. This proves that using this controller an energy optimal trajectory can be achieved. But the energy consumption does not improve consistently or significantly by using a drag model in the trajectory generation. In fact, in Fig. 6-12 it can be appreciated that for the diagonal flight case, the resulting simulation trajectory (blue shaded) is very similar when the trajectory was generated with or without a drag model.

It can be concluded, that when the drag force is included as a feedforward term in the controller, generating the reference trajectory with a drag model does not result in an advantage in terms of energy consumption or final state error. Neither he final state error nor the energy consumption does improve. Furthermore, it has to be considered that including the drag model in the trajectory optimization results in an approximately 20% longer computation time.

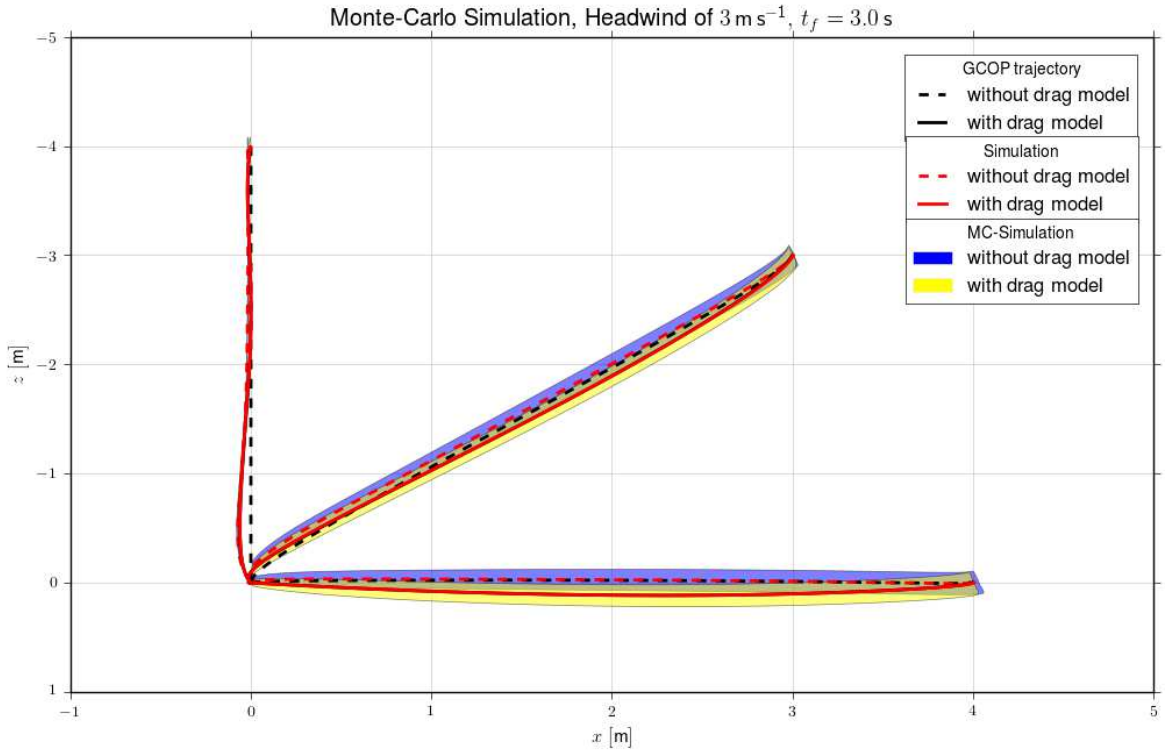


Figure 6-10: Reference trajectories (black) with and without drag, together with Monte-Carlo simulation trajectories (shaded) for the horizontal, diagonal and vertical test case and horizontal windspeed of $v_x = -3\mathrm{m\,s^{-1}}$. The trajectory tracking controller with aerodynamic feedforward achieves the best tracking from the controllers tested.

