ÉCOLE CENTRALE DE NANTES

MASTER CORO-IMARO
"CONTROL AND ROBOTICS"

2020 / 2021

Bibliography Report

Presented by

Elie Hatem

On January 15, 2021

Making Flips With Quadrotors In Constrained Environments

Jury

Evaluators:	Dr. Olivier Kermorgant	Associate Professor (ECN)
	Dr. Ina Taralova	Associate Professor (ECN)
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 to perform the aggressive maneuver 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. 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.

Secondly, I would also like to thank Dr. Ina Taralova for providing me with the valuable knowledge to create a proper bibliography.

I would like to thank my patient and understanding girlfriend Glysa, who has been with me for more than 5 years. Thank you for all the love, support and comfort that you have given me in these stressful 2 years. I hope that this Master degree will allow us to have a better future together.

I would like to thank my family as well: my parents Naji and Yolla, my sister Rebecca, my uncle and his wife Fadi and 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 degree. They have always been proud and encouraging.
I would not be here if it wasn't for them.

Notations

b	thrust factor
l	horizontal distance: from the center of the propeller to the CoG
Ω	spinning speed of a propeller
CT	continuous time
DT	discrete time
R_O	Working frame related to the environment
R_U	UAV frame
X	A vector expressed in its proper frame
${}^R[X]$	X expressed in the frame R

Abbreviations

UAV	unmanned aerial vehicle
IGE	in ground effect
OGE	out of ground effect
CoG	center of gravity
i.s.L.	in sense of Lyapunov
lpd	locally positive definite
lpsd	locally positive semi-definite
lnd	locally positive negative definite

List of Figures

List of Tables

Contents

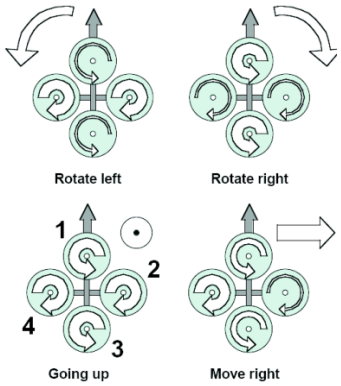
Introduction

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 fig. 1a. The quadrotor is typically built in a way such that a pair of opposite rotors rotate clockwise, whereas the other pair of rotors rotates in counter-clockwise. The attitude and the position of the drone are controlled by changing the spinning speed of the rotors. An example is shown 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.[2]

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.jpg`, accessed on 01/08/2021.

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.

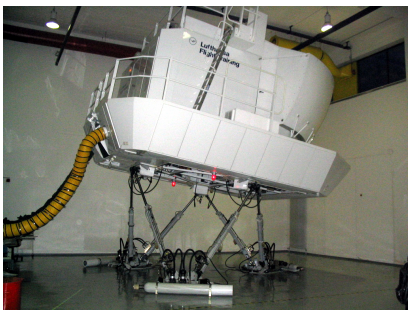
2. As quadrotors became more and more affordable and dependable, the number of real-world applications for quadrotors has grown significantly. They are being used for aerial photography, agriculture, surveillance, inspection tasks, in addition to many other uses as well.

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 parallel manipulators. Moreover, parallel manipulators can be used in a wide variety of applications that demand precision and high payload combined with high speed.[14]



(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.

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

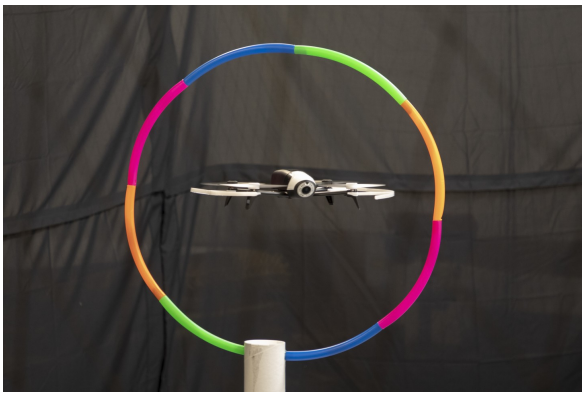
[2] https://en.wikipedia.org/wiki/Parallel_manipulator#/media/File:Prototype_robot_parall%C3%A8le_PAR4.jpg, accessed on 01/08/2021.

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 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 multi-flip maneuvers. By doing multi-flip maneuvers, full rotations around one or more axes of the body of the quadrotor can be done. In addition, the quadrotor must also perform the multi-flip maneuvers in a constrained environment.

(a) Quadrotor performing a triple flip.[15]



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

Figure 3: Representation of the issues to be tackled in this 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 bibliography is structured as follows:

Chapter 1 is devoted to introduce the system modeling of quadrotors. Specifically, a simplified dynamic model of the quadrotor will be presented by using Euler-Lagrange formalism. Then, moving on from the simple dynamic model, a more detailed dynamic model will be presented by using the Newton-Euler formalism. Finally, the state-space model of the quadrotor will be derived.

Chapter 2 provides an overview of state of the art in quadrotor control in addition to introducing the different potential control methods that can be used during the master thesis in order to properly control the quadrotor.

Chapter 3 provides detailed explanations of how multi-flip maneuvers can be handled. Then, the link between a quadrotor performing a flip and a parallel robot crossing a singularity will be explained. In the end, a literature review is provided in order to show how the problem is tackled by different researches.

Chapter 4 is devoted to trajectory optimization. By using trajectory optimization, it will be possible to create feasible trajectories for quadrotors to perform the aggressive maneuvers in constrained environments.

Chapter 1

System Modeling

The principal objective of this chapter is to demonstrate the dynamic model of the quadrotor. In this case, both the Euler-Lagrange and the Newton-Euler formalism are used for expressing the mathematical and physical model of the quadrotor platform. However, the notion of position and orientation will be explained first. The lecture notes of Dr. Fantoni is considered as the reference [1].
After that, the derivation of the state space model which will be coded on the controller of the quadrotor is performed [2].

1.1 Notion of Position and Orientation

The configuration of a vehicle, namely the *pose*, includes the parameters that pertmits to describe the mobile frame R_M with respect to the working frame R_O. In addition, there are several methods to describe the rotation of a rigid body in space, for instance, Euler angles, Cardan angles, etc. [16]. The main distinction between the Euler angles and the Cardan angles is that Cardan angles express rotations around three different axes (e.g. $x-y-z$, or $x-y''-z''$), whereas Euler angles utilize the same axis for both the first and third elemental rotations(e.g., $z-x-z$, or $z-x'-z''$)
A common representation of the pose of a vehicle in 3D Euclidean space is the following:

$$q = [\ x \quad y \quad z \quad \psi \quad \theta \quad \phi\ ]^\intercal \tag{1.1}$$

The pose can be described as the transformation from the working frame related to the environment R_O to the mobile frame R_M based on the four following procedures [1]:

- The translation $\overrightarrow{OM}$ from R_O to R_M: $(M, \overrightarrow{s_0}, \overrightarrow{n_0}, \overrightarrow{a_0})$.
- The rotation $(\psi, \overrightarrow{a_0})$ where ψ is the yaw angle from $R_O(M, \overrightarrow{s_0}, \overrightarrow{n_0}, \overrightarrow{a_0})$ to $R_1(M, \overrightarrow{s_1}, \overrightarrow{n_1}, \overrightarrow{a_0})$.
- The rotation $(\theta, \overrightarrow{n_1})$ where θ is the pitch angle from R_1 to $R_2(M, \overrightarrow{s_2}, \overrightarrow{n_1}, \overrightarrow{a_1})$.
- The rotation $(\phi, \overrightarrow{s_2})$ where ϕ is the roll angle from R_2 to $R_M(M, \overrightarrow{s_2}, \overrightarrow{n_2}, \overrightarrow{a_2}) = R_M$

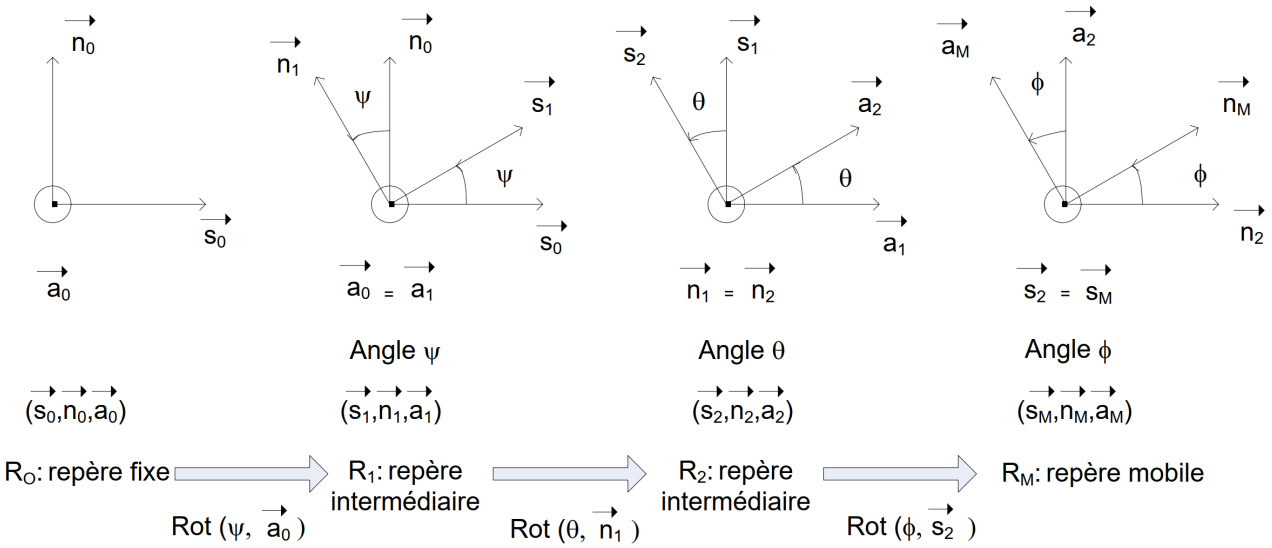


Figure 1.1: Representation of the Euler Transformations [1]

The related change of coordinates matrices are expressed as follows:

$$L = {}^{O}A_1 = Rot(\psi, \overrightarrow{a_0}) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.2}$$

$$T = {}^{1}A_2 = Rot(\theta, \overrightarrow{n_1}) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{1.3}$$

$$R = {}^{2}A_M = Rot(\phi, \overrightarrow{s_2}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{1.4}$$

with L, T and R denoting rotation matrices of roll, pitch and yaw respectively. Hereafter, $Rot(\psi, \overrightarrow{a_0})$, $Rot(\theta, \overrightarrow{n_1})$ and $Rot(\phi, \overrightarrow{s_2})$ will denote the corresponding change of coordinate matrices.
Thus, the transition matrix from the mobile frame to the fixed frame is then expressed as follows:

$$\begin{aligned} LTR = {}^{O}A_M &= Rot(\psi, \overrightarrow{a_0})Rot(\theta, \overrightarrow{n_1})Rot(\phi, \overrightarrow{s_2}) \\ &= \begin{bmatrix} \cos\psi\cos\theta & -\sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \sin\psi\sin\theta + \cos\psi\sin\theta\cos\phi \\ \cos\theta\sin\psi & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & -\sin\phi\cos\psi + \sin\psi\sin\theta\cos\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \end{aligned} \tag{1.5}$$

1.2 Inputs

The principal forces and torques that the quadrotor is subject to, are generated by the rotations of the propellers.

