ÉCOLE CENTRALE DE NANTES

MASTER CORO-IMARO
"CONTROL AND ROBOTICS"

2020 / 2021

Master Thesis Report

Presented by

Elie Hatem

On July 3, 2021

Making Flips With Quadrotors In Constrained Environments

Jury

Evaluators:	Dr. Olivier Kermorgant	Associate Professor (ECN)
	Dr. Damien SIX	Robotics Engineer (CNRS)
Supervisor(s):	Dr. Sébastien Briot	Researcher (CNRS)
	Dr. Isabelle Fantoni	Research Director (CNRS)

Laboratory: Laboratoire des Sciences du Numérique de Nantes LS2N

Abstract

Within the rapidly growing aerial robotics market, one of the most substantial challenges in the quadrotor community is performing aggressive maneuvers, especially multi-flip maneuvers. A proper physical definition of the issue is not addressed by the current approaches in the field and several key aspects of this maneuver are still overlooked. It can be shown, in particular, that making a flip with a quadrotor means crossing the parallel singularity of the dynamic model. The aim of the master thesis is to explore the possibility of defining aggressive trajectories for quadrotors on the basis of their dynamic model degeneracy analysis and to adapt various strategies to control the robot in a closed loop. In addition, the possibility of performing the aggressive maneuvers in constrained environments will also be investigated. Therefore, the analysis will be extended from the previous studies to create general feasible trajectories that will allow quadrotors to perform aggressive multi-flip maneuvers while passing through a constrained environment and while guaranteeing a satisfactory degree of robustness to the uncertainties of the dynamic model.

Keywords: quadrotors, parallel robots, aggressive maneuvers, multi-flips, constrained environmen.

Acknowledgements

I would like to express my special thanks and gratitude to my supervisors Dr. Sébastien Briot and Dr. Isabelle Fantoni who gave me the opportunity to work on this wonderful project which encapsulates control theory, dynamics and quadrotors, which are all subjects that are very interesting for me. This project has allowed me to perform research on all of these topics and I am now more knowledgeable thanks to my supervisors. Moreover, I would like to thank them for believing in my capabilities and giving me the confidence and the support when I needed it.

I would like to thank my patient and understanding girlfriend Glysa, who has been with me for more than 6 years. Thank you for all the love, support and comfort that you have given me in these stressful 2 years.

I would like to thank my family as well: my parents Naji and Yolla, my sister Rebecca, my uncle Fadi, his wife Lara and my aunt Bernadette. They have provided me with the emotional and economical support from the very beginning and they gave me the opportunity to travel and study for this Master's degree. They have always been proud and encouraging.
I would not be here if it wasn't for them.

Notations

I_{xx}, I_{yy}, I_{zz}	Diagonal terms of the inertia matrix
$\omega_x, \omega_y, \omega_z$	angular rates with respect to the x,y and z axes respectively
ϕ, θ, ψ	roll, pitch and yaw angles respectively
l	Arm length of a quadrotor
T	Total thrust input of the quadrotor
τ	Total torque of the quadrotor
T_s	Sample Time
N	Prediction horizon
m	Control horizon
J	Objective function

Abbreviations

UAV	unmanned aerial vehicle
CoG	center of gravity
MPC	model predictive control
NMPC	nonlinear model predictive control
HLC	high level commander
SMC	sliding mode control
MIMO	multi-input multi-output
SQP	sequential quadratic program
NLP	nonlinear program

List of Figures

List of Tables

Contents

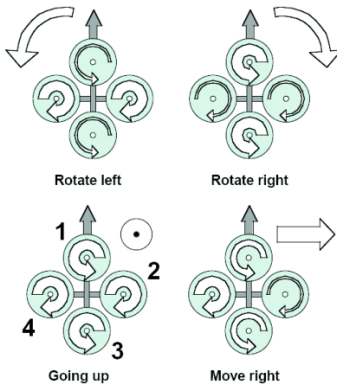
Introduction

The aim of this section is to provide a general summary of the robotic platform that is used for this master thesis and to illustrate the main objective of the research work. In specific, in the sections below, quadrotors and parallel robots are briefly presented.

The quadrotor platform

A quadrotor is a type of unmanned aerial vehicle (UAV) with four rotors and six degrees of freedom. Typically, drones have a small size and low inertia which allows them to be controlled by simple flight control systems. It is typically designed in a cross-configuration such that the electronics are held in the center of the platform and the rotors are placed at the borders. An example of a real quadrotor, namely the DJI Phantom, is shown in figure 1a. The quadrotor is typically built in a way such that a pair of opposite rotors rotates in a clockwise direction, whereas the other pair rotates in a counter-clockwise direction. The attitude and the position of the drone are controlled by changing the spinning speed of the rotors, as showin in figure 1b.

(a) A DJI Phantom quadcopter (UAV)[1]



(b) Representation of the concept of a quadrotor. The width of the arrows is proportional to the angular speed of the propellers.[6]

Figure 1: A commercial quadrtotor platform with a representation of the quadrotor concept.

The distinctive mechanical design of the quadrotor permits the actuation system to control all of the six degrees of freedom even though it is under-actuated. This is due to the fact that the rotational and translational dynamics are tightly coupled. Thus, all the translational and rotational motions can be carried off by properly controlling the magnitude and direction of the spinning speed of the rotors.

[1] https://en.wikipedia.org/wiki/Quadcopter#/media/File:Quadcopter_camera_drone_in_flight.

Over the last few years, quadrotors have gained a large popularity in academia and in the industry. This is due to several reasons, such as:

1. Quadrotors are very simple to design and they can be easily assembled using relatively cheap components.

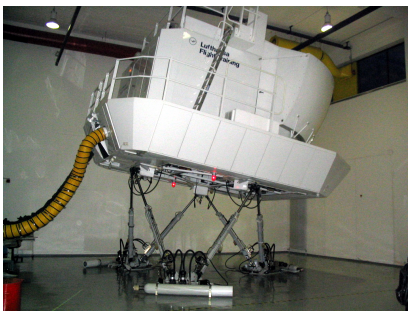
2. As quadrotors became more and more affordable and dependable, the number of real-world applications for quadrotors has grown significantly. They are being used for aerial photography, agriculture, surveillance, inspection tasks, in addition to many other uses as well.

3. Quadrotors are quite agile and maneuverable during flight, especially when compared to other types of UAVs.

However, one of the main challenges in the quadrotors community is the capability to design control and planning methods that will allow the quadrotors to carry out aggressive maneuvers. The fast dynamics associated with typically small dimensions of such agile quadrotors, along with several aerodynamic effects that will become crucial during aggressive flight maneuvers, are just a few of the main problems that are faced during the system control design. Moreover, accurate tracking of the provided trajectory is a big issue in the case of aggressive maneuvers when the rotors are commanded high speeds and accelerations, which will cause rotors to become saturated and may also cause delays.