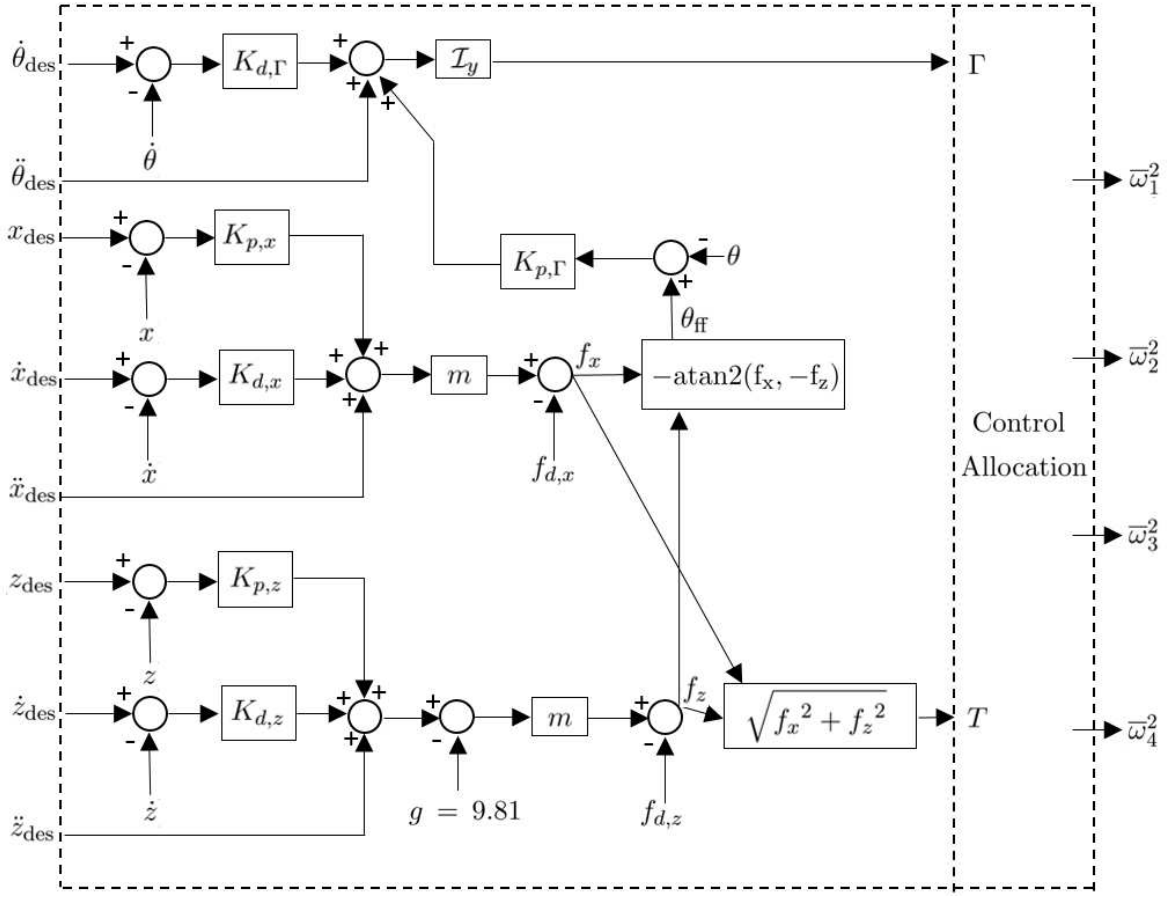


Figure 6-11: Schematic of Trajectory Tracking Controller with Drag Force Feedforward

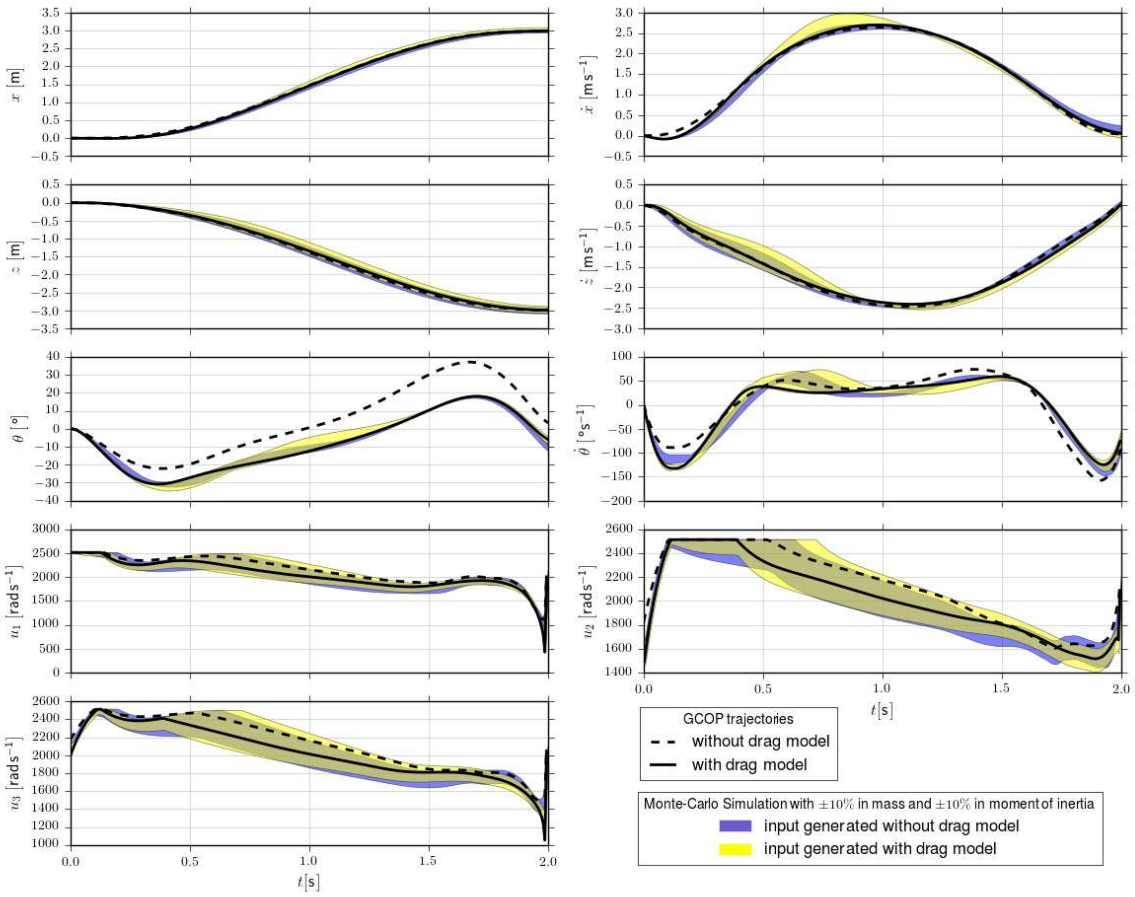


Figure 6-12: Reference trajectories (black) with and without drag, together with Monte-Carlo simulation trajectories (shaded) for the diagonal test case and horizontal windspeed of $v_x = -3\,\mathrm{m\,s^{-1}}$.

6-2-5 Energy Consumption

The energy consumptions for the three flight cases and the different controllers is computed with equation 6-4 and shown in Figure 6-13.