The forces The forces that are acting on the quadrotor are the weight and the consequent lift which is created by the 4 rotors:

$$\overrightarrow{F_\xi} = \overrightarrow{P} + \sum \overrightarrow{f_i} \tag{1.6}$$

with the weight is expressed as $\overrightarrow{P} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix}$

And, $\overrightarrow{f_i}$ is the lift resulting from every individual motor M_i given that the motors M_1 and M_3 are rotating in the counter-clockwise direction and the motors M_2 and M_4 are rotating in the clockwise direction. So, $\overrightarrow{f_i}$ is expressed as $\overrightarrow{f_i} = \begin{bmatrix} 0 \\ 0 \\ f_i \end{bmatrix} = k_f \begin{bmatrix} 0 \\ 0 \\ \omega_i^2 \end{bmatrix}$

$f_i = k_f \omega^2; i = 1, \ldots, 4$
ω_i: rotational velocity of every motor M_i
k_f: Thrust coefficient

The consequent lift is the full thrust of the UAV and is expressed by $u = \sum \overrightarrow{f_i}$. It is lying on the same line as the u_z axis of the R_U frame where the final expression of $\overrightarrow{F_\xi}$ is made of two parts . The first part (the weight) is expressed in the fixed frame and the second part (the total thrust u) is expressed in the mobile frame:

$$\overrightarrow{F_\xi} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + k_f \begin{bmatrix} 0 \\ 0 \\ \omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2 \end{bmatrix} = \overrightarrow{F_\xi} = {}^{O}\begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + k_f {}^{U}\begin{bmatrix} 0 \\ 0 \\ \omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2 \end{bmatrix} \tag{1.7}$$

The torques Regarding the torques, they are also divided into two parts. The first part is the result of the relations between the velocities of the rotors. Whereas the other part is generated by the gyroscopic effects that are resulting from the change of attitude direction of the rotors. In fact, a rotation around the u_x or u_y axis coupled to the rotation of the rotors that is executed around the u_z axis, generates a torque respectively on u_x or u_y axis.
By taking $\tau = \begin{bmatrix}\tau_1 & \tau_2 & \tau_3\end{bmatrix}^\intercal$ to be the consequent torque of the quadrotor. It is divided into two parts, namely τ_a which contains the rotation torques (roll,pitch and yaw) on the 3 axes $(\tau_\phi, \tau_\theta, \tau_\psi)$ and the torque τ_G which is the result of the gyroscopic effects of the rotors, which is usually neglected in many works.

$$\tau = \tau_a + \tau_G \tag{1.8}$$

with:

- $\tau_a = \begin{bmatrix}\tau_\phi \\ \tau_\theta \\ \tau_\psi\end{bmatrix} = \begin{bmatrix}l(f_4 - f_2) \\ l(f_3 - f_1) \\ \sum \tau_{M_i}\end{bmatrix}$
  $\sum \tau_{M_i} = \sum k_\tau \omega_i^2 = k_\tau(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2); i = 1, \dots 4$: motor torque about the axis.
  k_τ: drag coefficient.
  l: distance between the center of gravity (CoG) of the quadrotor and the axis of motor M_i.
- $\tau_G = \overrightarrow{\omega} \times J_r \begin{bmatrix}0 \\ 0 \\ \omega_1 - \omega_2 + \omega_3 - \omega_4\end{bmatrix} = \begin{bmatrix}-qJ_r(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ pJ_r(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ 0\end{bmatrix}$
  $\overrightarrow{\omega} = \begin{bmatrix}p \\ q \\ r\end{bmatrix}$: angular velocities. ω_i: spinning speeds of motors M_i. J_r: rotor inertia.

Thus, the total torque τ is expressed as follows:

$$\tau = \begin{bmatrix}\tau_1 \\ \tau_2 \\ \tau_3\end{bmatrix} = \tau_a + \tau_G = \begin{bmatrix}l(f_4 - f_2) - qJ_r(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ l(f_3 - f_1) + pJ_r(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ k_\tau(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)\end{bmatrix} \tag{1.9}$$

System Inputs

Thus, there exists two types of forces and torques that are acting on the quadrotor platform: the translation force F_ξ which is called the thrust u, and the torque τ which represents all the torques about the distinct axes $(\tau_\phi, \tau_\theta, \tau_\psi)$.
Rather than controling the velocities of the motors, the control in torque and thrust is synthesized, from which motor velocities are effortlessly inferred. Thus, the quadrotor has four control inputs, which are the inputs of its motors:

$$\begin{aligned} u_1 &= u \text{ (thrust)} \\ u_2 &= \tau_\phi \\ u_3 &= \tau_\theta \\ u_4 &= \tau_\psi \end{aligned}$$

The various torques and forces which are taken into account in this study are illustrated in figure 1.2.

1.3 Dynamical equations

The dynamics equations of motion of a mechanism can be expressed by using either the Euler-Lagrange formalism or the Newton-Euler formalism.

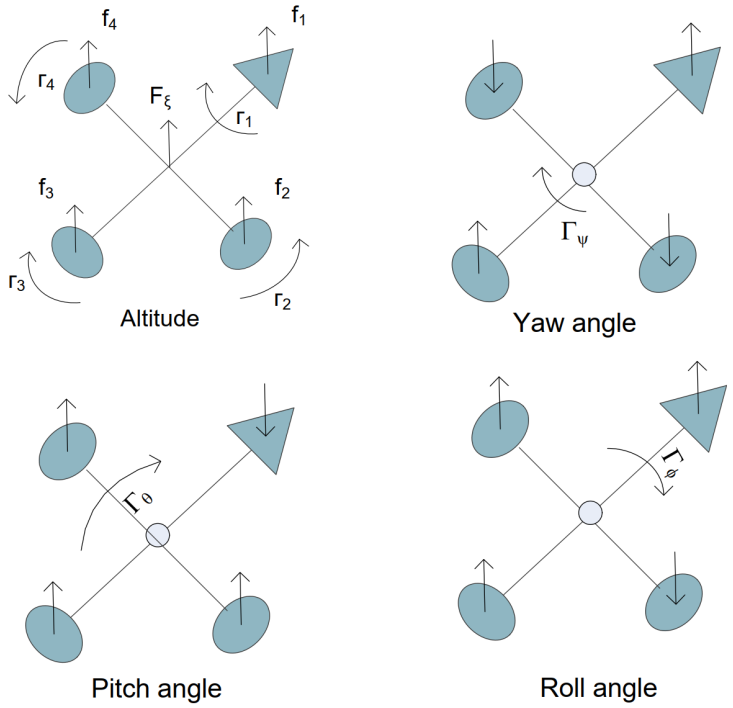


Figure 1.2: Model of the quadrotor represented with motor torques and Euler angles [1].

The quadrotor's model [2], when the Euler-Lagrange formalism is used [17], is founded on the equations of energy of the mechanical system. The Newton-Euler formalism is given in [18]. In both cases, the model is computed based on the following assumptions:

- The quadrotor has a rigid structure.
- The quadrotor has a symmetrical structure.
- The center of gravity (CoG) and the fixed frame at the center of the body are assumed to be coincident.
- The propellers of the quadrotor are assumed to be rigid.
- The thrust and drag forces are assumed to be proportional to the square of the spinning speed of each propeller.

1.3.1 Euler-Lagrange Formalism

The Lagrangian is expressed as follows:

$$L(q,\dot{q}) = T_{Trans} + T_{rot} - U = L_{Trans} + L_{Rot} \tag{1.10}$$

with:

- q: generalized coordinates.
- U: Potential energy ($U = mgU_z$)
- $L_{Rot} = T_{Rot}$

By utilizing the Euler-Lagrange equations expressed in (1.11) below, the equations of motion can be inferred:

$$\frac{d}{dt}\left(\frac{\partial L(q_i,\dot{q}_i)}{\partial \dot{q}_i}\right) - \frac{\partial L(q_i,\dot{q}_i)}{\partial q_i} = \tau_i = \begin{bmatrix} F_\xi \\ \tau \end{bmatrix} \tag{1.11}$$

with F_ξ depicting the translation forces expressed in the mobile frame and τ depicting the torques. Thus, the Euler-Lagrange equations can be decomposed into two parts, the part dealing with the translation, and the second part dealing with the rotation.
First, the focus will be on the part related to the translation.

Translation Model

The translational equations on the fixed frame are expressed as follows:

$$\frac{d}{dt}\left(\frac{\partial L_{Trans}}{\partial \dot{\xi}}\right) - \frac{\partial L_{Trans}}{\partial \xi} = F_\xi \tag{1.12}$$

The Lagrangian of translation in the frame R_O is expressed as follows:

$$L_{Tran} = T_{Trans} - U = \frac{1}{2} m \dot{\xi}^\intercal \dot{\xi} - mgz \tag{1.13}$$

With $T_{Trans} = \frac{1}{2} m \dot{\xi}^\intercal \dot{\xi}$ representing the kinetic energy which is expressed in the fixed frame such that $\dot{\xi} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^\intercal$ is expressed the fixed frame R_O.
F_ξ must be expressed in R_O for the purpose of obtaining the equations of translation in the fixed frame, because F_ξ is typically measured in the mobile frame R_U.
First, the derivative of L_{Trans} must be calculated with respect to $\dot{\xi}$

$$\frac{\partial L_{Trans}}{\dot{\xi}} = \frac{\partial (\frac{1}{2} m \dot{\xi}^\intercal \dot{\xi})}{\partial \dot{\xi}} = m\dot{\xi} \tag{1.14}$$

After that, equation (1.14) can be differentiated with respect to time as follows:

$$\frac{d}{dt}\left(\frac{\partial L_{Trans}}{\dot{\xi}}\right) = \frac{d}{dt}(m\dot{\xi}) = m\ddot{\xi} \tag{1.15}$$

Then, L_{Trans} is now differentiated with respect to ξ as follows:

$$\frac{\partial L_{Trans}}{\partial \xi} = mg\overrightarrow{U_z} \tag{1.16}$$

Thus, the final model of the translation in the mobile frame becomes:

$$\frac{d}{dt}\left(\frac{\partial L_{Trans}}{\partial \dot{\xi}}\right) - \frac{\partial L_{Trans}}{\partial \xi} = F_\xi \Rightarrow m\ddot{\xi} + mg\overrightarrow{U_z} = F_\xi \tag{1.17}$$

Finally, all that is left is to transform the forces of translation and express them in the fixed frame in order to obtain the equations of translation in the fixed frame:

$$\begin{aligned} F_\xi = {}^{O}\begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + {}^{U}\begin{bmatrix} 0 \\ 0 \\ u \end{bmatrix} \Rightarrow {}^{O}F_\xi = {}^{O}\begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + {}^{O}\begin{bmatrix} 0 \\ 0 \\ u \end{bmatrix} \\ \Rightarrow {}^{O}F_\xi = {}^{O}\begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + {}^{O}A_U \; {}^{O}\begin{bmatrix} 0 \\ 0 \\ u \end{bmatrix} = {}^{O}\begin{bmatrix} (\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi)u \\ (-\sin\phi\cos\psi + \sin\psi\sin\theta\cos\phi)u \\ \cos\theta\cos\phi u - mg \end{bmatrix} \end{aligned} \tag{1.18}$$

Thus, the translational model of the quadrotor can finally be expressed as follows:

$$\begin{cases} m\ddot{x} = (\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi)u \\ m\ddot{y} = (-\sin\phi\cos\psi + \sin\psi\sin\theta\cos\phi)u \\ m\ddot{z} = \cos\theta\cos\phi u - mg \end{cases} \tag{1.19}$$

The translation model in (1.19) can be simplified if small angles of order 2 for pitch and roll are used:

$$\begin{cases} m\ddot{x} = (\phi\sin\psi + \theta\cos\psi)u \\ m\ddot{y} = (-\phi\cos\psi + \theta\sin\psi)u \\ m\ddot{z} = u - mg \end{cases} \tag{1.20}$$

Thus, it can be observed that in this case, the vertical dynamics is fully decoupled from the rest of the system.
Now, as for the physical interpretation of this case:

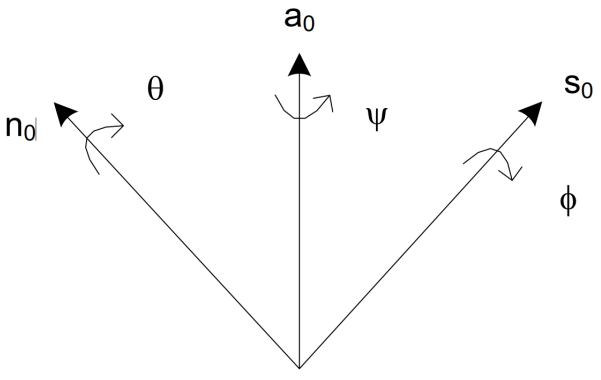


Figure 1.3: Angles of rotation [1].

If small angles are considered for the following types:

- Case 1: θ small, $\phi = 0$

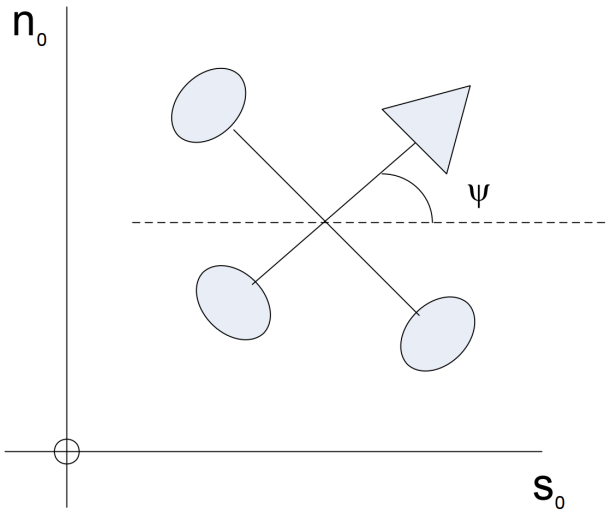


Figure 1.4: Top view of the quadrotor [1].

$$\begin{cases} m\ddot{x} = \theta \cos\psi u \\ m\ddot{y} = \theta \sin\psi u \\ m\ddot{z} = u - mg \end{cases} \tag{1.21}$$

As can be observed from equation (1.21), the pitch angle θ is not zero. This means that the quadrotor is tilted forward. Thus, equation(1.21) shows that the more the UAV is tilted, the more it accelerates. And, if for example $\psi = \frac{\pi}{2}$, it means that $m\ddot{x} = 0$ and $m\ddot{y} = \theta u$, so the quadrotor will progress forward along the axis of n_0 (figure 1.4). And if $\psi = 0$, it means that $m\ddot{x} = \theta u$ and $m\ddot{y} = 0$, so the quadrotor will progress forward along the axis of s_o (figure 1.4).