Parallel manipulators

A parallel manipulator is a mechanical system that consists of two connected platforms, the fixed platform and the moving platform. The latter is linked to the fixed platform thanks to at least two serial chains that are working in parallel. When compared to serial manipulators, parallel manipulators are more accurate and rigid. In addition, the ability to install the motors next to the fixed platform is a very important feature for them. Moreover, parallel manipulators can be used in a wide variety of applications that demand precision and high payload combined with high speed.[7]



(a) Gough-Stewart used for a flight-simulator application.[2]

(b) The "PAR4" 4 degrees of freedom, high-speed, parallel robot prototype.[3]

Figure 2: Two examples of parallel robots.

jpg, accessed on 01/08/2021.

[1] https://en.wikipedia.org/wiki/Stewart_platform#/media/File:Simulator-flight-compartment.jpeg, accessed on 01/08/2021.

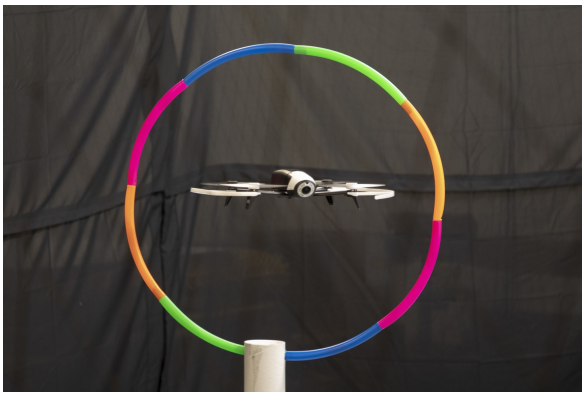
[2] https://en.wikipedia.org/wiki/Parallel_manipulator#/media/File:Prototype_robot_parall%C3%A8le_PAR4.jpg, accessed on 01/08/2021.

However, parallel manipulators are subject to singularities which can lead to big problems in the robot workspace in case they were not handled correctly. Thus, the study of the singular configuartions of parallel manipulators is very important. Because, even just before reaching a singularity, the performance of the parallel manipulator will decrease dramatically. Moreover, the robot may loose the ability of moving in a certain direction, gain uncontrollable motions and the mechanism could even break. The main difference between serial and parallel manipulators is that singularity configurations may also appear inside the workspace of the robot (depending on the dimensions of the robot) and not just at the boundaries of the robot workspace, which can significantly decrease the area of the robot workspace. As a result, many works have been developed by robotics researchers in order to allow parallel manipulators to safely cross these singularities by using trajectory planning and specific control methods.

The goal of this thesis

This master thesis lies at the intersection of parallel robotics and aerial robotics. The two fields may seem very different from each other. However, quadrotors can be seen as a particular case of a parallel manipulator. In fact, a parallel manipulator is made up of a wrench system, applied by the robot limbs on the moving platform. And, this wrench system will define the motion of the moving platform. In the same manner, each propeller in a quadrotor can be considered as a limb of a parallel robot and the moving platform to be controlled can be considered as the body of the drone. Specifically, the goal of this master thesis is to study a distinct class of aggressive maneuvers for quadrotors, namely flip maneuvers. By doing flip maneuvers, full rotations around one or more axes of the body of the quadrotor can be done. In addition, the quadrotor should also be able to perform the flip maneuvers in constrained environments.

(a) Quadrotor performing a triple flip.[8]



(b) Quadrotor going though a loop.[1]

Figure 3: Representation of the issues to be tackled in the master thesis.

[1] `https://newatlas.com/drones/muscle-signals-drone-control/#gallery:2`, accessed on 01/08/2021.

Outline of the work

The rest of the report is structured as follows:

Chapter 1 provides an overview of the state of the art in the control of quadrotors and will later on focus on the main control method that will be used, namely Model Predictive Control (MPC). Moreover, a literature review of MPC applications on quadrotors will be presented. Furthermore, an overview on the software used to design a MPC controller will be presented. Finally, the state of the art in flipping maneuvers will be presented.

Chapter 2 provides a detailed explanation of the quadrotor dynamics for the planar (2D) and 3D quadrotors. Morever, an Extended Kalman Filter (EKF) is then designed for the planar quadrotor case to be used in the presence of noisy measurements (states) and noisy control inputs. Finally, simulation results using MPC to reach a single waypoint and to follow circular trajectories with and without noise are presented.

Chapter 3 focuses on the trajectory generation of a flip trajectory where different optimization problems with different objective functions and initial conditions will be used in order to find the optimal flipping trajectory that satisfies the dynamic constraints of the quadrotor which will be used in the experimentation phase. Moreover, simulations with a MPC are then performed using the optimal flip trajectory.

Chapter 4 focuses on the simulations that were performed using ROS2 and Gazebo, in addition to the experimentation results.

Chapter 1

State of the art

In the following sections of this chapter, differential flatness, the general control architecture of a quadrotor, different potential control approaches (linear and nonlinear), and the main control method that will be used to control a quadrotor, namely Model Predictive Control (MPC) will be explained.

1.1 Differential Flatness

In the quadrotor community, a well-established finding is that the dynamic model of a quadrotor is differentially flat. Moreover, the control design problem in nonlinear systems will be considerably simplified. Precisely, a system with state $\mathbf{x} \in \mathbb{R}^n$ and input $\mathbf{u} \in \mathbb{R}^m$ is considered to be *differentially flat* if there exists a set of *flat outputs* $\mathbf{y} \in \mathbb{R}^m$ which have the following form:

$$\mathbf{y} = \mathbf{y}(\mathbf{x}, \mathbf{u}, \dot{\mathbf{u}}, ..., \mathbf{u}^{(p)}) \tag{1.1}$$

With,

$$\begin{cases} \mathbf{x} = \mathbf{x}(\mathbf{y}, \dot{\mathbf{y}}, ..., \mathbf{y}^{(q)}) \\ \\ \mathbf{u} = \mathbf{u}(\mathbf{y}, \dot{\mathbf{y}}, ..., \mathbf{y}^{(r)}) \end{cases} \tag{1.2}$$

As a result, the new set of variables is required to be a function of the state, the input and the derivatives of the input. Moreover, this set should also have the same dimensions as the control input. In this manner, it is possible to rewrite both the state and the input in function of the flat outputs and the derivatives of the flat outputs. This is a very useful property in underactuated systems where $m < n$, such as quadrotors, because, it will allow to generate trajectories in the lower dimensional space m, then this trajectory will be mapped into the full dimensional space n. An example of this is shown in figure 1.1 below:

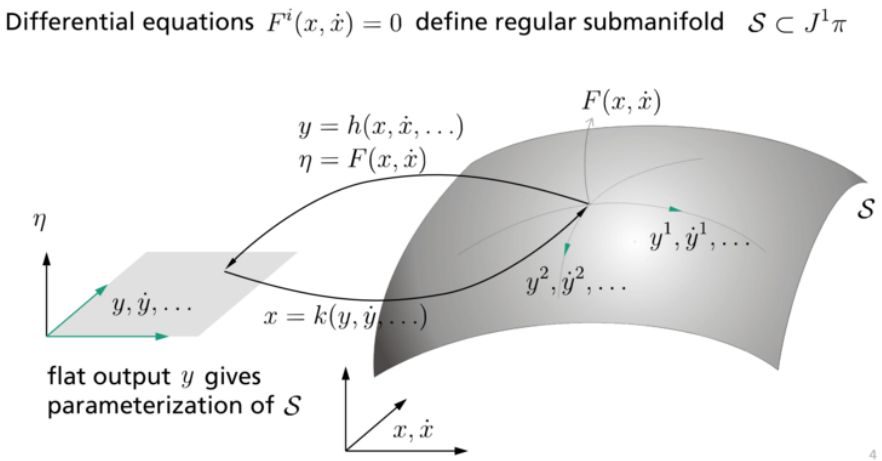


Figure 1.1: Mapping between different dimensional spaces as a result of differential flatness. [1]

Another well known example of systems is a car, in which the underactuation is the result of the nonholonomic constraints that are imposed by the wheels. So, for a car, a generated trajectory for (x, y) position of the rear-wheels is enough to specify all the viable trajectories of the system. Formal proofs that the quadrotor system is differentially flat can be found in [9], and [10] for the full model with first-order aerodynamics. The standard choice of flat outputs for the quadrotor is the coodinates of the center of mass and the yaw angle:

$$\mathbf{y} = \begin{bmatrix} x & y & z & \psi \end{bmatrix}^\intercal \tag{1.3}$$

Consequently, the problem of generating a feasible trajectory for a quadrotor then tracking it can be dimensionally decreased from a 6-dimensional space to a 4-dimensional space. By reason of the tight coupling between the rotational and translational dynamics, then defining a trajectory in function of the flat outputs $\mathbf{y}$ is sufficient to properly define the full dynamics $\mathbf{x}$.

1.2 General Control Architecture

Recently, many researchers have developed interest in the control of quadrotors. As a result, various control approaches have been proposed. The most known control architecture [10] consists of three nested control loops, as shown in figure 1.2, in order to generate the suitable motor commands to follow the desired signal. This controller is known as the cascaded controller. This strategy assumes that the attitude dynamics of a quadrotor are much faster than the translational dynamics.

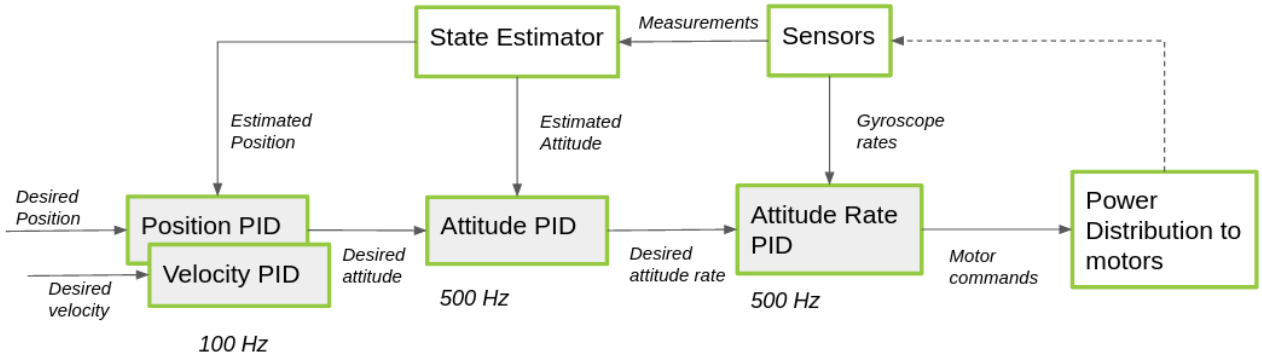


Figure 1.2: Cascaded PID controller that is present on the crazyflie 2.1 quadrotor [2].

In this case, the position and velocity controller, attitude controller and attitude rate controller are all PID controllers. And, it is evident that the attitude dynamics (who's attitude and attitude rate controllers operate at a frequency of 500Hz) are considered to be much faster than the translational dynamics (who's position and velocity controllers operate at a frequency of 100Hz). Moreover,

Attitude Rate PID controller directly controls the attitude rate. It receives the gyroscope rates after they have been filtered and uses the error between the desired attitude rate and current attitude rate, and outputs the motor commands which are then directly sent to the power distribution.

Attitude PID controller directly controls the attitude of the drone. It takes the estimated attitude from the state estimator and uses the error between the desired attitude and the current attitude to output the desired attitude rate.

Position and Velocity PID controller is the most outerloop of the cascaded PID controller. It receives the position or the velocity input from the high level commander which are then handled to output the desired attitude.

1.3 General Control Approaches

1.3.1 Method of Linearization

By using extreme assumptions, it is feasible to apply linear control techniques in order to control a quadrotor ([11], [12]). Particularly, this can be made by doing a linearization of the full dynamic model around an equilibrium point $\overline{\mathbf{x}}$ and by using the assumption that the vehicle is only capable of oscillating lightly around the hover point. It is very easy to observe that a feasible equilibrium is provided by a configuration where the center of mass is at a random position $\overline{\mathbf{r}}$ and all the other elements of the state are set to zero. So, the nominal input $\mathbf{u} = \overline{\mathbf{u}}$ to sustain such equilibrium can be assessed as the thrust that is required to compensate the gravity force:

$$\overline{\mathbf{u}} = \begin{bmatrix} f \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} mg \\ \mathbf{0_{3\times 1}} \end{bmatrix} \tag{1.4}$$

At this stage, the complete non-linear dynamics that have the form :

$$\dot{\mathbf{x}} = \overline{\mathbf{f}}(\overline{\mathbf{x}}, \overline{\mathbf{u}}) \tag{1.5}$$

can now be linearized around the hover point $(\overline{\mathbf{x}}, \overline{\mathbf{u}})$ as shown below.

$$\dot{\mathbf{x}} = \left[\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\right]_{(\bar{\mathbf{x}},\bar{\mathbf{u}})} \mathbf{x} + \left[\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\right]_{(\bar{\mathbf{x}},\bar{\mathbf{u}})} \mathbf{u} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{1.6}$$

It can be demonstrated that both matrices $\mathbf{A}$ and $\mathbf{B}$ can be used to determine a linear system that is both controllable and observable [11]. Thus, any control technique that is linear can now be used on the quadrotor in order to keep it around a desired equilibrium point, such as optimal LQR/LQG [13, 14] control or simple PD or PID controller [15, 16].