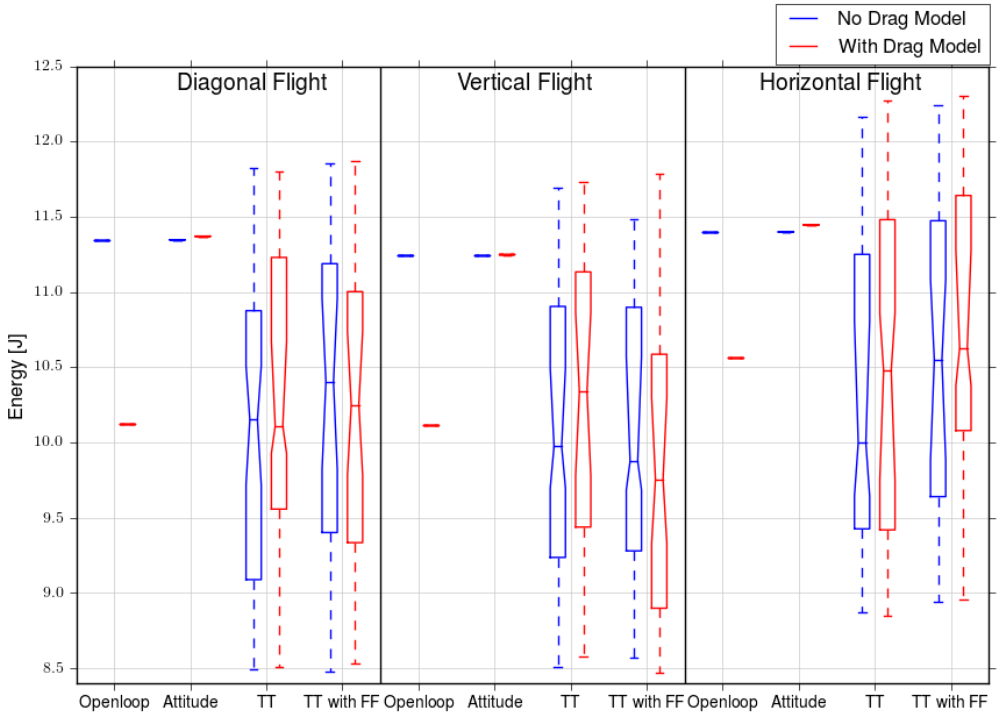


Figure 6-13: Energy consumed by simulation trajectories, for all test cases and horizontal wind-speed of $v_x = -3\mathrm{m\,s}^{-1}$. Openloop, Attitude, Trajectory Tracking, Trajectory Tracking with aerodynamic force feedforward, for reference trajectories with and without drag.

Figure 6-13 shows the openloop trajectories with drag model, which are the ideal solution in terms of energy consumption. The reason why their standard deviation or IQR is zero is that for the openloop controller the model error only influences the resulting simulation trajectory, but not the feedforwarded control input $\mathbf{u} = [\overline{\omega_1}^2, \overline{\omega_2}^2, \overline{\omega_3}^2, \overline{\omega_4}^2]^T$.

The attitude controller achieves a similar median energy for the reference trajectories without drag, as both controller feedforward the same trajectory without drag model. For the reference trajectories with drag their median energy is higher. This is due to the fact that the attitude controller doesn't take advantage of the drag model, which has a negative drag constant $C_z^B < 0$. The IQR for this controller is zero, as the attitude θ is not influenced by the drag forces.

The median energies of the trajectory tracking controller with drag force feedforward are centered around the ideal openloop solution. The reference trajectories without drag model are tracked similarly well.

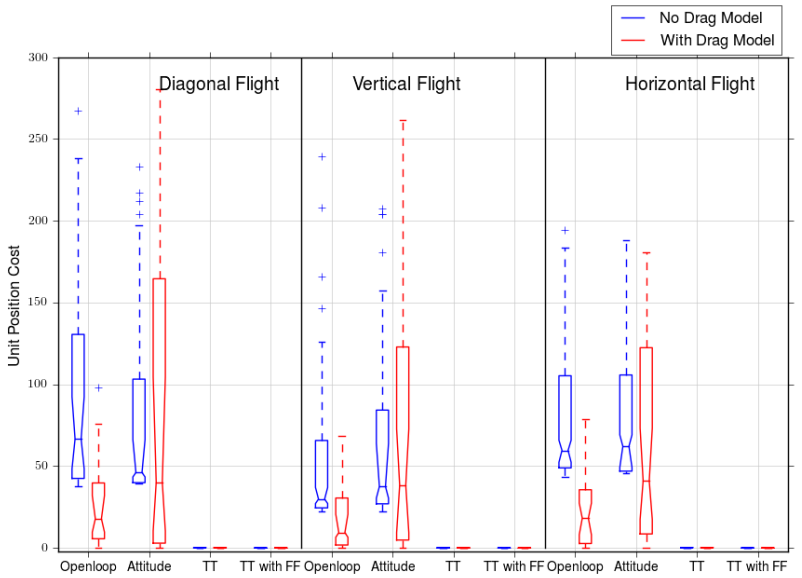
6-2-6 Final State Error

In order to quantitatively judge the final state error, a new measure is introduced,

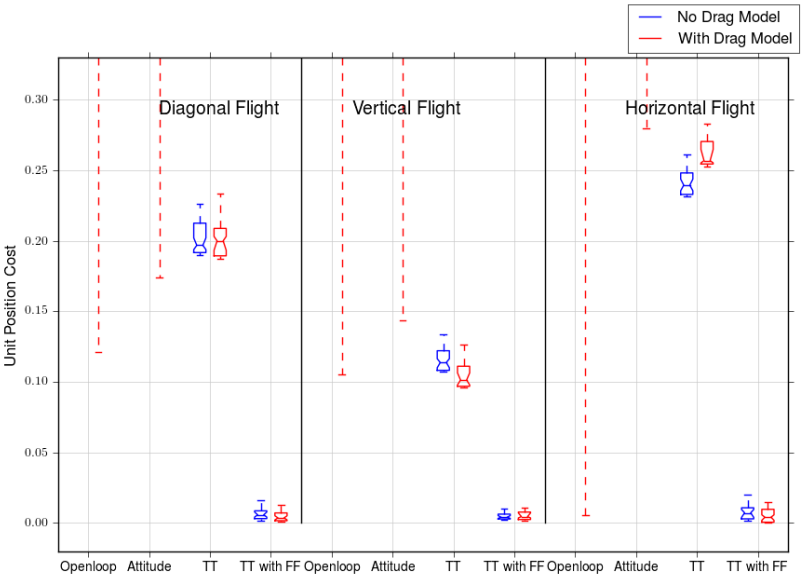
$$J_f = (x_{\text{des}} - x(t_f))^2 + (\dot{x}_{\text{des}} - \dot{x}(t_f))^2 + (\ddot{x}_{\text{des}} - \ddot{x}(t_f))^2 + (z_{\text{des}} - z(t_f))^2 + (\dot{z}_{\text{des}} - \dot{z}(t_f))^2 + (\ddot{z}_{\text{des}} - \ddot{z}(t_f))^2 \tag{6-21}$$

which equally weights the error in the final state for the positions, velocities and accelerations, the result is plotted in 6-14.

From Figure 6-14 we can conclude that the trajectory tracking controller with drag force feedforward achieves the best trajectory tracking, when compared to the other controllers. The median cost on the final state error is lower than for the trajectory tracker without drag force feedforward, by an average of 98%.



(a)



(b)

Figure 6-14: Unit Cost on the final state for the 4 tested controller, as a measure of tracking quality. It is computed for all test cases and horizontal windspeed of $v_x = -3\mathrm{m\,s}^{-1}$. Openloop, Attitude, Trajectory Tracking, Trajectory Tracking with aerodynamic force feedforward, for reference trajectories with and without drag. **(a)** shows entire range, **(b)** zooms into the cost for the results from the trajectory trackers.