- Case 2: $\theta = 0$, ϕ is small

$$\begin{cases} m\ddot{x} = \phi \sin\psi u \\ m\ddot{y} = -\phi \cos\psi u \\ m\ddot{z} = u - mg \end{cases} \tag{1.22}$$

In this case, the quadrotor is tiled on the left or right side. If, for instance, $\psi = 0$:

  - $m\ddot{x} = 0$ which is natural because in this case, the quadrotor can only roll.
  - $m\ddot{y} = -\phi u$. So, if the quadrotor has to move forward along the n_0 axis (figure 1.4), the ϕ must be negative.

Now, moving on to the rotational model.

Rotational Model

Contrarily to translational model, the rotational model in the fixed frame is tedious to compute. It is usually computed in the mobile frame for computational simplicity. Indeed, L_{rot} includes the expression of the inertia, which is constant with respect to the mobile frame. Whereas in

the fixed frame, the expression of inertia will not be constant anymore, and when L_{rot} is differentiated with respect to time, the complexity of the calculation will increase [19]. Furthermore, gyroscopic effects are not neglected.

Angular Velocities

A relationship exists between the body-fixed angular velocity vector $\begin{bmatrix} p & q & r \end{bmatrix}^\intercal$, and the rate of change of the Euler angles $\begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^\intercal$. This relationship can be found by resolving the Euler rates into the body-fixed coordinate frame.
Based on the theorem of decomposition of rotations, which is expressed as follows:

$$\overrightarrow{\Omega} = \Omega_x \overrightarrow{x} + \Omega_y \overrightarrow{y} + \Omega_z \overrightarrow{z} +$$

In the equation above, the three vectors express the rotation vectors and the vector $\overrightarrow{\Omega}$ is the vector of rotation in general. Moreover, the angular velocity of the body-fixed frame usually called $\eta = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$
The expression of η can be expressed in terms of the variations of Euler angles:

$$^U\eta = \dot{\psi}\overrightarrow{a_0} + \dot{\theta}\overrightarrow{n_1} + \dot{\psi}\overrightarrow{s_2}$$

where $\overrightarrow{n_1}$ represents the intermediate axis of rotation which is observed in figure 1.1.

$$^U\overrightarrow{\eta} = Rot(-\phi)Rot(-\theta)\begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + Rot(-\phi)\begin{bmatrix} 0 \\ \dot{\psi} \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix}$$

$$^U\overrightarrow{\eta} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix}\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{1.23}$$

Then, the transition matrix R_r which expresses the decomposition of the angular velocity vector $\overrightarrow{\eta}$ with respect to $\begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^\intercal$ in the mobile frame [19] and [16] is defined as follows:

$$R_r = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \tag{1.24}$$

So, the angular velocity vector expressed in the mobile frame is:

$$^U\overrightarrow{\eta} = R_r \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{1.25}$$

Thus, by expanding equation (1.23), the following is obtained:

$$\begin{cases} p = \dot{\phi} - \dot{\psi}\sin\theta \\ q = \dot{\theta}\cos\phi + \dot{\psi}\cos\theta\sin\phi \\ r = -\dot{\theta}\sin\phi + \dot{\psi}\cos\phi\cos\theta \end{cases} \tag{1.26}$$

1.3.2 Newton-Euler Formalism

Translational Model

Based on the Newton's second law:

$$\sum \overrightarrow{F} = m\overrightarrow{a} \tag{1.27}$$

with

- $\overrightarrow{F} = \overrightarrow{F}_{\xi} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix}$: vector of external forces.
- $\overrightarrow{a} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}$: accelerations vector.
- m: mass of the quadrotor.

So, by applying Newton's second law on the quadrotor problem at hand:

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = {}^{O}\begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + {}^{U}\begin{bmatrix} 0 \\ 0 \\ u \end{bmatrix} = {}^{O}\begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + k_f \, {}^{U}\begin{bmatrix} 0 \\ 0 \\ \omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2 \end{bmatrix}$$

If u is then expressed in the fixed frame, the equations of the translational model in (1.19) that were found using the Euler-Lagrange formalism, are found again here:

$$\begin{cases} m\ddot{x} = (\sin\psi \sin\phi + \cos\psi \sin\theta \cos\phi)u \\ m\ddot{y} = (-\sin\phi \cos\psi + \sin\psi \sin\theta \cos\phi)u \\ m\ddot{z} = \cos\theta \cos\phi u - mg \end{cases} \tag{1.28}$$

Rotational Model

Based on the Newton-Euler equations, for a rotating frame [18]:

$$I\dot{\omega} + \omega \times I\omega = \tau \tag{1.29}$$

with

- $I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$
- $\eta = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$
- $\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix}$

Thus, equation (1.29) becomes:

$$\begin{bmatrix} I_{xx}\dot{p} \\ I_{yy}\dot{q} \\ I_{zz}\dot{r} \end{bmatrix} + \begin{bmatrix} qr(I_{zz} - I_{yy}) \\ pr(I_{xx} - I_{zz}) \\ pq(I_{yy} - I_{xx}) \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} \tag{1.30}$$

If equation 1.9 is used, then input vector τ can be substituted to acquire the final model of the rotation in the body frame:

$$\begin{cases} \dot{p} = qr\frac{(I_{yy}-I_{zz})}{I_{xx}} + \frac{l}{I_{xx}}(f_4 - f_2) - \frac{1}{I_{xx}}qJ_r(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ \dot{q} = pr\frac{(I_{zz}-I_{xx})}{I_{yy}} + \frac{l}{I_{yy}}(f_3 - f_1) + \frac{1}{I_{yy}}pJ_r(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ \dot{r} = pq\frac{(I_{xx}-I_{yy})}{I_{zz}} + \frac{1}{I_{zz}}k_\tau(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{cases} \tag{1.31}$$

If the nonlinear model is then simplified by using very small angles, then $\begin{bmatrix} p \\ q \\ r \end{bmatrix} \approx \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$ will be obtained from equation (1.26), and the following model of rotation is obtained:

$$\begin{cases} \ddot{\phi} = \dot{\theta}\dot{\psi}\frac{(I_{yy}-I_{zz})}{I_{xx}} + \frac{l}{I_{xx}}(f_4 - f_2) - \frac{1}{I_{xx}}\dot{\theta}J_r(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ \ddot{\theta} = \dot{\psi}\dot{\phi}\frac{(I_{zz}-I_{xx})}{I_{yy}} + \frac{l}{I_{yy}}(f_3 - f_1) + \frac{1}{I_{yy}}\dot{\phi}J_r(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ \ddot{\psi} = \dot{\theta}\dot{\phi}\frac{(I_{xx}-I_{yy})}{I_{zz}} + \frac{1}{I_{zz}}k_\tau(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{cases} \tag{1.32}$$

It should be noted that equation (1.32) is only valid for very small angles.

1.4 State-Space Model

The translational and rotational models (1.28) and (1.32) respectively that were developed in subsection 1.3.2 express the differential equations of the system. However, for the purpose of control design, it is desirable to reduce the complexity and simplify the model to satisfy the real-time limitations of the embedded control loop. Thus, the thrust and the drag coefficients are assumed to be constant. As a result, the system can be expressed in state-space form $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ with $\mathbf{x}$ the state vector and $\mathbf{u}$ the control input vector.
The state vector has the following form:

$$\mathbf{x} = \begin{bmatrix} \phi & \dot{\phi} & \theta & \dot{\theta} & \psi & \dot{\psi} & z & \dot{z} & x & \dot{x} & y & \dot{y} \end{bmatrix}^\intercal \tag{1.33}$$

With,

$$\begin{aligned} x_1 &= \phi & \qquad x_7 &= z \\ x_2 &= \dot{x}_1 = \dot{\phi} & x_8 &= \dot{x}_7 = \dot{z} \\ x_3 &= \theta & x_9 &= x \\ x_4 &= \dot{x}_3 = \dot{\theta} & x_{10} &= \dot{x}_9 = \dot{x} \\ x_5 &= \psi & x_{11} &= y \\ x_6 &= \dot{x}_5 = \dot{\psi} & x_{12} &= \dot{x}_{11} = \dot{y} \end{aligned} \tag{1.34}$$

Moreover, the control input vector has the following form:

$$\mathbf{u} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^\intercal \tag{1.35}$$

Where the control inputs are mapped by:

$$\begin{cases} u_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ u_2 = b(-\Omega_2^2 + \Omega_4^2) \\ u_3 = b(\Omega_1^2 - \Omega^3) \\ u_4 = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{cases} \tag{1.36}$$

As stated earlier when defining equation (1.32), the transformation matrix between the rate of change of the attitude angles $(\dot{\phi}, \dot{\theta}, \dot{\psi})$ and the angular velocities of the body (p, q, r) can be regarded as the identity matrix if the disturbances due to hover flight are small. As a result, the following can be written:

$$(\dot{\phi}, \dot{\theta}, \dot{\psi}) \approx (p, q, r) \tag{1.37}$$

Simulation tests have demonstrated that this assumption is reasonable [2].

From equations (1.28), (1.32),(1.33) and (1.35), the following expression is obtained after simplification:

$$f(\mathbf{x},\mathbf{u}) = \begin{pmatrix} \dot{\phi} \\ \dot{\theta}\dot{\psi}a_1 + \dot{\theta}a_2\Omega_r + b_1u_2 \\ \dot{\theta} \\ \dot{\phi}\dot{\psi}a_3 - \dot{\phi}a_4\Omega_r + b_2u_3 \\ \dot{\psi} \\ \dot{\theta}\dot{\psi}a_5 + b_3u_4 \\ \dot{z} \\ g - (\cos\phi\cos\theta)\frac{1}{m}u_1 \\ \dot{x} \\ u_x\frac{1}{m}u_1 \\ \dot{y} \\ u_y\frac{1}{m}u_1 \end{pmatrix} \tag{1.38}$$

With,

$$\begin{array}{l|l} a_1 = (I_{yy} - I_{zz})/I_{xx} & b_1 = l/I_{xx} \\ a_2 = J_r/I_{xx} & b_2 = l/I_{yy} \\ a_3 = (I_{zz} - I_{xx})/I_{yy} & b_3 = l/I_{zz} \\ a_4 = J_r/I_{yy} & \\ a_5 = (I_{xx} - I_{yy})/I_{zz} & \end{array} \tag{1.39}$$

$$\begin{aligned} u_x &= (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi) \\ u_y &= (\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi) \end{aligned} \tag{1.40}$$

It is important to note that the angles and the derivatives of the angles do not depend on the components of the translation in the system represented by equation (1.38). Contrarily, the translation components depend on the angles. So, the system represented by equation(1.38) can be depicted as two subsystems, the angle subsystem and the translation subsystem as shown in figure

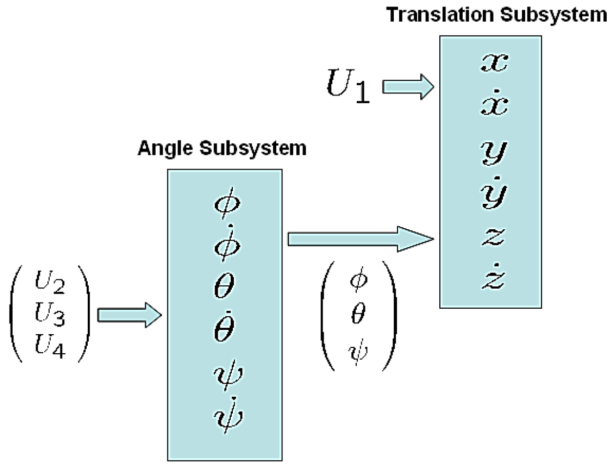


Figure 1.5: Link between the rotation and the translation subsystems.[2]

Chapter 2

Control of quadrotors

In the following sections of this chapter, differential flatness, the general control architecture of a quadrotor and different potential control approaches (linear and nonlinear) that can be used to control a quadrotor are explained.

2.1 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:

$$\mathbf{y} = \mathbf{y}(\mathbf{x}, \mathbf{u}, \dot{\mathbf{u}}, ..., \mathbf{u}^{(p)}) \tag{2.1}$$

With,

$$\begin{cases} \mathbf{x} = \mathbf{x}(\mathbf{y}, \dot{\mathbf{y}}, ..., \mathbf{y}^{(q)}) \\ \\ \mathbf{u} = \mathbf{u}(\mathbf{y}, \dot{\mathbf{y}}, ..., \mathbf{y}^{(r)}) \end{cases} \tag{2.2}$$

Thus, 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. 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 [20], and [21] 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:

$$\mathbf{y} = \begin{bmatrix} x & y & z & \psi \end{bmatrix}^\intercal \tag{2.3}$$

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 $\mathbf{x}$.