1.3.2 Nonlinear Control Methods

In order to perform more complex tasks and follow aggressive trajectories, nonlinear control methods are required. A comprehensive literature review on this topic is beyond the scope of this work and several works can be found, such as [17], which provides a general overview on nonlinear control of quadrotors. However, some nonlinear control methods deserve to be mentioned due to their extensive use and applications:

Sliding Mode control It is a control technique that is nonlinear presenting exceptional attributes of robustness,accuracy, easy tuning and execution. The aim of SMC systems is to drive the system states to a specific surface in the state space, called *"sliding surface"*. Upon reaching the sliding surface, sliding mode control allows the states to remain on the close neighborhood of the sliding surface. Therefore, the sliding mode control consists of a controller design with two parts. The first part contains the design of a sliding surface in order for the sliding motion to fulfill design requirements. The second deals with selecting a control law that makes the switching surface interesting with respect to the system state [18]. There exist two main benefits of sliding mode control. Firstly, the behavior of the dynamics of the system can be changed according to a specific selection of the sliding function. Secondly, the response of the closed loop system becomes completely insensitive to some special uncertainties. This principle goes beyond bounded model parameter uncertainties, interference and non-linearity. In a practical sense, SMC allows the control of nonlinear processes that are affected by external noise and heavy model uncertainties. The most important principles of SMC are shown in the following

significant references [18, 19, 20]. Researchers have also studied the problems appearing in the practical execution of this class of techniques. [21] Interested readers can refer to the book [22] which presents a very modern overview of the most promising current line of theoretical and practical research in the domain.

Backstepping control The main idea is to divide the system into successive subsystems and to apply a recursive algorithm which will stabilize each subsystem after the other [23]. However, this method is not robust, but it is computationally fast. In order to handle disturbances, Fang et al. [24] implemented an integral backstepping control law, in which the integral term was shown to reduce steady state errors and the response time of the system greatly.

Adaptive control This method is required when the parameters that are characterizing the system contain errors or are unknown. This type of control algorithms contains a parameter adaptation law, which is enclosed in the control to track the desired trajectory of the system, even if the model of the system is not completely known. For instance, Diao et al. [25] obtained good performance even though the inertial parameters of the quadrotor and the aerodynamic coefficients were not perfectly known. This method is convenient in some cases, such as the existence of unpredictable wind [26] or pick-and-place applications with small loads.

In the next section, the main control method that will be used in the master thesis will be explained.

1.4 Model Predictive Control

1.4.1 General Idea

There exist "open-loop" methods [27] in which the control input sequence $\mathbf{u}(t)$ is designed using a model of the system and a set of constraints. However, the problem with this approach is that modeling errors and noise are not taken into consideration. So, these inputs will not necessarily generate the desired response from the system. Because of that, a *"closed-loop"* strategy is required in order to cancel out these errors. So, an approach that can be used is called *"Model Predictive Control"* (MPC). This method is also known as *"receding horizon control"* [3] because the *"prediction horizon"* (which is a finite horizon) translates forward by one time step after the current optimization problem is solved . In short, MPC is a *feedback control* algorithm which uses a model of the system to predict the future outputs of the system and it solves an optimization problem on-line in order to select an optimal control.

Basic strategy The basic strategy of MPC is the following:

- At time step k, the system model and an optimizer will be used in order to design a sequence of control inputs

$$\mathbf{u}(k|k), \mathbf{u}(k+1|k), \mathbf{u}(k+2|k), \mathbf{u}(k+3|k), ..., \mathbf{u}(k+N|k)$$

  starting from the current state $\mathbf{x}(k)$ over a prediction horizon N.

- Only the first step of the sequence of control inputs will be applied on the system.

- The processes above are then iterated for time $(k+1)$ at state $\mathbf{x}(k+1)$.

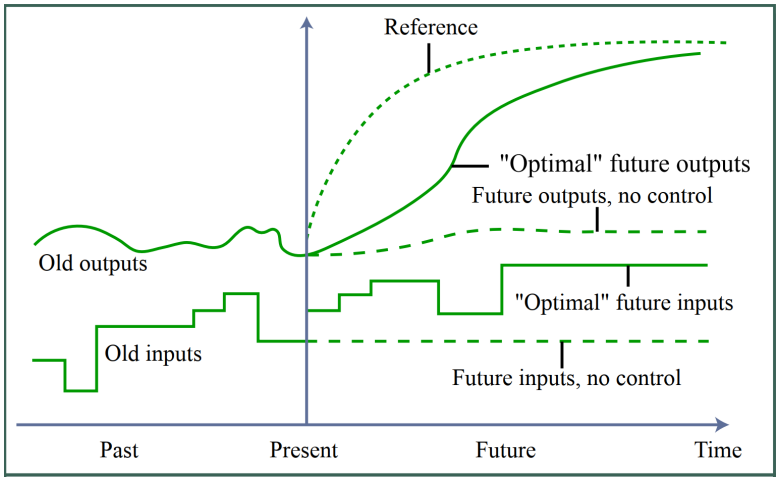


Figure 1.3: Basic idea of MPC [3].

It should be noted that the control algorithm of MPC is based on numerically solving an optimization problem at each time step. And, it is a constrained optimization in general.

Advantages and drawbacks of MPC There are several advantages when using MPC:

- MPC is able to control multi-input multi-output (MIMO) systems which might have interactions between their inputs and outputs.

- MPC explicitly accounts for the constraints that are imposed on the system. So, it does not just design a controller to keep the system away from the constraints.

- MPC can easily handle nonlinear dynamics and time-varying plant dynamics, because the controller is explicitly a function of the model of the system which can be modified in real-time.

The main drawback of MPC is that it usually requires a powerful and fast processor with a large memory in order to properly solve the problem at hand. This is because MPC solves an optimization problem at each time step.
There have been many commercial applications of MPC starting from the early 1970s in the process industry. The table 1.1 below shows the different companies that have used MPC in different industries in 2014.

Table 1.1: The number of MPC applications in different industries in 2014 [5].

Application	Aspen	Honeywell	Adersa	CCI	Pavilion	Total
Refining	950	300	290	-	15	1555
Chemicals	437	55	12	21	25	550
Food	-	-	48	-	14	62
Pulp paper	21	39	-	-	3	63
Gas and air	11	13	-	24	-	48
Polymer	5	-	-	-	22	27
Utilities	7	9	-	6	-	22
Other	39	-	51	6	-	96
Total	1470	416	401	57	79	2423

However, as computational power has increased throughout the years thanks to the advancement of technology, there has been a renewed curiosity in applying this control approach to systems with fast dynamics for which the computational complexity is significantly larger when compared to the industrial applications for which computational complexity was not a concern (since MPC was applied on systems with slow dynamics in that case).

1.4.2 Design Parameters

The different parameters that can be tuned in a MPC controller ([28, 29, 30]) are the following:

- The sample time T_s.