6-2-7 Variations in simulation time

To study the saturation's effect on the optimality and on the energy cost of the simulated trajectories, the simulations for the same diagonal, horizontal and vertical flight problem were run using the trajectory tracking controller with drag force feedforward for different simulation end times $t_f = [2\text{ s}, 3\text{ s}, 4\text{ s}, 5\text{ s}]$. The average power is defined as the total energy consumption normalized over the simulation time t_f.

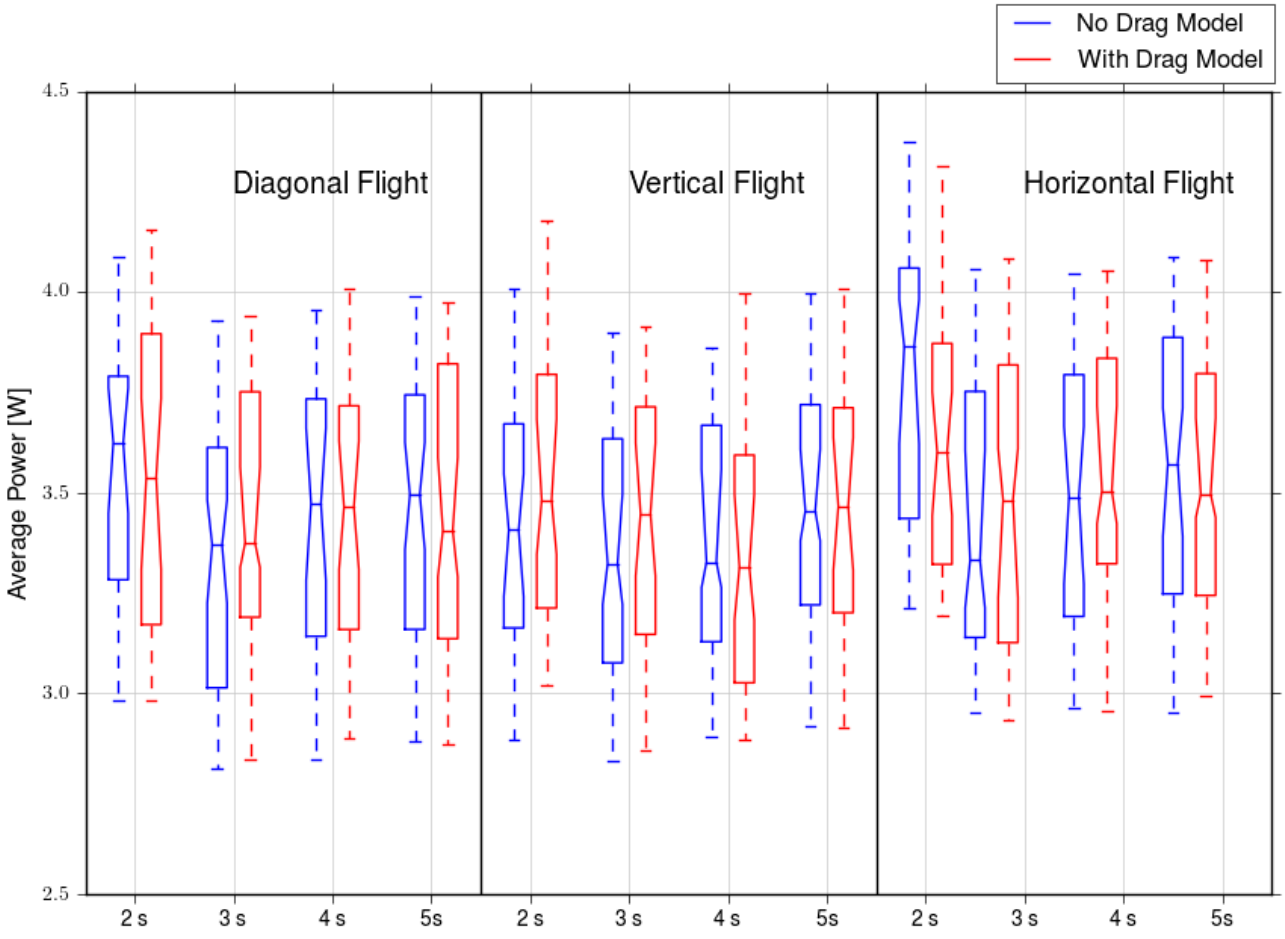


Figure 6-15: Average Power consumption for all test cases for trajectory tracking controller with aerodynamic force feedforward. Horizontal windspeed of $v_x = -3\,\text{m}\,\text{s}^{-1}$.

For a simulation time of 2 s the average power increases slightly, due to engine saturation. No consistent or significant improvement of the average power is visible when the drag model is used for the trajectory generation in the DDP-solver.

6-2-8 Backwind Results

Until now, using the trajectory tracking controller with force feedforward, the reference trajectories with and without drag model do not show a clear difference in terms of energy consumption for trajectories with a headwind of $v_x = -3\mathrm{m\,s^{-1}}$. In order to be complete, the analysis is continued for a backwind of $v_x = +3\mathrm{m\,s^{-1}}$ and a simulation time of $t_f = 3$ s, using the DDP reference trajectories with and without drag.