2.2 General Control Architecture

In the last few years, many researches have developed interest in control of quadrotors. As a result, various control approaches have been proposed. The most known control architecture [21] consists of three nested control loops, as shown in figure 2.1, in order to generate the suitable thrust in each actuator to follow the desired signal. This strategy assumes that the attitude dynamics of a quadrotor are much faster than the translational dynamics. Assuming that Euler angles are used to define the attitude and that a navigation module generates the desired trajectory $(\boldsymbol{r}_d(t), \psi_d(t))$ as shown in section 2.1, then:

Position controller has the objective of driving the errors occurring on the translational dynamics to zero. And, the outputs of this outer loop are the thrust $f = u_1$, which is sent to the motor controller, and the desired attitude $(\theta_d(t), \phi_d(t))$, which corresponds to the reference signal of the attitude controller.

Attitude controller has the goal of driving the errors occuring on the rotational dynamics to zero. This controller generates the inputs $\boldsymbol{\tau} = \begin{bmatrix} u_2 & u_3 & u_4 \end{bmatrix}^\intercal$ that are then sent to the motor controller.

Motor controller This controller receives the control inputs $\boldsymbol{\tau} = \begin{bmatrix} f & \boldsymbol{\tau} \end{bmatrix}^\intercal$ and maps them into the desired spinning velocities Ω_i for each individual rotor based on equation (1.36). Moreover, low-level control laws are designed and realized in the firmware of the drone to make the convergence from the actual rotations to these desired values.

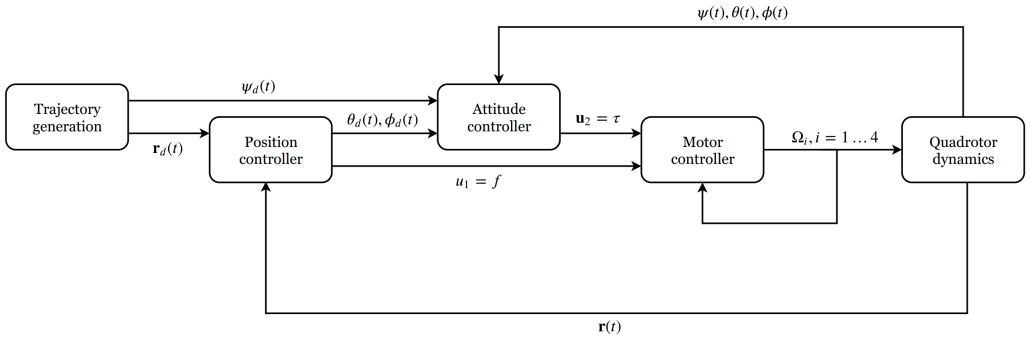


Figure 2.1: General control architecture of a quadrotor.

2.3 General Control Approaches

2.3.1 Method of Linearization

By using extreme assumptions, it is feasible to apply linear control techniques in order to control a quadrotor ([22], [23]). 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:

$$\overline{\mathbf{u}} = \begin{bmatrix} f \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} mg \\ \mathbf{0_{3\times 1}} \end{bmatrix} \tag{2.4}$$

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

$$\dot{\mathbf{x}} = \overline{\mathbf{f}}(\overline{\mathbf{x}}, \overline{\mathbf{u}}) \tag{2.5}$$

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

$$\dot{\mathbf{x}} = \left[\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\right]_{(\bar{\mathbf{x}},\bar{\mathbf{u}})} \mathbf{x} + \left[\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\right]_{(\bar{\mathbf{x}},\bar{\mathbf{u}})} \mathbf{u} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{2.6}$$

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 [22]. 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 [24, 25] control or simple PD or PID controller [26, 27].

2.3.2 Internal Lyapunov Stability

Before defining the *Lyapunov Direct Method*, the notions of stability will be thoroughly defined first.

2.3.2.1 Notions of Stability

For a general system without any control input

$$\dot{x}(t) = f(x(t), 0, t) \qquad (CT) \tag{2.7}$$

$$\dot{x}(k+1) = f(x(k), 0, k) \qquad (DT), \tag{2.8}$$

it is said that a point $\overline{x}$ is called an *equilibrium point* from time t_0 for the continuous system (CT) if $f(\overline{x}, 0, t) = 0$, $\forall t \geq t_0$. Moreover, in the discrete time (DT) case, the point $\overline{x}$ is an equilibrium point from time k_0 if $f(\overline{x}, 0, k) = 0$, $\forall k \geq k_0$. If the system begins from state $\overline{x}$ at time t_0 or k_0, then the system will stay there and will not change with time. It is possible for nonlinear system to have more than one equilibrium point (equilibria). There also exists another class of special solutions in the case of nonlinear systems, these solutions are called *periodic* solution. However, it is outside the scope of this paper and interested readers are referred to [28] for a more in depth explanation. So, the focus will be on equilibria. It is desired to identify the *stability* of the equilibria in some way. For instance, it is desired to know if, given some small perturbation to the system, the state would either come back to the equilibrium point, remain close to it in some sense, or it diverges.

The most useful notion of stability for an equilibrium point of a nonlinear system is provided by the definition below. Assuming that the equilibrium point is at the origin, because if $\overline{x} \neq 0$, a simple translation can be done to obtain a system that is equivalent with the equilibrium at the origin.

Asymptotic stability A system is said to be *asymptotically stable* about its own equilibrium point at the origin if the following two conditions are satisfied [3]:

1. For any $\epsilon > 0$, $\exists \delta_1 > 0$ such that if $\|x(t_0)\| < \delta_1$, then $\|x(t)\| < \epsilon, \forall t > t_0$.
2. $\exists \delta_2$ such that if $\|x(t_0)\| < \delta_2$, then $x(t) \to 0$ at $t \to \infty$.

For the first condition, it is required that the state trajectory should be restricted to a randomly small *"ball"* that is centered at the equilibrium point and has a radius ϵ, when released from an *arbitrary* initial condition in a ball that has an adequately small (yet positive) radius δ_1. This is referred to as *stability in the sense of Lyapunov* (i.s.L.). It is also possible to have stability in the sense of Lyapunov without having asymptotic stability, in that case, it is said that the equilibrium point is *marginally stable*. Moreover, there also exist nonlinear systems that satisfy the second condition without being stable in the sense of Lyapunov. Furthermore, an equilibrium point that is *not* stable in the sense of Lyapunov is said to be *unstable*.

2.3.2.2 Lyapunov's Direct Method

General Idea If the following continuous-time system is considered

$$\dot{x}(t) = f(x(t)) \tag{2.9}$$

which has an equilibrium point at the origin (x=0).This system is called a time-invariant system because f does not depend explicitly on the time t. In such a system, the stability analysis of the equilibrium point is in general a tedious task. This is because there is no way to write a simple formula which relates the trajectory to the initial state. The main idea of *Lyapunov's direct method* is to establish properties of the equilibrium point (of the nonlinear system in general) by evaluating how a certain carefully chosen scalar function of the changes as the state of the system changes. (The term *"direct"* is used to distinguish this method from the Lypunov's *"indirect"* method, which tries to establish properties of the equilibrium point by assessing the behavior of the *linearized* system at that point [29].
As an example, a continuous and scalar function $V(x)$ that is equal to 0 at the origin and positive elsewhere is considered in some ball that is enclosing the origin. So, $V(0) = 0$ and $V(x) > 0$ when $x \neq 0$ in this ball. This $V(x)$ can be considered as an *"energy"* function. Also, let $\dot{V}(x)$ denote the time derivative of $V(x)$ throughout any trajectory of the system. In other words, $\dot{V}(x)$ varies as $x(t)$ varies proportionally to equation (2.9). If this derivative is negative in all the region (with the exclusion of the origin), then this means that the energy is decreasing with time in a strict manner. Moreover, since the lower bound of the energy is 0, then the energy must go to 0 with time. This means that all trajectories will converge to the origin (zero state). This idea is formalized below.

Lyapunov Functions Assuming V is a continuous map from $\mathbb{R}^n$ to $\mathbb{R}$, $V(x)$ is called a *locally positive definite* (lpd) function about x=0 if

1. $V(0) = 0$.
2. $V(x) > 0$, $0 < \|x\| < r$ for a given r.

Also, the function is called locally *positive semi-definite* (lpsd) if the strict inequality applied on the function in the second condition is changed to $V(x) \geq 0$. The function $V(x)$ is locally *negative definite* (lnd) if $-V(x)$ is lpd. Moreover, $V(x)$ is locally *negative semidefinite* (lnsd) if $-V(x)$ is lpsd. In order to graphically illustrate the locally positive definite function $V(x)$, it is useful to imagine having "contours" of constant V that form (at least in a tiny region about the origin) a set of nested surfaces that are encircling the origin. An example is given in figure 2.2 below.

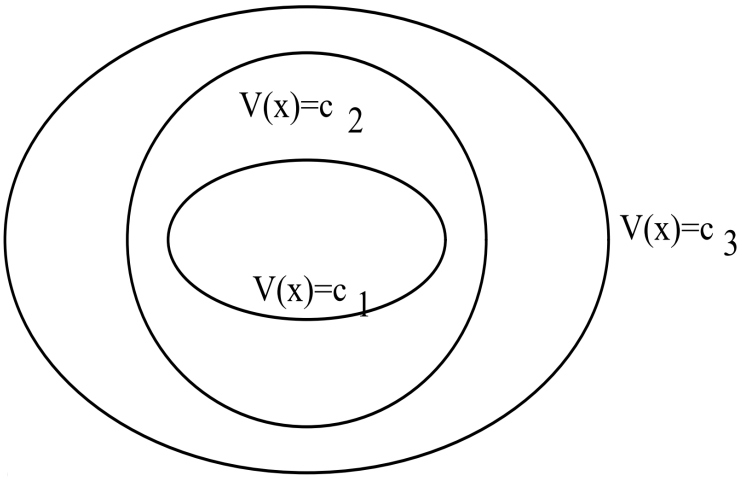


Figure 2.2: Different values for a Lyapunov function, where $c_1 < c_2 < c_3$. [3]

In the continuous time case, the focus in this bibliography will be restricted to $V(x)$ that has first partial derivatives that are continuous. (In the case of discrete time, only continuity will suffice, so differentiability is not required in that case.) The derivative of V with respect to time *along a trajectory of the system* 2.9 is denoted as $\dot{V}(x(t))$. This derivative is expressed as follows:

$$\dot{V}(x(t)) = \frac{dV(x)}{dx}\dot{x} = \frac{dV(x)}{dx}f(x) \tag{2.10}$$

where $\frac{dV(x)}{x}$ is the *Jacobian* of V with respect to x which contain the partial derivative V with respect to every component of x: $\frac{\partial V}{\partial x_i}$.
Moreover, assuming V is a lpd function (a "candidate Lyapunov function"), and $(\dot{V})$ is the derivative along trajectories of the system of equation 2.9. Then,V is called a *Lyapunov function* of the system of equation 2.9 if $\dot{V}$ is lnsd [3].

Lyapunov Theorem for Local Stability If a Lyapunov function for the system of equation (2.9) exists, this means that $x = 0$ is a stable equilibrium point in the sense of Lyapunov. Moreover, if $\dot{V} < 0$, $0 < \|x\| < r_1$, for some r_1, in other words, if $\dot{V}$ is lnd, this means that $x = 0$ is an equilibrium point that is asymptotically stable[3].

Lyapunov Theorem of Global Asymptotic Stability The region in the state space for which the earlier results hold is determined by the region over which $V(x)$ represents a Lyapunov function. It is interesting to determine the *"basin of attraction"* of an equilibrium point that is asymptotically stable. In other words, the basin of attraction is the set of initial conditions whose following trajectories conclude at the equilibrium point. Thus, an equilibrium point is *"globally asymptotically stable"* (also called asymptotically stable "in the large") if its basin of attraction is the entire state space.

If a function $V(x)$ has the following criteria below,

1. $V(x)$ is positive definite throughout the state-space.
2. $V(x)$ also has the added property that $|V(x)| \to \infty$ as $\|x\| \to \infty$.
3. The derivative of V (in other words $\dot{V}$) is negative definite throughout the state-space.

Then, the equilibrium point at the origin is globally asymptotically stable [3].

Discrete-Time Systems Fundamentally identical results hold for the system

$$x(k+1) = f(x(k)) \tag{2.11}$$

provided that $\dot{V}$ is represented as

$$\dot{V}(x) \triangleq V(f(x)) - V(x),$$

in other words as

$$V(\text{next state}) - V(\text{present state})$$

2.3.3 Model Predictive Control

2.3.3.1 General Idea

There exist "open-loop" methods [30] 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 approach is also known as *"receding horizon control"* [4] since the *"prediction horizon"* (finite horizon) shifts 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 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 $\mathbf{x}(k+1)$.