- The prediction horizon N.

- The control horizon m.

- The constraints.

- The weights.

Choosing the proper values for the parameters stated above is very important since they affect the performance of the controller and the computational complexity of the MPC algorithm.

Sample Time T_s The sample time determines the rate at which the controller executes the control algorithm.

- If T_s is too large, then when disturbances occur, the controller will be unable to react to the disturbances quickly enough.
- If T_s is too small, the controller will have much faster reaction times to disturbances and set-point changes. However, this comes at the cost of an excessive computational load.

In order to find reasonable balance between controller performance and computational effort, the general recommendation is to have between 10% and 25% of the minimum desired closed-loop response time .

Prediction horizon N The prediction horizon N (sometimes referred to with the variable p, however, N will be used instead for the remaining of this report) is the number of predicted future time steps of the system. It shows how far the controller predicts into the future. Thus, a prediction horizon must be chosen in such a way that it covers the significant dynamics of the system. However, it should be noted that a large prediction horizon should not be selected, since unexpected phenomenons could occur that may affect the dynamics of the system, which will cause a waste of computational power. The general recommendation is to increase N until additional increases will have little impact on the performance. The maximum N is the number of control intervals needed for the open-loop step response of the system to become infinite. However, having $N > 50$ is hardly ever required unless T_s is very small.

Control horizon m The control horizon is the set of future actions which will lead to the predicted plant output. It represents the number of control moves until time step m. After the first m steps, the remaining inputs will remain constant as shown in figure 1.3 for the *"Optimal"* future inputs. Moreover, each control input element in the control horizon is a free variable that will be computed by the optimizer. Thus, the smaller the control horizon, the fewer the computations. However, setting $m = 1$ may not give the best possible output for the system. And, similarly to the prediction horizon, if the control horizon is increased, this will lead to better predictions at the cost of increasing the computational complexity. Moreover, the general recommendation for the control horizon is to keep it much smaller than the prediction horizon, because:

- A smaller control horizon m will lead to less variables to optimize in the QP that will be solved at each control input interval. This will encourage quicker computation times.
- If delays are considered, then having $m < N$ is mandatory. Otherwise, some control input elements within the control horizon may not have any effect on the plant outputs before the prediction horizon ends, which will lead to a QP Hessian matrix which is singular.
- Having a small value for m encourages having an internally stable controller. However, this is not guaranteed.

Constraints A model predictive controller can integrate constraints on the inputs, the rate of change of the inputs and the outputs. In addition, the constraints can be *"soft"* constraints or *"hard"* constraints. *"Soft"* constraints represent constraints that are allowed to be violated if the MPC deems it to be necessary in order to find a solution for the control problem at hand. In addition, it should be noted that *hard* constraints are a set of constraints that cannot be violated by the MPC. However, applying hard constraints on both the inputs and outputs at the same time may cause conflicts to occur between the constraints, which may lead to an

unfeasible solution. Moreover, the general recommendation is to use soft constraints on the outputs and to avoid having hard constraints on both the inputs together with the rate of change of the inputs.

Weights Model Predictive Control could have many goals. A possible goal could be to have the outputs converge to their set-points as fast as possible. Another goal could be to have smooth control inputs in order to avoid aggressive control maneuvers. So, in order to achieve a balanced performance between these two competing goals, the input rates and the outputs can be weighted relatively to each other. It is also possible to adjust relative weights within the input rates and the outputs.

1.4.3 Basic formulation of a MPC problem

For a given set of plant dynamics which is first assumed to be linear:

$$\begin{cases} \mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k) \\ \mathbf{z}(k) = C\mathbf{x}(k) \end{cases} \tag{1.7}$$

and a cost function as follows:

$$J = \sum_{j=0}^{N} \{\|\mathbf{z}(k+j|k)\|_{R_{zz}} + \|\mathbf{u}(k+j|k)\|_{R_{uu}}\} + F\mathbf{x}(k+N|k)) \tag{1.8}$$

With:

- $\|\mathbf{z}(k+j|k)\|_{R_{zz}}$ is the weighted L^2-norm of the state, so it is expressed as follows:

$$\|\mathbf{z}(k+j|k)\|_{R_{zz}} = \mathbf{z}(k+j|k)^\intercal R_{zz}\mathbf{z}(k+j|k)$$

- $\|\mathbf{u}(k+j|k)\|_{R_{uu}}$ is the weighted L^2-norm of the control input sequence, so it is expressed as follows:

$$\|\mathbf{z}(k+j|k)\|_{R_{zz}} = \mathbf{z}(k+j|k)^\intercal R_{zz}\mathbf{z}(k+j|k)$$

- $F\mathbf{x}(k+N|k))$ is a terminal cost function which can also have weights if required.

It should be noted that if $N \to \infty$, and there are no additional constraints on $\mathbf{z}$ and/or $\mathbf{u}$, then the problem falls back to the discrete LQR problem [31]. Moreover, when limits are added on $\mathbf{x}$ and/or $\mathbf{u}$, then the general solution cannot be found anymore in analytical form, and it has to be solved numerically.
Furthermore, solving for a very long sequence of control inputs is irrelevant if the model used for the computations is expected to be erroneous or there are disturbances applied on the system, since only the first element of the optimized control sequence will be implemented. This is why MPC is generally designed by using a relatively small N.

Typical problem statement For a finite N and with $F = 0$ the problem can be expressed as follows:

$$\begin{aligned} \min_{u} \quad & J = \sum_{j=0}^{N} \{\|\mathbf{z}(k+j|k)\|_{R_{zz}} + \|\mathbf{u}(k+j|k)\|_{R_{uu}}\} \\ \text{s.t.} \quad & \mathbf{x}(k+j+1|k) = A\mathbf{x}(k+j|k) + B\mathbf{u}(k+j|k), \\ & \mathbf{x}(k|k) \equiv \mathbf{x}(k), \\ & \mathbf{z}(k) = C\mathbf{x}(k+j|k), \\ & |\mathbf{u}(k+j|k)| \leq u_{max} \end{aligned} \tag{1.9}$$

1.4.4 MPC applications on quadrotors

There have been numerous applications of MPC on quadrotors. The main MPC applications on drones can be separated into two different categories:

1.4.4.1 Centralized MPC

In this type of application, the MPC controller is a signle structure that encapsulates both the translational dynamics and the rotational dynamics. It takes the trajectory waypoints as inputs, and directly outputs the control inputs required to control the quadrotor. This type of implementation of the MPC controller produces the least amount of errors. However, this comes at a cost of a larger computational load.