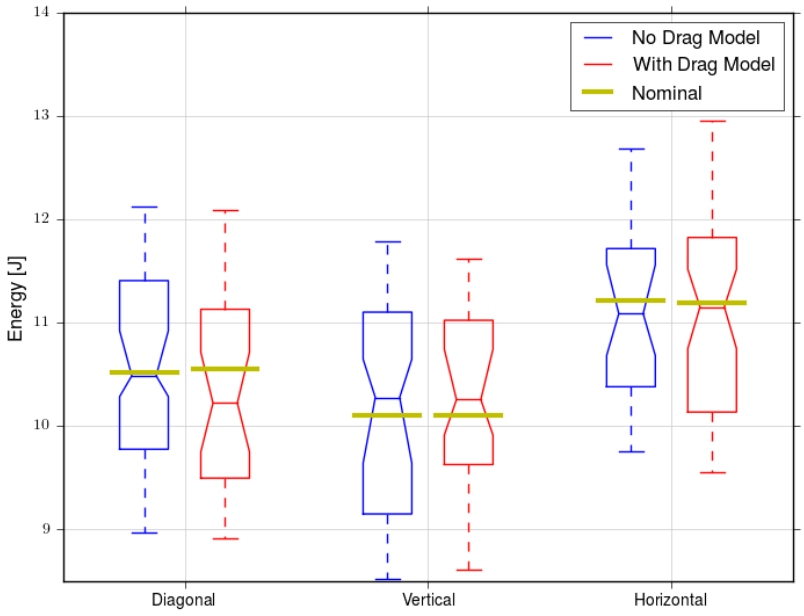


Figure 6-16: Energy required for simulation trajectories with a backwind of $v_x = -3\ \mathrm{m\,s^{-1}}$ using trajectory tracking controller with aerodynamic force feedforward. The yellow bars are the energy required for the nominal case without model error. The boxplots illustrate the results for the Monte Carlo Simulations with a simulation time of $t_f = 3$ s

Figure 6-16 shows in the diagonal case a lower median energy for the reference trajectory with drag, for both the vertical and diagonal cases the median energies are similar with and without drag. The nominal case (yellow bar) is the simulation result, when no modelling error is introduced in the simulation model, i.e. the dynamic models used in DDP and in the simulation are the same. Similar as for the nominal simulation results with headwind, using the trajectory with or without drag doesn't make a difference.

Chapter 7

Conclusion and Recommendations

7-1 Conclusion

In the context of trajectory generation for quadrotors, state of the art schemes assume a simple dynamic model, which neglect aerodynamic effects. The work presented in this thesis firstly shows that it is feasible to generate trajectories with a DDP optimal control solver which uses an aerodynamic model. Secondly, it shows that when a tracking controller is used which feedforwards the aerodynamic forces, a $\sim 10\%$ improvement in the energy consumption can be achieved. No performance improvement is achieved by using a reference trajectory which was generated using an aerodynamic model.

Focusing on the trajectory optimization, it can be concluded

- By enhancing the simple dynamic model with a linear drag force model which depends on the true airspeed, the energy consumption for a specific trajectory can be reduced significantly. For the crazyflie model it is about 9% improvement for a 3m/s headwind situation. This benefit is directly correlated to the negative drag constant in vertical direction and emphasizes the importance of using a high-quality aerodynamic model, which closely resembles the real system.
- Differential Dynamic Programming yields optimal trajectories for minimum-control effort, terminal control problems and reliably takes into account motor saturation. It is successfully validated with a reference solver. Nevertheless the result is sensitive to the cost weights on the controls and on the final state errors. Therefore initially the cost weights have to be tuned heuristically.
- Through normalization of the states and control inputs, the heuristic tuning of the cost weights can be avoided, which is a significant time-saving factor. Nevertheless, normalization also increases the computation time, as the transformation from physical states/control to normalized states/control and back must be repeated for every discretization step in the backward pass of the DDP algorithm.

- It is sufficient to initialize the DDP solver with a simple polynomial trajectory, which is derived from the differentially flat quadrotor model. This method is reliable and is able to take into account boundary conditions.
- Including an aerodynamic model in the DDP-solver results in an considerable increase in computation time of approximately $\sim 15\%$.

Focusing on the controller, it is concluded

- By feedforwarding wind information and trajectory accelerations, and controlling positions and velocities in close loop, the aerodynamic model can be successfully exploited.
- A robust controller performance was achieved for the proposed trajectory tracker with drag force feedforward. In a Monte-Carlo Simulation with model errors in mass and moment of inertia up to +-10%, a good tracking of the final state with a position error of $\pm$10cm was achieved.
- The Monte-Carlo Simulation shows that the energy consumption and final state error is independent of whether the trajectory was generated with or without an aerodynamic model. In other words, when the proposed trajectory tracking controller with drag force feedforward is used, no performance improvement is achieved by tracking a reference trajectory which was generated using an aerodynamic model.

7-2 Recommendations

- If a linear drag model is used, it is recommended to include the aerodynamic model in the controller and not in the DDP-solver, for optimal performance in terms of energy consumption and computational efficiency.
- For future work, it is recommended to perform the experiment on real hardware in a constant airfield (e.g. OJF windtunnel), in combination with a high-quality aerodynamic model, in order to validate the proposed controller structure experimentally.
- DDP enables the implementation of an aerodynamic model which depends of position (and time). Therefore it would be interesting to study the performance when flying in an environment with varying windspeed, e.g. perpendicularly through the airstream generated by a fan. It could be that for this experiment, generating the reference trajectory with drag model leads to better results than without.
- The benefit of feedforwarding the aerodynamic forces in the controller in a real experiment is dependent on the use of a high-quality aerodynamic model. Currently, promising work is done at TU Delft in the development of a Bebop quadrotor model, which is based upon polynomial terms. Computing the minimum-energy trajectory using such a model with better fit will give interesting insights into whether the model structure used in this thesis is sufficient to exploit aerodynamics effects or whether it is too simple.

- Running the trajectory optimization algorithm online in combination with a method to locally estimate (see (Tomic & Haddadin, 2015)), predict or even map windspeeds, might lead to interesting insights.

Chapter 8

APPENDIX

8-1 Trajectory Initialization

In order to generate an initial solution we want to take advantage of the fact that the planar model without drag model is differentially flat (Sira-Ramirez & Agrawal, 2004).

Based on the planar model, a trajectory requires eight boundary conditions along each axis: the initial and final jerks ($\dddot{\mathbf{x}}_0$, $\dddot{\mathbf{x}}_\mathrm{f}$), the accelerations ($\ddot{\mathbf{x}}_0$, $\ddot{\mathbf{x}}_\mathrm{f}$) and velocities ($\dot{\mathbf{x}}_0$, $\dot{\mathbf{x}}_\mathrm{f}$), the initial and final positions ($\mathbf{x}_0$, $\mathbf{x}_\mathrm{f}$).