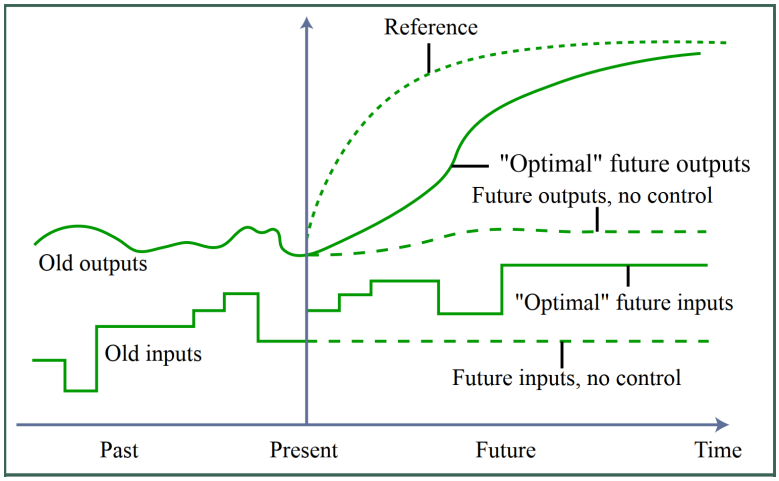


Figure 2.3: Basic idea of MPC [4].

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

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, because it 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 2.1 below shows the different companies that have used MPC in different industries in 2014.

Table 2.1: MPC applications in different industries in 2014 [13].

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).

2.3.3.2 Design Parameters

The different parameters that can be tuned in a MPC controller ([31, 32, 33]) 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 determine the rate at which the controller executes the control algorithm.

- If T_s is too large, then when a disturbance occurs, the controller will be unable to react to the disturbance quick enough.
- If T_s is too small, the controller will have much faster reaction times to disturbances and setpoint 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 close-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 paper) 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 2.3 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. However, it should be noted that hard constraints cannot be violated. In addition, applying hard constraints on both the inputs and outputs at the same time may cause conflict 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 and the rate of change of the inputs .

Weights Model Predictive Control could have many goals. One goal could be to have the outputs converge to their set-points as fast as possible. Another goal can 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 relative to each other. It is also possible to adjust relative weights within the input rates and the outputs.

2.3.3.3 Basic formulation

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

$$\begin{cases} \mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k) \\ \mathbf{z}(k) = C\mathbf{x}(k) \end{cases} \tag{2.12}$$

and a cost function as follows:

$$J = \sum_{j=0}^{N} \{\|\mathbf{z}(k+j|k)\|_{R_{zz}} + \|\mathbf{u}(k+j|k)\|_{R_{uu}}\} + F\mathbf{x}(k+N|k)) \tag{2.13}$$

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.

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 [34]. 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.
Moreover, solving for a very long sequence of control input 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 the optimized control sequence will be implemented. This is why MPC is designed by using a small N.

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

$$\begin{aligned} \min_{u} \quad & J = \sum_{j=0}^{N} \{\|\mathbf{z}(k+j|k)\|_{R_{zz}} + \|\mathbf{u}(k+j|k)\|_{R_{uu}}\} \\ \text{s.t.} \quad & \mathbf{x}(k+j+1|k) = A\mathbf{x}(k+j|k) + B\mathbf{u}(k+j|k), \\ & \mathbf{x}(k|k) \equiv \mathbf{x}(k), \\ & \mathbf{z}(k) = C\mathbf{x}(k+j|k), \\ & |\mathbf{u}(k+j|k)| \leq u_m \end{aligned} \tag{2.14}$$

The problem statement in (4.10) can be converted into a more standard optimization problem as follows:

$$\begin{aligned}
\mathbf{z}(k|k) &= C\mathbf{x}(k|k) \\
\\
\mathbf{z}(k+1|k) &= C\mathbf{x}(k+1|k) = C(A\mathbf{x}(k|k) + B\mathbf{u}(k|k)) \\
&= CA\mathbf{x}(k|k) + CB\mathbf{u}(k|k) \\
\\
\mathbf{z}(k+2|k) &= C\mathbf{x}(k+2|k) \\
&= C(A\mathbf{x}(k+1|k) + B\mathbf{u}(k+1|k)) \\
&= CA(A\mathbf{x}(k|k) + B\mathbf{u}(k|k)) + CB\mathbf{u}(k+1|k) \\
&= CA^2\mathbf{x}(k|k) + CAB\mathbf{u}(k|k) + CB\mathbf{u}(k+1|k) \\
&\vdots \\
\mathbf{z}(k+N|k) &= CA^N\mathbf{x}(k|k) + CA^{N-1}B\mathbf{u}(k|k) + \dots \\
&\quad + CB\mathbf{u}(k+(N-1)|k)
\end{aligned}$$

The equations above can then be grouped as follows:

$$\begin{bmatrix} \mathbf{z}(k|k) \\ \mathbf{z}(k+1|k) \\ \mathbf{z}(k+2|k) \\ \vdots \\ \mathbf{z}(k+N|k) \end{bmatrix} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^N \end{bmatrix} \mathbf{x}(k|k)$$
$$+ \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ CB & 0 & 0 & & 0 \\ CAB & CB & 0 & & 0 \\ \vdots & & & & \\ CA^{N-1}B & CA^{N-2}B & CA^{N-3}B & \dots & CB \end{bmatrix} \begin{bmatrix} \mathbf{u}(k|k) \\ \mathbf{u}(k+1|k) \\ \vdots \\ \mathbf{u}(k+N-1|k) \end{bmatrix}$$

Now $Z(k)$ and $U(k)$ can be defined as follows:

$$Z(k) \equiv \begin{bmatrix} \mathbf{z}(k|k) \\ \vdots \\ \mathbf{z}(k+N|k) \end{bmatrix} \quad , \quad U(k) \equiv \begin{bmatrix} \mathbf{u}(k|k) \\ \vdots \\ \mathbf{u}(k+N-1|k) \end{bmatrix}$$

It should be noted that:

$$\sum_{j=0}^{N} \mathbf{z}(k+j|k)^\intercal R_{zz} \mathbf{z}(k+j|k) = Z(k)^\intercal W_1 Z(k)$$

with W_1 denoting the weighting matrix.

Thus, the elements of the cost function in (4.10) can now be expressed as follows:

$$\begin{aligned} Z(k)^\intercal W_1 Z(k) + U(k)^\intercal W_2 U(k) &= (G\mathbf{x}(k) + HU(k))^\intercal W_1 (G\mathbf{x}(k) + HU(k)) + U(k)^\intercal W_2 U(k) \\ &= \mathbf{x}(k)^\intercal H_1 \mathbf{x}(k) + H_2^\intercal U(k) + \frac{1}{2} U(k)^\intercal H_3 U(k) \end{aligned} \tag{2.15}$$

With

$$H_1 = G^\intercal W_1 G, \quad H_2 = 2(\mathbf{x}(k)^\intercal G^\intercal W_1 H), \quad H_3 = 2(H^\intercal W_1 H + W_2)$$

Since the term $\mathbf{x}(k)^\intercal H_1 \mathbf{x}(k)$ in (2.15) does not contain the component to be minimized $U(k)$, this means that it will be constant throughout the optimization process. As a result, it can be omitted. Finally, the MPC problem can be re-written as:

$$\begin{aligned} \min_{\mathrm{U}(k)} \quad & \tilde{J} = H_2^\intercal U(k) + \frac{1}{2} U(k) H_3 U(k) \\ \text{s.t.} \quad & \begin{bmatrix} I_N \\ -I_N \end{bmatrix} U(k) \leq u_m \end{aligned} \tag{2.16}$$

Thus, the MPC problem has been transformed to the form of a quadratic program for which there exists various efficient tools to solve the problem.

2.3.3.4 MPC Observations

The current form of (2.16) assumes that the full state of the system is available. Moreover, a state estimator can also be used. In addition, the corresponding control is also assumed to be sensed and applied immediately on the system, this is usually a safe and reasonable assumption for most control systems. However, given that the optimization problem will be solved at each time step, then there is a possibility that accounting for this computation delay could be mandatory.

Case of no imposed constraints If there are no active constraints on (2.16), then the solution of the QP becomes

$$U(k) = -H_3^{-1} H_2 \tag{2.17}$$

which can be expressed as follows:

$$u(k|k) = -\begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix} (H^\intercal W_1 H + W_2)^{-1} H^\intercal W_1 G \mathbf{x}(k) \tag{2.18}$$

$$= -K\mathbf{x}(k) \tag{2.19}$$

which is nothing else but a state feedback controller.

Stability of MPC The stability of MPC greatly depends on the terminal cost and terminal constraints [35]. On the other hand, a classic result [36] states that if a Model Predictive Control algorithm was applied on a linear system with constraints, and assuming there exists terminal constraints:

- $\mathbf{x}(k+N|k) = 0$ for the predicted state $\mathbf{x}$.
- $\mathbf{u}(k+N|k) = 0$ for the computed future control input $\mathbf{u}$

And, if the optimization problem is realizable at time step k, then $\mathbf{x} = 0$ is stable.

2.3.3.5 Different MPC Methods

The MPC methods that are mainly used are the following:

- Linear time-invariant MPC.
- Adaptive MPC.
- Gain-Scheduled MPC.
- Non-linear MPC.

The method to be used is chosen based on the complexity of the system at hand and on the required goals to be reached.

In case of linear systems The method to be used is called linear time-invariant MPC. The constraints must be linear and the cost function must be quadratic. These characteristics will result in a convex optimization problem [37] which has a global optimum. An example of a convex function is shown in figure 2.4 below.

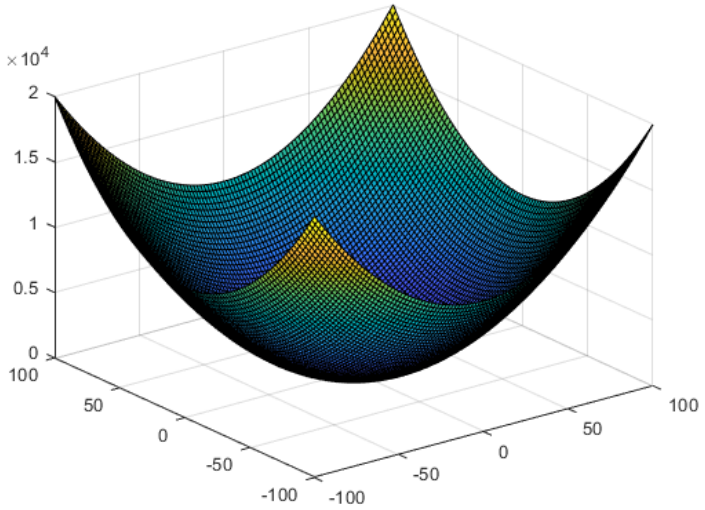


Figure 2.4: Representation of a convex function plotted using MATLAB.

In case of a nonlinear systems If the system is linearizable, then both Adaptive MPC and Gain-Scheduled MPC can be used in this case. But the constraints are still linear and the cost function is still quadratic. In this case, the nonlinear function can be linearized around an operating point, which will result in a linear function that approximates the nonlinear system well near the operating point. However, it should be noted that the linear function will not work well outside the operating region. This is why it is interesting to find multiple linearized models, with each model representing the nonlinear function well around it's operating point.

Adaptive MPC In this case, a linear model is computed on-line as the operating conditions change. And, the internal plant model used by the MPC is updated with the corresponding linear model at each time step. Moreover, it should be noted that the optimization problem remains the same across different operating points. In other words, the number of elements in the state and the constraints remain the same across different operating conditions [38].

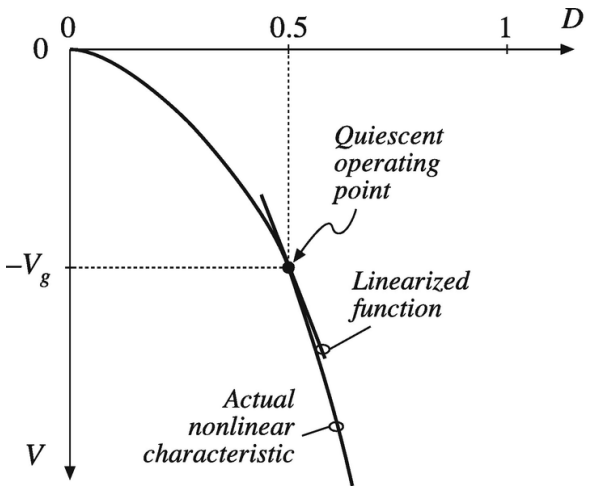


Figure 2.5: Example of the linearization of a nonlinear function around an operating point [5].