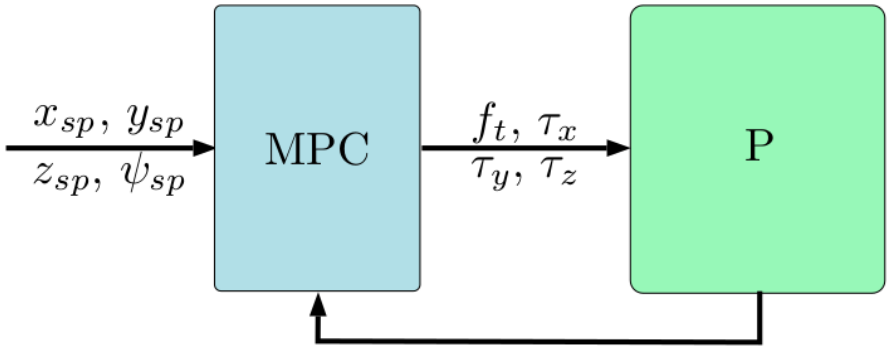


Figure 1.4: Example of a centralized MPC control loop [4].

As can be seen in figure 1.4, which represents a centralized control loop structure for a basic MPC, the right block is the plant which represents the quadrotor dynamics. The MPC bloc taks as input the desired trajectory, and outputs control inputs to allow the quadrotor to follow the trajectory.
It should be noted that different versions of the MPC controller shown in figure 1.4 can exist. For example, another version of the MPC could be that the state variables of roll and pitch angles can be added as outputs. This is mainly implemented in order to ensure that the quadrotor does not take extreme values in roll and pitch angles, which can potentially destabilize it, because no restrictions are being used for these angles otherwise.

1.4.4.2 Non-centralized MPC

In this type of application, the state-space of the system will be split to obtain two different controllers in a master-slave structure. This more simplified structure allows the implementation of MPC in different embedded systems, each one with less computational load when compared to the complete structure of a centralized MPC. For example, the simplified slave structure can be used to control the attitude of the quadrotor, whereas the master structure can be used to control the position and direction of the quadrotor. In addition, this splitting allows different sample times for the master and slave loops [32]. Furthermore, an implementation of non-centralized MPC could be of having both master and slave structures being MPC controllers. An example of this non-centralized MPC implementation is shown in figure 1.5 below:

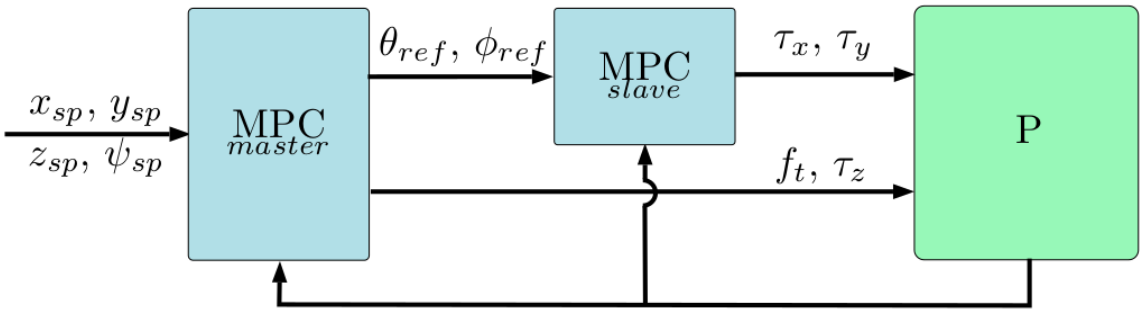


Figure 1.5: Example of a non-centralized MPC-MPC control loop [4].

Another implementation is using a MPC controller for the master structure and pairing it with the slave structure that is using a controller with a different control law, such as a PD-P controller, in order to further reduce the computational load. An example of this non-centralized MPC implementation is shown in figure 1.6 below:

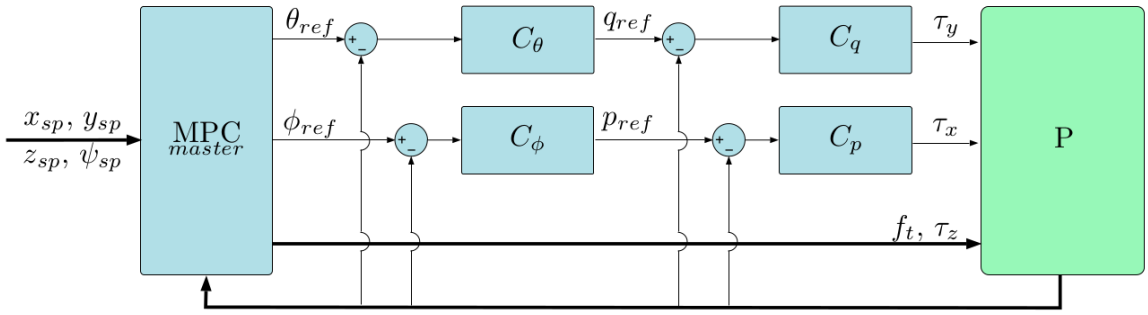


Figure 1.6: Example of a non-centralized MPC-PD-P control loop [4].

However, since the control law in this master thesis will be entirely focused on MPC controllers, and since the task will be to follow aggressive trajectories as perfectly as possible, then the centralized MPC was chosen since it is the simplest implementation of MPC and it results in the least amount of errors when compared to non-centralized MPC implementations. In addition, the computational load of MPC will not be a problem, since the MPC controller will not be running on the quadrotor itself. All the computations will be done on a computer which will have considerably larger computational power than the processor of the quadrotor. And, the control inputs will be sent to the quadrotor via radio signals. However, it should be noted that the quadrotor will be susceptible to delays.
In the next section, the software used to implement the MPC controller is presented.

1.5 acados – fast embedded optimal control solvers

1.5.1 The difference between acados and other embedded optimal control solvers

A big role in bringing MPC to real-time applications is played by the implementation of efficient embedded optimal control methods. This gave rise to software packages such as:

- `MPT`: for explicit MPC [33].
- `qpOASES`: an active-set solver for quadratic programming [34].
- `FORCES`: an interior-point solver for quadratically constrained QP and NLPs with optimal control structure [35].
- `ACADO Code Generation tool`: for tailored SQP based NMPC solvers [36].