In general, to comply with two boundary conditions on the 3th derivative of a polynomial, the degree of this polynomial has to be higher or equal to 5. Nevertheless in our case 4 of the 8 boundary conditions might cancel a constant out, as they are equal to zero. This means we need in total 8 constants to solve the polynomial, therefore we need a 7th order polynomial. The general solution, for an arbitrary axis, has the form

$$x(t) = \frac{C_1}{5040}t^7 + \frac{C_2}{720}t^6 + \frac{C_3}{120}t^5 + \frac{C_4}{24}t^4 + \frac{C_5}{6}t^3 + \frac{C_6}{2}t^2 + C_7 t + C_8 \,, \tag{8-1}$$

with the constants

$$C_1 = 840\left(\frac{\dddot{x}_\mathrm{f} - \dddot{x}_0}{t_\mathrm{f}^4} - \frac{12(\ddot{x}_\mathrm{f} - \ddot{x}_0)}{t_\mathrm{f}^5} + \frac{60(\dot{x}_0 + \dot{x}_\mathrm{f})}{t_\mathrm{f}^6} + -\frac{120(x_\mathrm{f} - x_0)}{t_\mathrm{f}^7} \right), \quad (8\text{-}2)$$

$$C_2 = 360\left(\frac{(22 - \frac{70}{3})\dddot{x}_0}{t_\mathrm{f}^3} - \frac{\ddot{x}_\mathrm{f}}{t_\mathrm{f}^3} - \frac{15\ddot{x}_0}{t_\mathrm{f}^4} + \frac{13\ddot{x}_\mathrm{f}}{t_\mathrm{f}^4} - \frac{72\dot{x}_0}{t_\mathrm{f}^5} - \frac{68\dot{x}_\mathrm{f}}{t_\mathrm{f}^5} + \frac{140(x_\mathrm{f} - x_0)}{t_\mathrm{f}^6} \right), \quad (8\text{-}3)$$

$$C_3 = -60\left(-\frac{\ddot{x}_\mathrm{f}}{t_\mathrm{f}^2} - \frac{20\ddot{x}_0}{t_\mathrm{f}^3} + \frac{14\ddot{x}_\mathrm{f}}{t_\mathrm{f}^3} - \frac{2\ddot{x}_0}{t_\mathrm{f}^2} - \frac{90\dot{x}_0}{t_\mathrm{f}^4} - \frac{78\dot{x}_\mathrm{f}}{t_\mathrm{f}^4} + \frac{168(x_\mathrm{f} - x_0)}{t_\mathrm{f}^5} \right), \quad (8\text{-}4)$$

$$C_4 = 4\left(-4\frac{\ddot{x}_0}{t_\mathrm{f}} - \frac{\ddot{x}_\mathrm{f}}{t_\mathrm{f}} + \frac{-30\dot{x}_0}{t_\mathrm{f}^2} + \frac{15\dot{x}_\mathrm{f}}{t_\mathrm{f}^2} - \frac{35\ddot{x}_0}{t_\mathrm{f}} - \frac{120\dot{x}_0}{t_\mathrm{f}^3} - \frac{90\dot{x}_\mathrm{f}}{t_\mathrm{f}^3} + \frac{210(\dot{x}_\mathrm{f} - \dot{x}_0)}{t_\mathrm{f}^4} \right), \quad (8\text{-}5)$$

$$C_5 = \dddot{x}_0 , \quad (8\text{-}6)$$

$$C_6 = \ddot{x}_0 , \quad (8\text{-}7)$$

$$C_7 = \dot{x}_0 , \quad (8\text{-}8)$$

$$C_8 = x_0 . \quad (8\text{-}9)$$

8-2 Validation Trajectories

Horizontal Flight

The results for the horizontal flight (see Fig. 8-1) show that in the acceleration phase, the pitch angle θ is higher with the drag model, as there is a drag force to overcome and both drag constants are positive. In the deceleration phase the pitch angle θ is lower for the drag model, as the drag is already applying a force in negative x-direction.

With and without drag the same max rotational rate $\dot{\theta}_{max} = 40\ ^\circ\,\mathrm{s}^{-1}$ is reached. It is apparent that with drag, the maximum rotational speed is reached later, as the drag creates a longer acceleration phase.

Furthermore it is apparent that the thrust curve with drag is not symmetrical, as the quadrotor with drag requires in the acceleration phase a higher thrust and in the deceleration phase less thrust due to the aerodynamic drag force.

The torque shows a strong deflection at $t = t_0$ and $t = t_f$. The curve coincides for the case with and without drag, as the slope of the rotational speeds $\dot{\theta}$ are comparable and $u_2 = \ddot{\theta}$.

The vertical position z and velocity $\dot{z}$ show slight deviations from zero, which are less than one centimetre and two centimetres per second, and therefore negligible.Their magnitudes coincides for both solvers.