Gain-Scheduled MPC In this case, linearization of the nonlinear model is done offline at the operating points of interest. Then, a linear MPC controller is designed for each operating point. However, it should be noted that unlike Adaptive MPC, each controller is now independent from the other. In other words, the number of elements in the state and the constraints are different across different operating conditions. It should also be noted that for this method, an algorithm must be designed to switch between the predefined MPC controllers for different operating conditions. Moreover, Gain-Scheduled MPC also uses more memory than Adaptive MPC [39].

Non-linear MPC If the system is nonlinear and cannot be linearized, and both the constraints and the cost function are also nonlinear, then Nonlinear MPC can be used in that case. It is the most powerful method of all the different methods mentioned earlier, since it uses the most accurate representation of the plant. Thus, predictions are more accurate. However, Nonlinear MPC is the most difficult method to solve in real-time. Because, in that case the problem becomes a non-convex optimization problem. Thus, the cost function may have many local minima, and finding the global minimum may be hard in that case. An example of a non-convex function is shown in figure 2.6 below.

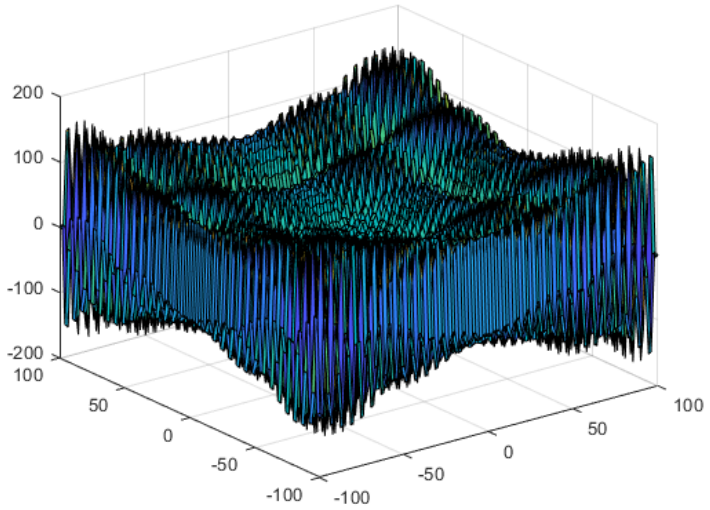


Figure 2.6: Representation of a non-convex function plotted using MATLAB.

2.3.3.6 Strategies to Improve Computational Time

As demonstrated in (2.16), an MPC problem is formulated as a QP problem that tries to minimize a quadratic cost function in general. Moreover, MPC computations become more complex as the number of state elements, the number of constraints and the MPC parameters increase. Moreover, as stated in section 2.3.3.1, if the MPC controller is running on applications with slow dynamics, then computational complexity is not a concern. However, if MPC is running on applications with fast dynamics, then the computational complexity becomes crucial. In addition, it is important to note that the matrices that are stored in the processor of the system for MPC computations grows with the increasing number of optimization variables. Thus, this could cause a memory problem. So, to reduce the complexity and computational time of MPC, the following methods can be used:

Order reduction techniques They are used to discard states that do not contribute to the dynamics of the system [40]. And, using order reduction techniques will also reduce the memory usage of the controller.

Explicit MPC Instead of solving the optimization problem online for the current state, *Explicit* MPC solves it offline for each value of the state $\mathbf{x}$ within a given range [41]. Thus, for each state value within a given range, the Explicit MPC precomputes the optimal solution. And, this solution consists of linear functions that are piecewise affine and continuous is $\mathbf{x}$. An example of Explicit MPC applied on a one-dimensional system is shown in figure 2.7 below.

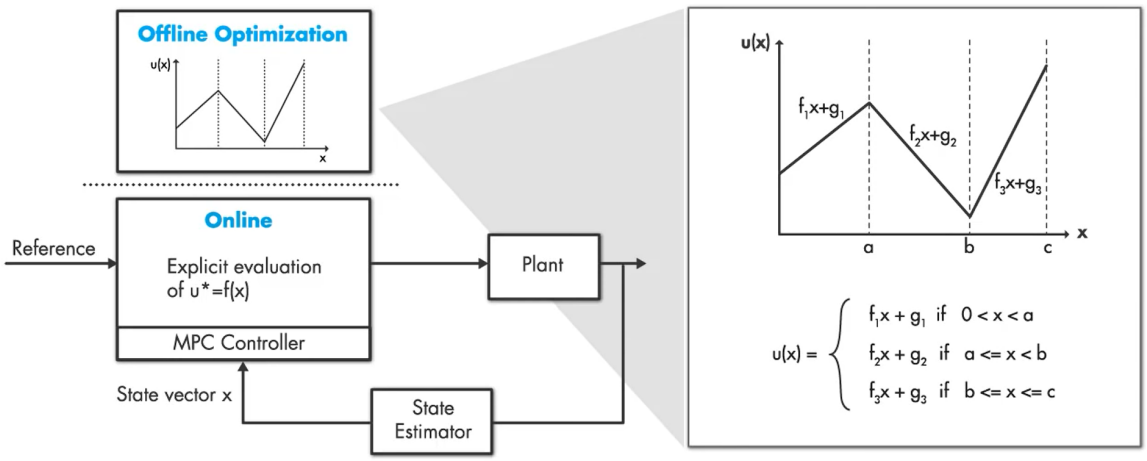


Figure 2.7: Example of an Explicit MPC controller applied on a one-dimensional system [6].

As can be observed from figure 2.7, the constraints cut the solution space into different regions. And, each region maps into a unique solution. Another example of an Explicit MPC applied on a two-dimensional system is shown in figure 2.8 below.

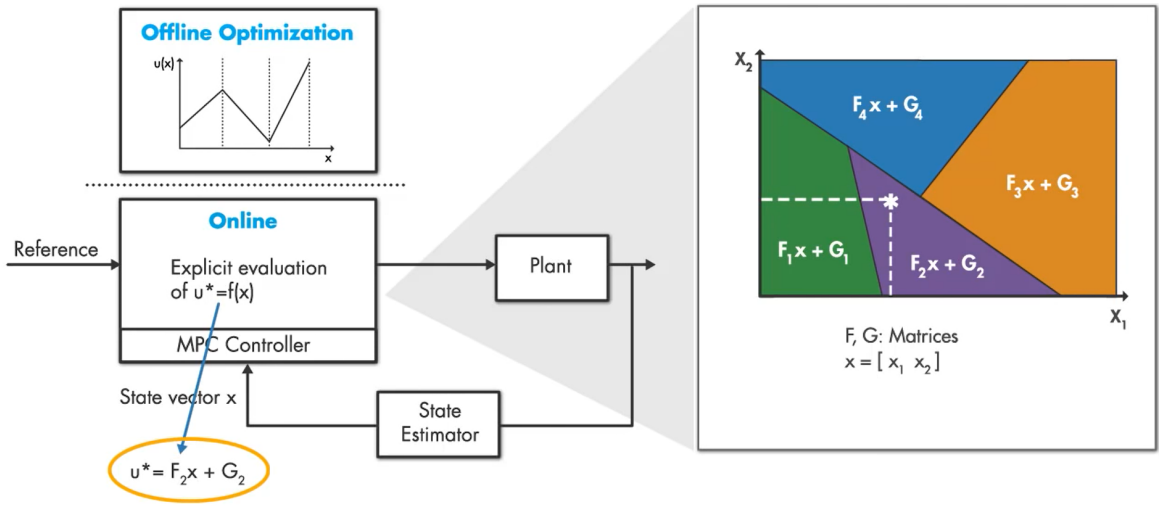


Figure 2.8: Example of an Explicit MPC controller applied on a two-dimensional system [6].

As can be observed from figure 2.8, the Explicit MPC finds the region that the current state lies in and evaluates the linear function that creates the current control input.
Thus, it can be concluded that if the iterative optimization process is reduced to linear function evaluations, then this will greatly simplify the online computations. However, if there is a large number of regions that the state can lie in, then searching for the current state region could sometimes be time consuming. Also, having many regions could cause memory problems for the processor. Thus, the number of regions can be reduced by merging some regions together. However, thee computed solution in that case is not optimal anymore [42].

Suboptimal Solution It is another method to improve the computational time of the MPC controller. In a MPC problem, the optimal solution is very unpredictable and can drastically change between each time step. In addition, the computational time may exceed the sample time T_s. So, it is mandatory to make sure that a solution can be found within T_s and that there still exists additional time for other tasks that need to be executed [43]. A simple example is provided in figure 2.9 below where a maximum value for the iterations is determined, and it is taken as 5 for illustration purposes.

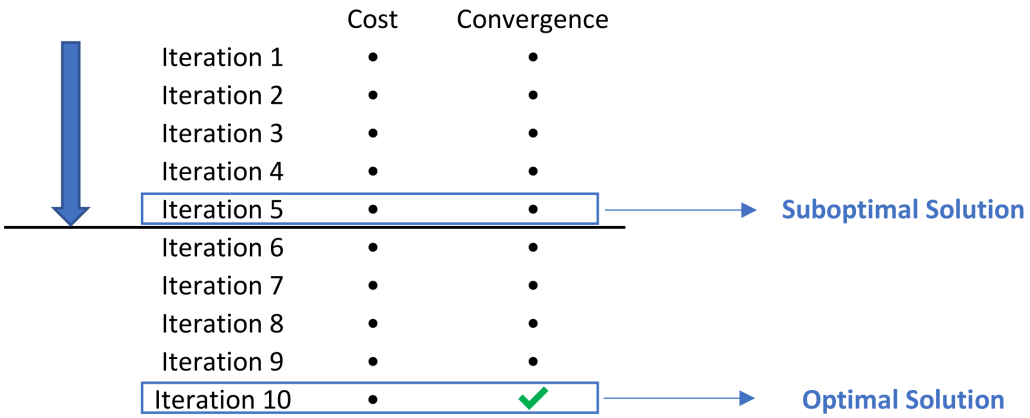


Figure 2.9: Example showing how the suboptimal solution is found when the maximum number of iterations is set to 5.

As can be seen from figure 2.9, the optimal solution is reached in 10 iterations. However, the controller will stop at iteration 5 and take the suboptimal solution. Moreover, it is important to note that the suboptimal solution still satisfies all the constraints of the optimization problem.

The only question that remain is how to determine the maximum number of iterations. This can be done by testing the algorithm on the hardware directly and identifying the execution time used by each iteration. Then, the maximum number of iterations is chosen such that the total execution time does not exceed the sample time of the controller.

2.3.3.7 Existing MPC toolboxes

Some of the available tools that allow to solve the MPC problem are listed below.

- **Model Predictive Control Toolbox**[1]: it is made by MathWorks (closed-source).
- **MPCtools**[2]: it is a free and open-source toolbox for MATLAB and Simultink that permits to create and simulate basic MPC controllers by using linear state space models.
- **do-mpc**[3]: it is a free and comprehensive open-source toolbox for robust model predictive control which is written in Python language.
- **Control Toolbox**[4]: it is an efficient library for control, estimation, optimization and motion planning in robotics, which is written in C++ language.

2.3.4 Sliding mode control

2.3.4.1 Main Principles

Sliding mode control (SMC) is a control technique that is nonlinear presenting exceptional attributes of robustness, accuracy, easy tuning and execution.
The aim of SMS 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 includes 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 [44].
There exists 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 [44, 45, 46]. Researchers have also studied the problems appearing in the practical execution of this class of techniques. [47]
Moreover, the book [48] presents a very modern overview of the most promising current line of theoretical and practical research in the domain.

[1] `https://www.mathworks.com/products/model-predictive-control.html`, accessed on 01/10/2021.
[2] `http://www.control.lth.se/research/tools-and-software/mpctools/`, accessed on 01/10/2021.
[3] `https://www.do-mpc.com/en/latest/`, accessed on 01/10/2021.
[4] `https://github.com/ethz-adrl/control-toolbox`, accessed on 01/10/2021.