One challenge in developing software for embedded optimal control lies in the trade-off between flexibility, memory usage and speed. Moreover, many of the software packages mentioned above are based on *automatic code generation*. This allows to have self-contained and efficient routines. However, the size of the problem and the choice of algorithms are usually fixed for one specific optimal control instance, which will result in a loss of flexibility. In addition, generated code is often hard to read, thus hard to debug. Finally, it is hard to predict if an automatic code optimization strategy is beneficial or counterproductive for some types of problems. This means that in some cases, a programmer can create faster and more efficient code.
For these reasons, `acados` does not rely on automatic code generation to perform linear algebra operations. Instead, the high-performance linear algebra package `BLASFEO` is used.
Furthermore, other embedded optimal control softwares use global data in order to not sacrifice execution speed and memory. The resulting codebase is difficult to understand, maintain and extend. However, `acados` organizes the code in a modular way, with formal interfaces between the different algorithmic components. This makes it easier to interchange solvers, routines and libraries needed for the embedded control algorithm.
Finally, `acados` uses `CasADi` modeling language instead of `Mathematics`, `sympy` or `MATLAB Symbolic Toolbox`. Many of these softwares make use of expression trees to represent mathematical funcions, which could lead to a large code size, high memory usage and slow evaluation of high-order derivatives for non-trivial models. Whereas `CasADi` is based on expression graphs. This usually leads to smaller and faster code which make it more suitable for embedded applications.
In addition, it is free and open-source software. In summary, `acados` offers the following:

- It contains efficient optimal control algorithms implemented in C.
- it has a modular architecture enabling rapid prototyping of solution algorithms.
- It interfaces to `C++`, `Python` and `MATLAB`.
- It uses the high-performance linear algebra package `BLASFEO`.
- It is compatible with `CasADi` expressions.
- It is deployable on a variety of embedded devices.
- It is a free and open-source software.

1.5.2 Algorithm components for embedded nonlinear optimal control

1.5.2.1 Nonlinear Optimal Control

The nonlinear optimal control problems that are typically solved by nonlinear MPC (NMPC) have the following form in continuous-time:

$$\begin{aligned} \min_{x(\cdot),z(\cdot),u(\cdot)} \quad & \int_{t=0}^{T} l(x(t),z(t),u(t))dt + M(x(T)) \\ \text{s.t.} \quad & x_0 = \bar{x}_0 \\ & 0 = f(\dot{x}(t),x(t),z(t),u(t)) \qquad t \in [0,T], \\ & 0 \geq g(x(t),z(t),u(t)) \qquad t \in [0,T] \end{aligned} \tag{1.10}$$

With:

- l: Lagrange term or running cost.
- M: Mayer term or terminal cost.
- x: differential states with $x \in \mathbb{R}^{n_x}$.
- z: algebraic variables with $z \in \mathbb{R}^{n_z}$.
- u: control inputs with $u \in \mathbb{R}^{n_u}$.
- f: Dynamics (as implicit differential-algebraic equations).
- g: Nonlinear path constraints.

1.5.2.2 Multiple Shooting Discretization

In `acados`, the nonlinear OCP is discretized with a **multiple shooting** approach. The following elements are introduced:

- Time grid: $[t_0, t_1, \ldots, t_N]$ with $t_k < t_{k+1},\ k = 0, \ldots, N-1$
- Discretized state variables: $x_0, x_1, \ldots, x_N$
- Algebraic variables $z_0, z_1, \ldots, z_{N-1}$
- Controls $u_0, u_1, \ldots, u_{N-1} \Rightarrow$ piecewise constant parametrization is used.

The numerical simulation routine on each time interval $[t_k, t_{k+1})$ can be written as:

$$\begin{bmatrix} x_{k+1} \\ z_k \end{bmatrix} = \phi_k(x_k, u_k), \quad k = 0, 1, \ldots, N-1$$

By performing multiple shooting discretization on the continuous-time optimal control problem formulation in equation (1.10), this will result in the formulation of the nonlinear program (NLP) as shown below.

1.5.2.3 General Form of the Resulting Nonlinear Program

The general form of the resulting nonlinear program that can be handled by `acados` is:
The resulting (general form) NLP problem becomes:

$$
\begin{aligned}
\min_{\substack{x_0,\ldots,x_N \\ u_0,\ldots,u_{N-1} \\ z_0,\ldots,z_{N-1} \\ s_0,\ldots,s_N}} \quad & \sum_{k=0}^{N-1} l_k(x_k, u_k, z_k) + M(x_N) + \sum_{k=0}^{N} \rho_k(s_k) \\
\text{s.t.} \quad & \begin{bmatrix} x_{k+1} \\ z_k \end{bmatrix} = \phi_k(x_k, u_k) && k = 0, 1, \ldots, N-1, \\
& 0 \geq g_k(x_k, z_k, u_k) - J_{s,k} s_k && k = 0, 1, \ldots, N-1, \\
& 0 \geq g_N(x_N) - J_{s,N} s_N, \\
& 0 \leq s_k && k = 0, 1, \ldots, N-1.
\end{aligned} \tag{1.11}
$$

This NLP can be solved by any general-purpose NLP solver, like `IPOPT` [37]. But, `acados` will use embedded optimal control methods to solve the NLP which are better suited in a real-time setting.

Chapter 2

Actual work

When dealing with rectangled triangles (see Figure 2.1) I sometimes used this theorem from [38]:

$$a^2 + b^2 = c^2 \tag{2.1}$$

The demonstration is in Appendix A.

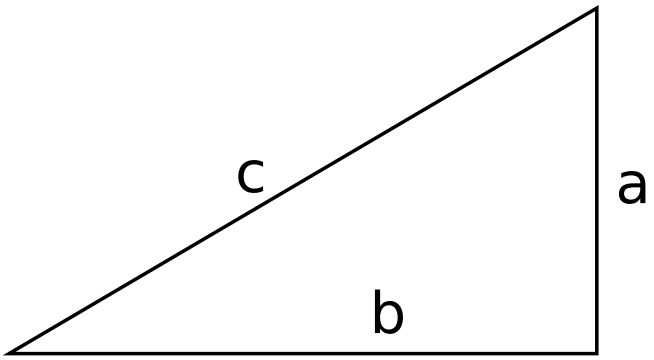


Figure 2.1: A triangle with letters

Chapter 3

Experiments

When trying to draw a rectangled triangle, my program comes up with Figure 3.1 that is neither rectangled nor a triangle.

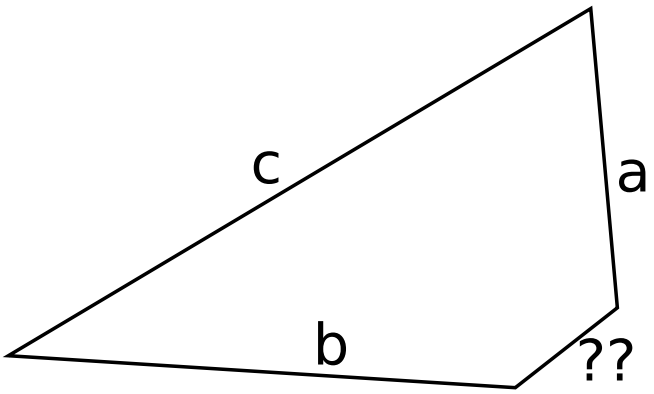


Figure 3.1: Triangle drawn by my program. Note the 4th side.

Unless there is a bug in my program, which is unlikely, this research indicates that the whole theory on triangles having 3 sides has been wrong for years, maybe decades.