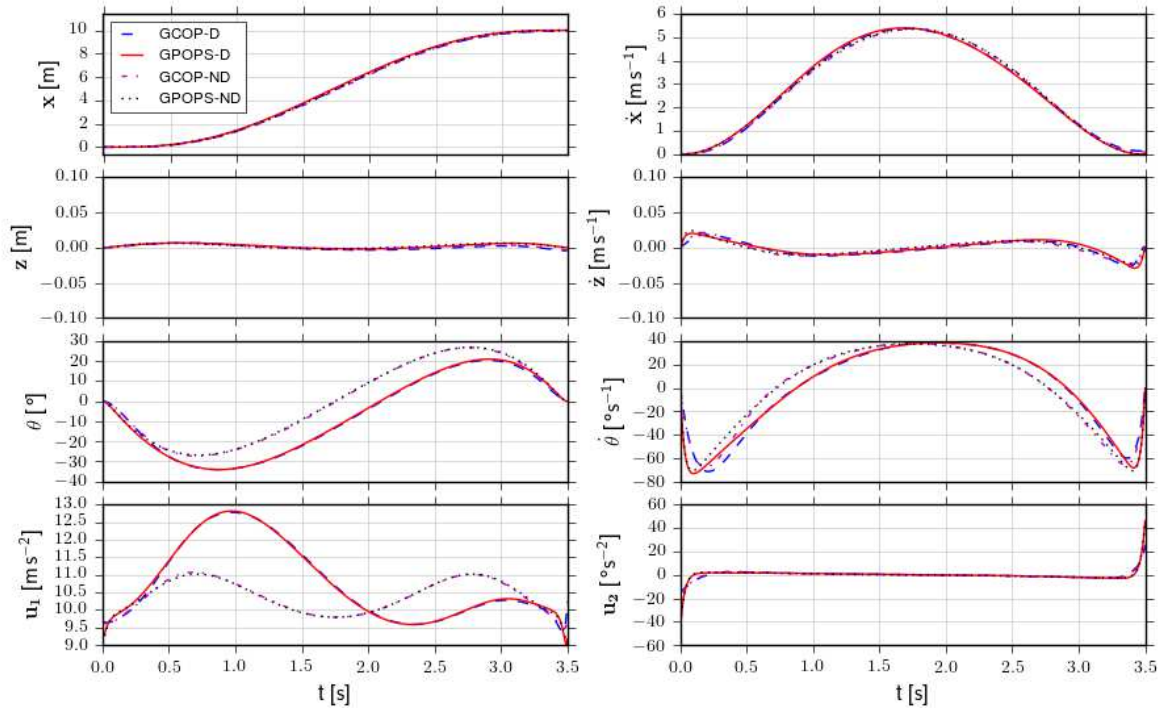


Figure 8-1: Validation trajectories for the horizontal flight problem without (ND) and with (D) drag model for both solvers.

Diagonal Flight

The results for the diagonal flight problem (see Fig. 8-2) show a combination of the phenomena seen in the horizontal and vertical problem.

As already seen in the horizontal case, the pitch angle θ is higher in the acceleration phase than in the deceleration phase, for both GCOP and GPOPS solutions with drag. For the case with no drag, it is similar in both phases.

The vertical accelerations show large oscillations for the models without drag, when compared to the model with drag, which shows a more "damped" behaviour.

The solution for the vertical position from GPOPS with and without drag and the curve from GCOP with drag are very similar. The result from GCOP without drag model shows a slight oscillation in z.

The curves with drag reach the maximum rotational speed $\dot{\theta}$ later.

In a similar manner, the max. velocity $\dot{x}_{max}$ is reached earlier in the case with drag.

Like in the vertical flight problem, the thrust input u_1 shows a large value at the start of the simulation and a very low values at the simulation end.

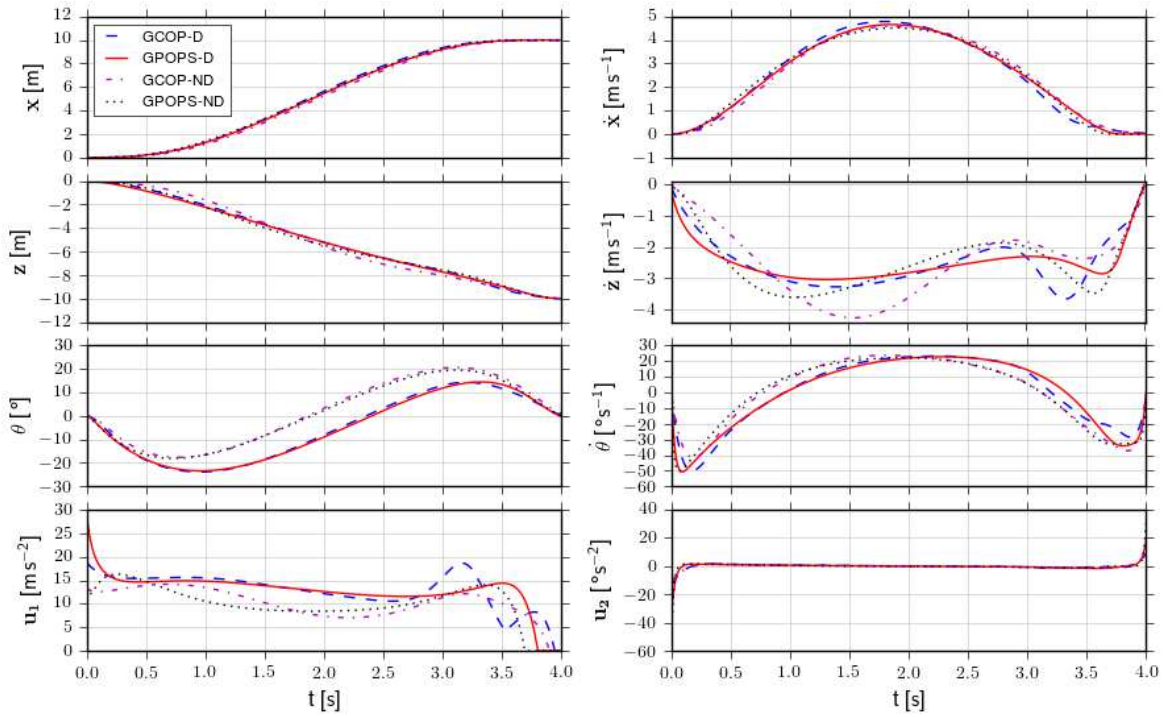


Figure 8-2: Validation trajectories for diagonal flight problem without (ND) and with (D) drag model for both solvers.

Bibliography

A. R. S. Bramwell, D. B., G. Done. (2001). *Bramwell's helicopter dynamics*. Oxford: Butterworth-Heinemann. (Second edition)

Benallegue, A., Mokhtari, A., & Fridman, L. (2006). Feedback linearization and high order sliding mode observer for a quadrotor UAV. *International Workshop on Variable Structure Systems, 2006. VSS'06.*, 365–372.

Bertsekas, D. P. (2001). *Dynamic programming and optimal control, second edition*. Nashua: Athenas Scientific.

Bitcraze. (2016). *This is the analyses of finding a pwm to thrust transfer function.* `https://wiki.bitcraze.io/misc:investigations:thrust`.

Bruno Siciliano, e. a. (2010). *Robotics - modelling, planning and control*. Berlin: Springer.

Cabecinhas, D., Naldi, R., Marconi, L., Silvestre, C., & Cunha, R. (2012). Robust take-off for a quadrotor vehicle. *IEEE Transactions on Robotics*, *28*, 734–742.

DE O. PANTOJA, J. (1988). Differential dynamic programming and newton's method. *International Journal of Control*, *47*(5), 1539–1553.

Garg, D., & Patterson, M. (2009). An overview of three pseudospectral methods for the numerical solution of optimal control problems. *Advances in the Astronautical Sciences*, *135*.

Garimella, G. (2016). *Research gowtham garimella.* `http://flyingmanipulators.lcsr.jhu.edu/about/research/`.

Geoffroy, P., Mansard, N., Raison, M., Achiche, S., & Todorov, E. (2014). From inverse kinematics to optimal control. *Advances in Robot Kinematics*, 409–418.