2.3.4.2 Simple Description

Considereing the SISO nonlinear system:

$$\dot{x} = f(x,t) + g(x,t)u \tag{2.20}$$
$$y = h(x,t) \tag{2.21}$$

Where y and u represent the scalar output and input variable, and $x \in \mathbb{R}^n$ represents the state vector.
The goal of the control is to make the output variable y follow a chosen profile y_{DES}. This means that it is needed that the output error variable $e = y - y_{DES}$ tend to a small proximity of zero following a transient of reasonable duration.
As stated earlier, the synthesis of SMC requires two phases:

Phase 1: *"Sliding Surface Design".*

Phase 2: *"Control Input Design"*

In the first phase, a specific scalar function σ of the system state is defined such that:

$$\sigma \mathrm{x} : \mathbb{R}^n \to \mathbb{R}.$$

In many cases, the sliding surface relies on the tracking error e_y, along with a specific number of its derivatives

$$\sigma = \sigma(e, \dot{e}, \ldots, e^{(k)}) \tag{2.22}$$

The function σ must be chosen in a way that when $\sigma = 0$, it will result in a stable differential equation such that any $e_y(t) \to 0$ as $t \to \infty$.
The most common choices for the sliding manifold are the following:

$$\sigma = \dot{e} + c_0 e \tag{2.23}$$
$$\sigma = \ddot{e} + c_1 \dot{e} + c_0 e \tag{2.24}$$
$$\sigma = e^{(k)} + \sum_{i=0}^{k-1} c_i e^{(i)} \tag{2.25}$$

The number of derivatives that should be included (k in (2.25)) must be $k = r - 1$, where r is the input-output relative degree of (2.20)-(2.21).
With correctly chosen c_i coefficients, if σ is driven to 0, then the error and its derivatives will decrease to 0 exponentially.
Provided that such property holds, the goal of the control system is to drive σ to 0.
From a geometrical perspective, the equation $\sigma = 0$ represents a surface in the error space that is called *"sliding surface"*. The trajectories of the system that is being controlled are forced to be on the sliding surface, along which the behavior of the system satisfies the design requirements.
A common form for the sliding surface depends on a scalar paramete p, and is expressed as follows:

$$\sigma = \left(\frac{d}{dt} + p\right)^k e \tag{2.26}$$

$$k = 1 \quad \sigma = \dot{e} + pe \tag{2.27}$$
$$k = 2 \quad \sigma = \ddot{e} + 2p\dot{e} + p^2 e \tag{2.28}$$

The parameter p can be chosen randomly, and it defines the particular pole of the derived *"reduced dynamics"* of the system when sliding.
The integer parameter k is on the other hand somewhat crucial, it is required be equal to $r-1$, with r being the relative degree between y and u. This signifies that the relative degree of the σ variable is one.
The next phase (**Phase 2**) is about determining a control action that guides the trajectories of the system onto the sliding manifol. This means that the control is capable of driving the σ variable to zero in finite time.
There exist many approaches that are based on the approach of sliding mode control:

- *"Standard"* (also called *"first-order"*) sliding mode control.
- *"High-order"* sliding mode control.

Emphasis is dedicated to the second order sliding mode approach, and some references to the higher order approaches are also granted. Frequent characteristic of all sliding mode-based techniques is that no specific information about the original system dynamics is required. In other words, the controlled system will be treated as an entirely uncertain "black box" object.

2.3.4.3 First-Order Sliding Mode Control

Along the manifold $\sigma = 0$, the control is discontinuous. So, it can be expressed as follows:

$$u = -Usgn(\sigma) \tag{2.29}$$

This means that:

$$u = \begin{cases} -U & \sigma > 0 \\ U & \sigma < 0 \end{cases} \tag{2.30}$$

With U having a constant, positive and sufficiently large value.

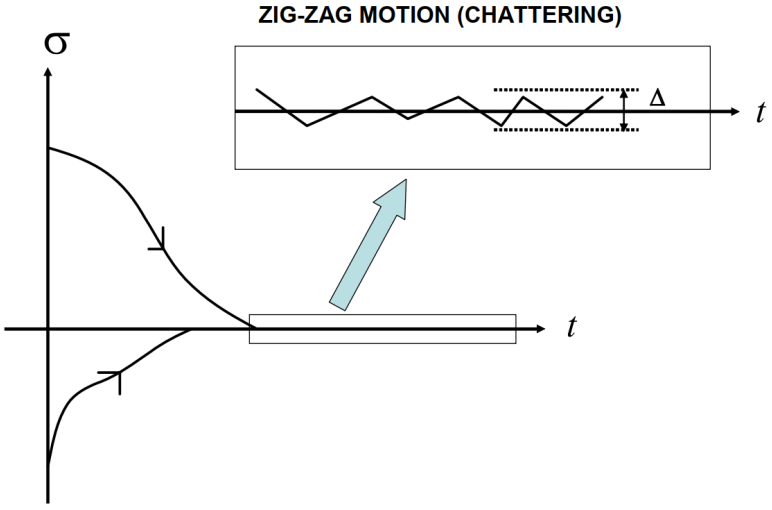


Figure 2.10: Typical evolution of σ starting from different initial conditions. [7]

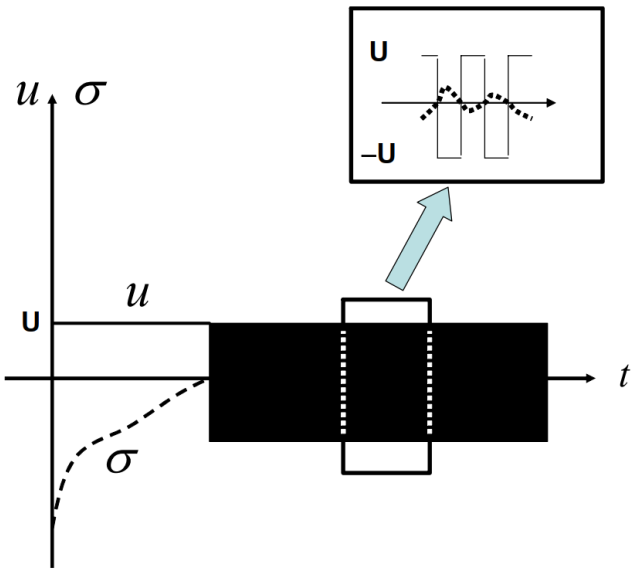


Figure 2.11: Typical evolution of the control signal u (σ is represented by dashed lines). [7]

In steady condition the control variable u will alternate at very high (theoretically infinite) frequency along the values u=U and u=-U, which can be observed in figure 2.11.
The discontinuous switching control with high frequency in figure 2.11 is suitable in electrical implementations (where PWM control signals are usually used) but it results in an increase in oscillations and numerous different problems in several areas, such as the control of mechanical systems.
In order to find a solution for the problem presented above (referred to as *"chattering phenomenon"*) roughly (smoothed) execution of sliding mode control techniques have been proposed where the discontinuous "sign" term is substituted with a continuous and smooth approximation using one of the two functions below.

$$\begin{array}{lllll} \text{SAT} & u = -Usat(\sigma;\epsilon) \equiv -U\dfrac{\sigma}{|\sigma|+\sigma} & \epsilon > 0 & \epsilon \approx 0 \\ \text{TANH} & u = -Utanh(\sigma/\epsilon) & \epsilon > 0 & \epsilon \approx 0 \end{array} \tag{2.31}$$

Unfortunately, this approach is successful only when uncertainties are not applied on the system and the control action that prevents these uncertainties can be set to zero in the sliding mode.

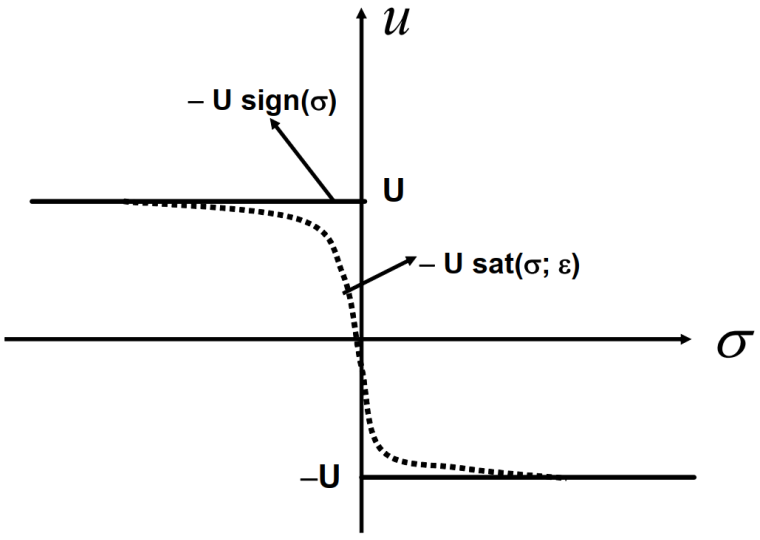


Figure 2.12: Smooth approximations of sliding mode control. [7]

2.3.4.4 Second order sliding mode control

By using the above-described smooth approximations, some problems will be reduced. However, this comes at the expense of a loss of robustness.
Second order sliding mode control algorithms are effective approaches that entirely resolve the chattering problem without compromising the properties of robustness.
There has been many studies on second order sliding mode control. Two of the most interesting papers are [49, 50]. An overview of the mentioned papers is shown below.

SuperTwisting Algorithm It is of the most used (2-SMC) algorithms. This algorithm is expressed with the following equations:

$$\begin{aligned} u &= \lambda\sqrt{|\sigma|}sign(\sigma) + w \\ \dot{w} &= -W sign(\sigma) \end{aligned} \tag{2.32}$$

An appropriate way for tuning its parameters is by using the following relations:

$$\lambda = \sqrt{U} \qquad\qquad W = 1.1U \tag{2.33}$$

Where U is a positive constant to be taken sufficiently large.
Practically, one has to gradually increase U until good performance is attained in the closed loop system. This type of single-parameter *"trial and error"* tuning is specifically suitable in practical applications.
The super-twisting algorithm can be considered to be a nonlinear version of a classical PI controller. This correlation becomes clear by referring to figure 2.13 below.

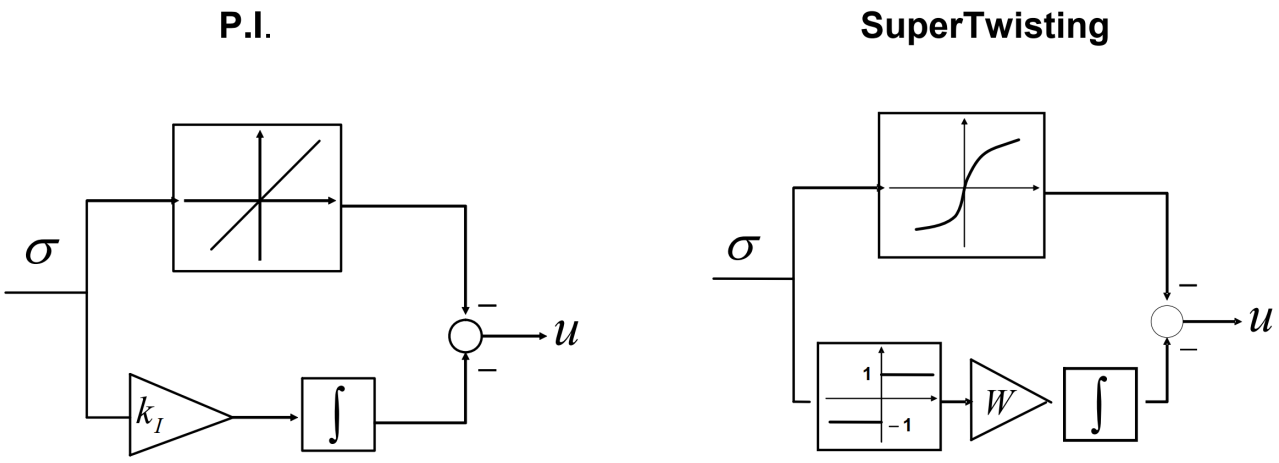


Figure 2.13: Block diagrams of a PI (left) and Super-Twisting (right) controllers.[7]

Second order SMC resolves the chattering problem because the control law is now a continuous function of time. In the sight of dynamics that are not modeled, some remaining chattering exist. But there are some design methods to second-order modes that can limit such an aspect which is not desired.

2.3.5 Other Types of Nonlinear Control Methods

Although the approaches that have been mentioned so far are potentially useful for this master thesis, there is still some other approaches that deserve to be mentioned because of their popularity and applications:

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 [51]. However, this method is not robust, but it is computationally fast. In order to handle disturbances, Fang et al. [52] 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. [53] 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 [54] or pick-and-place applications with small loads.

Moreover, other methods for the control of quadrotors could be found in the next chapter, where the state of the art in the performance of flipping maneuvers with quadrotors is provided.

Chapter 3

Multi-flips maneuver with quadrotors

The aim of this chapter is to explain the physics of the multi-flip maneuver, in addition to the major difficulties that relate to the implementation of flipping maneuvers with quadrotors. Furthermore, a review of the existing limitations that the current approaches have are shown and the existing scientific literature is presented. The content of this chapter were thoroughly covered by the previous master students Marco Orsingher and Antonio Marino for which their papers were used as references for this chapter. In fact, Orsingher [55] managed to perform the flipping maneuver in simulations on MATLAB and ROS by using polynomial trajectory generation techniques, however real experiments were unsuccessful. Marino [56] tackled the same problem the very next year by using trajectory optimization techniques. After successful simulations in ROS, Marino was able to perform a single flip in real experimentation. However, due to the pandemic, no further experiments were possible.