Conclusion

Appendix A

Proof of theorem 2.1

Proof. (2.1) was already demonstrated in [39]. □

Bibliography

[1] K. R. Klemens Fritzsche, Yuhang Guo, "Some remarks concerning differential flatness and tangent systems." *23rd International Conference on System Theory, Control and Computing (ICSTCC),10.1109/ICSTCC.2019.8886157*, pp. 173–179, 2019.

[2] "Controllers in the crazyflie." [Online]. Available: https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/sensor-to-control/controllers/

[3] Jonathan How, *Principles of Optimal Control.* MIT, 2008.

[4] R. S. Alvarez-Valle and P. S. Rivadeneira, "Design of predictive control structures to track trajectories for multi-rotor unmanned aerial vehicle," in *2019 IEEE 4th Colombian Conference on Automatic Control (CCAC)*, 2019, pp. 1–6.

[5] K. Kozak, "State-of-the-art in control engineering," *Journal of Electrical Systems and Information Technology*, vol. 1, p. 1–9, 05 2014.

[6] S. Bouabdallah, "Design and control of quadrotors with application to autonomous flying," Ph.D. dissertation, École Polytechnique Fédérale de Lausanne, 2007.

[7] J. Gallardo-Alvarado, *An Overview of Parallel Manipulators.* Manhattan, NY: Springer, 2016.

[8] S. Lupashin and R. D'Andrea, *Adaptive fast open-loop maneuvers for quadrocopters.* Manhattan, NY: Springer, 2012.

[9] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525.

[10] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories." *IEEE Robot. Autom. Lett*, pp. 620–626, 2018.

[11] F. Sabatino, "Quadrotor control: modeling, nonlinear control design, and simulation." Master's thesis, 2015.

[12] S. Bouabdallah, A. Noth and R. Siegwart., "Pid vs lq control techniques applied to an indoor micro quadrotor." *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 2451–2456, 2004.

[13] Ian D. Cowling, Oleg A. Yakimenko, James F. Whidborne, and Alastair K. Cooke., "A prototype of an autonomous controller for a quadrotor uav." *2007 European Control Conference (ECC)*, pp. 4001–4008, 2007.

[14] Ly Dat Minh and Cheolkeun Ha., "Modeling and control of quadrotor mav using visionbased measurement." *2010 International Forum on Strategic Technology (IFOST)*, pp. 70–75, 2010.

[15] Keun Uk Lee, Han Sol Kim, Jin Bae Park, and Yoon Ho Choi., "Hovering control of a quadrotor." *2012 12th International Conference on Control, Automation and Systems (ICCAS)*, pp. 162–167, 2012.

[16] Bora Erginer and Erdinc Altug., "Modeling and pd control of a quadrotor vtol vehicle." *2007 IEEE Intelligent Vehicles Symposium*, pp. 894–899, 2007.

[17] A. Zulu and SamuelJohn, "A review of control algorithms for autonomous quadrotors," 2016.

[18] V. Utkin, "Variable structure systems with sliding modes," *IEEE Transaction on Automatic Control*, pp. 212–222, 1977.

[19] R. A. DeCarlo, S. H. Zak, and G. Mathews, "Variable structure control of nonlinear multivariable systems: A tutorial," *Proceedings of the IEEE, vol. 76, No. 3*, 1988.

[20] J. Y. Hung, W. Gao, and J. Hung, "Variable structure control: A survey," *IEEE Trans. on Industrial Electronics, vol. 40, No. 1*, 1993.

[21] K. Young, V. Utkin, and . Özgüner, "A control engineer's guide to sliding mode control," *IEEE Transactions on Control Systems Technology 7*, pp. 328–342, 1999.

[22] G. Bartolini, L. Fridman, A. Pisano, and E. Usai, *Modern Sliding Mode Control Theory. New Perspectives and Applications.* Springer Lecture Notes in Control and Information Sciences, 2008.

[23] T. Madani and A. Benallegue, "Backstepping control for a quadrotor helicopter." *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3255–3260, 2006.

[24] Z. Fang and W. Gao, "Adaptive integral backstepping control of a micro-quadrotor." *2nd International Conference on Intelligent Control and Information Processing (ICICIP)*, pp. 910–915, 2011.

[25] C. Diao, B. Xian, Q. Yin, W. Zeng, H. Li, and Y. Yang, "A nonlinear adaptive control approach for quadrotor uavs," *2011 8th Asian Control Conference (ASCC)*, pp. 223–228, 2011.

[26] G. Antonelli, E. Cataldi, P. Robuffo Giordano, S. Chiaverini, and A. Franchi, "Experimental validation of a new adaptive control scheme for quadrotors mavs," *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2439–2444, 2013.

[27] R. Gabasov, F. Kirillova, and N. Balashevich, "Open-loop and closed-loop optimization of linear control systems," *Asian Journal of Control*, vol. 2, pp. 155 – 168, 09 2000.

[28] MathWorks, "Choose sample time and horizons," https://www.mathworks.com/help/releases/R2018a/mpc/ug/choosing-sample-time-and-horizons.html?s_eid=PSM_15028, 2018, from the Model Predictive Control Toolbox.

[29] ——, "Specify constraints," https://www.mathworks.com/help/releases/R2018a/mpc/ug/specifying-constraints.html?s_eid=PSM_15028, 2018, from the Model Predictive Control Toolbox.

[30] ——, "Tune weights," https://www.mathworks.com/help/mpc/ug/tuning-weights.html?s_eid=PSM_15028, 2018, from the Model Predictive Control Toolbox.

[31] S. Kostova, I. Ivanov, L. Imsland, and N. Georgieva, "Infinite horizon lqr problem of linear discrete time positive systems," *Proceeding of the Bulgarian Academy of Sciences*, vol. 66, 01 2013.

[32] K. Ling, W. Bingfang, H. Minghua, and Z. Yu, "A model predictive controller for multirate cascade systems," in *Proceedings of the 2004 American Control Conference*, vol. 2, 2004, pp. 1575–1579 vol.2.

[33] M. Herceg, M. Kvasnica, C. Jones, and M. Morari, "Multi-Parametric Toolbox 3.0," in *Proc. of the European Control Conference*, Zürich, Switzerland, July 17–19 2013, pp. 502–510, http://control.ee.ethz.ch/~mpt.

[34] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpoases: a parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, p. 327–363, 2014.

[35] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones, "Efficient interior point methods for multistage problems arising in receding horizon control," *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, 2012.

[36] ——, "Efficient interior point methods for multistage problems arising in receding horizon control," *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, 2012.

[37] "Ipopt documentation." [Online]. Available: https://coin-or.github.io/Ipopt/

[38] O. S. Pythagoras, "Theorem," *Some old journal*, vol. 1, no. 1, Feb. -580.

[39] O. A. Euclides, "Elements," *Self-published*, vol. 1, no. 1, Feb. -300.