Gill, P., Murray, W., & Saunders, M. (2005). SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Review*, *47*(1), 99–131.

Gill, P. E., Murray, W., & Saunders, M. A. (2008). *Users guide for snopt version 7: Software for large-scale nonlinear programming.*

Gillula, J. H., Huang, H., Vitus, M. P., & Tomlin, C. J. (2010). Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice. *2010 IEEE International Conference on Robotics and Automation (ICRA)*, 1649–1654.

Guillaume Allibert, Abeywardena, D., Bangura, M., & Mahony, R. (2014). Estimating Body-

Fixed Frame Velocity and Attitude from Inertial Measurements for a Quadrotor Vehicle. *IEEE*.

Hoffmann, G. M., Waslander, S. L., & Tomlin, C. J. (2008). Quadrotor helicopter trajectory tracking control. *AIAA*, 1–14.

IPATE, G., VOICU, G., & DINU, I. (2015). Research on the use of drones in precision agriculture. *University Politehnica of Bucharest Scientific Bulletin*, *77*(ISSN 1454-2358), 263–274.

Josua Braun, T. S. (2015). *Moderne methoden der regelungstechnik 2, tum.*

Kirk, D. E. (2006). *Optimal control theory - an introduction.* Mineola, New York: Dover Publications.

Kobilarov, M. (2016). *Jhu gcop.* `https://github.com/jhu-asco/gcop`.

Leisham, J. G. (2006). *Principles of helicopter aerodynamics.* Cambridge: Cambridge Aerospace Series. (2nd Edition)

Leishman, R. C., Macdonald, J. C., Beard, R. W., & McLain, T. W. (2014). Quadrotors and Accelerometers: State Estimation with an Improved Dynamic Model. *IEEE Control Systems*, *34*, 28–41.

Luque-Vega, L. F., Castillo-Toledo, B., Loukianov, A., & Gonzalez-Jimenez, L. E. (2014). Power line inspection via an unmanned aerial system based on the quadrotor helicopter. *MELECON*, 393–397.

Mahony, R., Kumar, V., & Corke, P. (2012). Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor. *IEEE Robotics Automation Magazine*, 20–32.

Maier, M. (2013). *Learning quadrotor maneuver from optimal control.* Unpublished diploma Thesis, TU Muenchen.

Markwalter, B. (2015). Flights of fancy: Products in the unmanned systems marketplace. *IEEE Consumer Electronics Magazine*, *4*, 46–48.

Martin, P., & Salaun, E. (2010, May). The true role of accelerometer feedback in quadrotor control. *IEEE International Conference on Robotics and Automation (ICRA)*, 1623–1629.

Mathew, N., Smith, S. L., & Waslander, S. L. (2015). Planning paths for package delivery in heterogeneous multirobot teams. *IEEE Transactions on Automation Science and Engineering*(IEEE 2015-4), 1298–1308.

Mellinger, D., & Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. *2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2520–2525.

Mueller, M. W., Hehn, M., & D'Andrea, R. (2015). A computationally efficient motion primitive for quadrocopter trajectory generation. *IEEE Trans. Robot.*, 1294–1310.

Murray, D., & Yakowitz, S. (1984). Differential dynamic programming and newton's method for discrete optimal control problems. *Journal of Optimization Theory and Applications*, *43*, 395-414.

Omari, S., Hua, M. D., Ducard, G., & Hamel, T. (2013). Nonlinear control of VTOL UAVs incorporating flapping dynamics. *International Conference on Intelligent Robots and Systems (IROS)*, 24192425.

Pierre-Jean, B., Callou, F., Vissiere, D., & Petit, N. (2011). The navigation and control technology inside the ar.drone micro uav. *18th IFAC World Congress*.

Pounds, P., Mahony, R., & Corke, P. (2010). Modelling and control of a large quadrotor robot. *Control Engineering Practice, Special Issue on Aerial Robotics*, 691–699.

Rao, A. V., Benson, D. A., Darby, C., Patterson, M. A., Francolin, C., Sanders, I., et al.

(2010). Algorithm 902: Gpops, a matlab software for solving multiple-phase optimal control problems using the gauss pseudospectral method. *ACM Trans. Math. Softw.*, *37*(2), 22:1–22:39.

Sharifi, F., Zhang, Y., & Gordon, B. W. (2011). Voronoi-based coverage control for multi-quadrotor uavs. *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 991–996.

Sira-Ramirez, H., & Agrawal, S. K. (2004). *Differentially flat systems*. New York: CRC Press.

Sreenath, K., Michael, N., & Kumar, V. (2013). Trajectory generation and control of a quadrotor with a cable-suspended load - A differentially-flat hybrid system. *IEEE International Conference on Robotics and Automation (ICRA)*, 4888–4895.

T. Hamel, Mahony, R., Lozano, R., & Ostrowski, J. (2002). Dynamic modelling and configuration stabilization for an x4 flyer. *IFAC*.

Tassa, Y., Mansard, N., & Todorov, E. (2003). Fast model predictive control for reactive robotic swimming. *International Conference on Intelligent Robots and Systems*.

Tassa, Y., Mansard, N., & Todorov, E. (2014). Control-limited differential dynamic programming. *Robotics and Automation (ICRA), IEEE*, 1168–1175.

Tedrake, R. (2009). LQR-Trees: Feedback motion planning on sparse randomized trees. *Robotics: Science and Systems V*.

Tomic, T., & Haddadin, S. (2015). Simultaneous estimation of aerodynamic and contact forces in flying robots: Applications to metric wind estimation and collision detection. *IEEE International Conference on Robotics and Automation (ICRA)*, 5290–5296.

Tomic, T., Maier, M., & Haddadin, S. (2014). Learning quadrotor maneuvers from optimal control and generalizing in real-time. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 1747–1754.

Waslander, S. L., & Wang, C. (2009). Wind disturbance estimation and rejection for quadrotor position control. *AIAA Infotech Aerospace Conference and AIAA Unmanned... Unlimited Conference*.

Watkinson, J. (2004). *The art of the helicopter*. Oxford: Butterworth-Heinemann.

Zairi, S., Hazry, D., et al. (2010). *Stability mechanism of an quadrotor for easy sprayer aerial vehicle* (Report). Malaysia: Universiti Malaysia Perlis (UniMAP).