3.1 Physics of a quadrotor flip

In order to perform the tedious process of multi-flip maneuvers with a quadrotor, high angular velocity and large angle attitude control are needed. More conventionally, the multi-flip trajectory can be expressed as a rotation of $2n\pi$ about a flipping axis $\mathbf{a}$, with n the number of desired flips to be carried out by the quadrotor. Physically, the maneuver can be divided into 4 different phases, which are shown in figure 3.1.

Climb phase The quadrotor must accelerate in a vertical direction as much as possible. This will ensure that the required height to perform the multi-flip maneuver is achieved. In fact, while performing the multi-flip maneuver, the thrust forces cannot balance out the weight of the drone. This will cause the drone to start falling. The duration of this phase is based on the number of flips that must be performed by the quadrotor. This number will be chosen by the user.

Multi-flip phase This stage begins the moment the vehicle starts rotating and continues until the flipping angle reaches the chosen value of $2n\pi$. In this phase, the quadrotor hits the maximum altitude because of the required high angular velocity and starts falling because of the gravity effect. Generally, all the focus is on following the desired closed-loop attitude, and the vertical position is not controlled. the time spent in this phase depends on the desired number of flips n and on the maximum angular velocity achievable by the quadrotor.

Descent phase In order for the quadrotor not to crash into the ground, the main idea is to set the maximum thrust reference to the actuation system. This will balance out the gravity force. The higher the number of required flips, the faster the quadrotor will fall due to gravity effects. Thus, this phase ends when the descent is fully balanced out. The user

may also decide what the duration of this phase is if the physics of the problem allow a slower descent.

Re-stabilization phase In this phase, the altitude of the quadrotor is adjusted to a desired value. Moreover, this phase ends when a new control signal is provided by the user.

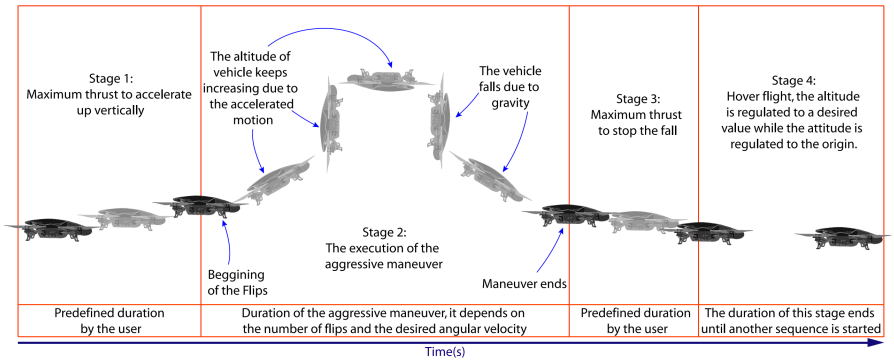


Figure 3.1: Representation of the different phases needed for performing multi-flip maneuvers[8].

3.2 Control Approaches for Multi-Flip Maneuvers

The research community has tackled the multi-flip problem numerous times on quadrotors and different methods have been suggested to achieve up to triple flips both indoor and outdoor. The existing research can be stored into three different methods: hybrid systems theory, open-loop iterative learning and closed-loop attitude control. This section gives an inclusive overview and suitable references on the state of the art in flipping maneuvers with quadrotors.

3.2.1 Hybrid Systems Theory

A convenient approach for dealing with complex systems is to divide them into a set of individual modes. Each mode will have its own continuous dynamics equations. Then, they will be analyzed by using hybrid systems theory. Gillula et al. ([9, 57]) suggested using this strategy for quadrotor multi-flips. The main idea is to breakdown the flipping maneuver into a sequence of individual phases and then use an appropriate technique from hybrid systems theory to generate provable transitions between the modes. By doing this procedure, the maneuver will be able to be performed safely. The separate modes that are considered in this work are shown in figure 3.2, which should be right from right to left. When compared to the different phases shown in section 3.1, the climb phase is called the *"impulse"*, the multi-flip phase is the *"drift"* and the descent and the re-stabilization phases are merged together and are called *"recovery"*.

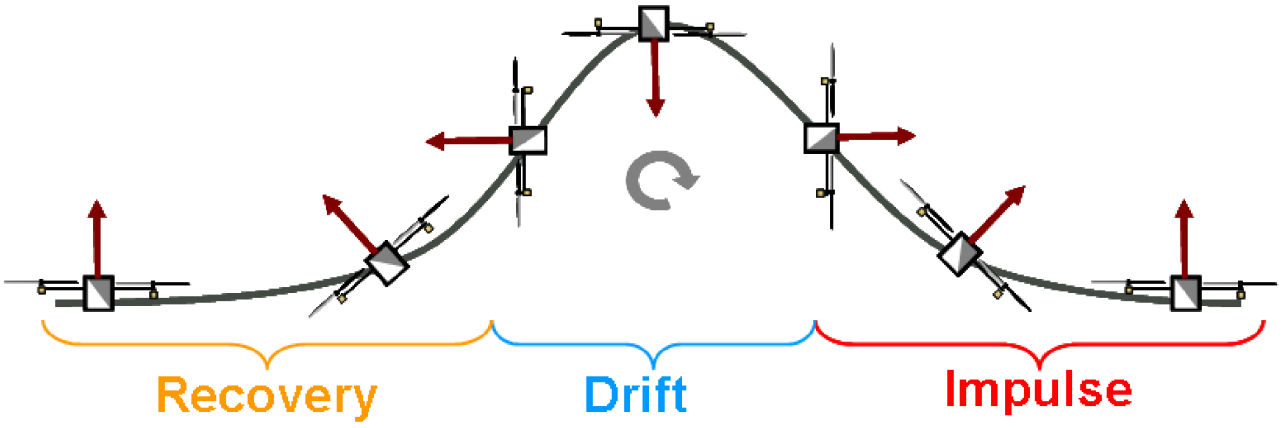


Figure 3.2: Representation of the individual phases during a backlip (from right to left)[9]

The tool used for generating feasible transitions between the individual modes is called the *"theory of reachable sets"*. By considering a system characterized by the following dynamics:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{d}) \tag{3.1}$$

with $\mathbf{x}$ the state of the system, $\mathbf{u} \in U$ the control input of the system and $\mathbf{d} \in D$ a bounded noise applied on the system. In that case, two different types of reachable sets are established:

1. A *capture set* which represents all the states for which, for any possible value of the noise $\mathbf{d}$, the control input $\mathbf{u}$ is still capable of driving the state into a desired state region in a finite time horizon t.

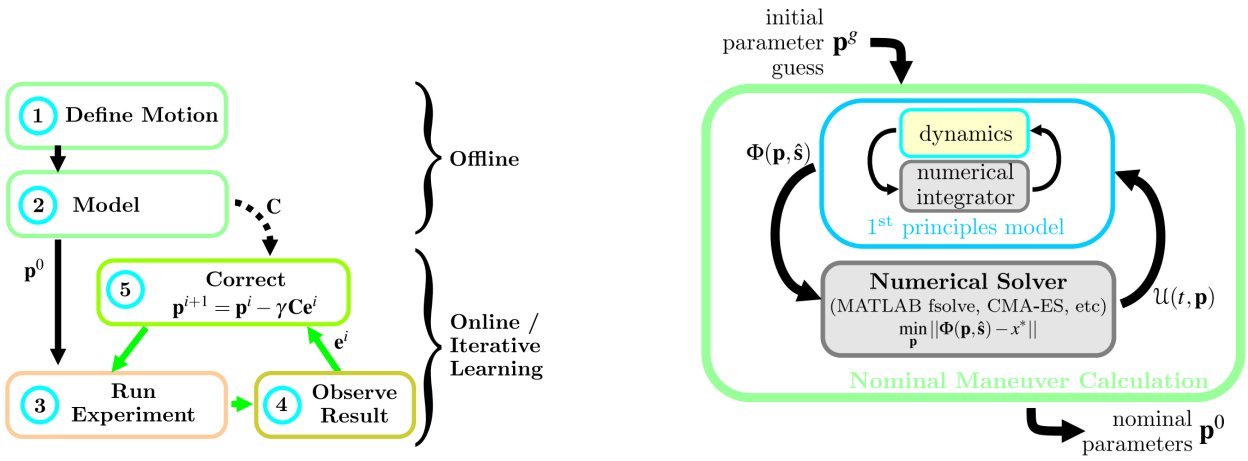
2. An *avoid set* which represents all the states for which, for any possible value of the control input $\mathbf{u}$, the noise $\mathbf{d}$ will drive the system into an undesired set. Thus, the roles of $\mathbf{u}$ and $\mathbf{d}$ are reversed when compared to the capture set.

Generating quadrotor maneuvers using reachable sets is achievable by using a backward approach. The main idea consists of starting from the final desired set, the dynamics of the n-th mode can be run in reverse to build a capture set for that phase. This capture set becomes the target region for the $(n-1)$-th node and the procedure will be repeated between every 2 modes. By applying this procedure, feasible transitions between a chain of individual modes can be built. From a mathematical perspective, the problem is formulated as a differential battle between the control and the noise. In fact, the aim of the control input $\mathbf{u}$ is to keep the state of the system away from the region of undesired sets which the noise $\mathbf{d}$ is attempting to drive the state of system into (*avoid set*) and also to reach a desired set. Whereas the noise attempts to drive the state of the system out of it (*capture set*). This method has been proven to be successful with a 1.1 kg quadrotor which was able to perform a backflip outdoors. However, the main disadvantage of this method is that it necessitates a long procedure of generating such capture sets and avoid sets.

3.2.2 Open-Loop Iterative Learning

One of the main issues that is related to performing aggressive maneuvers with quadrotors is that having an exact physical modeling of the quadrotor during an intense flight procedure is very complicated. In addition, generating reference trajectories for aerobatic flight is not an easy operation. Beginning from these factors, Lupashin et al. ([58, 59, 10]) designed a simple

open-loop iterative learning method to perform multi-flip maneuvers without performing any aerodynamic modeling nor trajectory planning. The workflow of this method is shown in figure 3.3 on the left.



(a) The global workflow of the algorithm

(b) The global workflow of the algorithm

Figure 3.3: The open-loop iterative learning workflow for performing multi-flips with quadrotors[10].

The first step which is needed is to determine the desired motion in a parameterized manner, so that it is possible to optimize vector $\mathbf{p}$ over such parameters. $\mathbf{p}$ contains the time and the accelerations needed during each phase of the flip maneuver. The second procedure which is needed to initialize the iterative learning algorithm is to calculate the nominal motion and to build the correction matrix. Starting from a simplified model of the quadrotor and a coarse initial guess $\mathbf{p}^g$, then a numerical solver is utilized to calculate the nominal parameters $\mathbf{p}^0$ in order to execute the multi-flip maneuver. In fact, assuming that the desired final state $\mathbf{x}^*$ can be attained with the control input $\mathbf{u}(\mathbf{p}, t)$ and by referring to $\mathbf{\Phi}(\mathbf{p}, \hat{\mathbf{s}})$ as the final state of the nominal system after the maneuver is executed.(with the vector $\hat{\mathbf{s}}$ representing physical constants), then the nominal set of parameters $\mathbf{p}^0$ is the solution of the following optimization problem, as shown in figure 3.3:

$$\mathbf{p}^0 = \arg\min \|\mathbf{\Phi}(\mathbf{p}, \hat{\mathbf{s}}) - \mathbf{x}^*\| \tag{3.2}$$

Moreover, by denoting $\mathbf{\Phi}(\mathbf{p}^0, \hat{\mathbf{s}})$ as the final state of the nominal model obtained using the nominal set of parameters calculated earlier, then the correction matrix for the iterative learning algorithm is expressed as follows:

$$\boldsymbol{C} = \left(\frac{\partial \mathbf{\Phi}(\mathbf{p}^0, \hat{\mathbf{s}})}{\partial \mathbf{p}} \right)^{-1} \tag{3.3}$$

In the end, steps 3 to 5 in figure 3.3a will then run online in an iterative manner until a proposed convergence criterion is satisfied. For $i = 0 \dots n$, the motion is carried out on the quadrotor itself while using the current set of parameters $\mathbf{p}^i$ and the final measured error $\mathbf{e}^i$. Subsequently, the parameters are updated according to the following law:

$$\mathbf{p}^{i+1} = \mathbf{p}^i - \gamma \mathbf{C} \mathbf{e}^i \tag{3.4}$$

with γ being a step size which is defined by the user. This method is very simple does not require complex computations. But, it is by definition an open-loop method. Thus, there are no guarantees that the maneuver will be performed successfully. In fact, a triple flip maneuver

has been done during experiments with this method, however the repeatability of this result is a big problem. In addition, "creating a parameterized input trajectory formulation that results in a robust learning strategy remains a difficult and tedious task" [10].

3.2.3 Closed-Loop Attitude Control

The most known strategy for obtaining great results in flipping maneuvers of quadrotors is to use closed-loop control just for the attitude part of the platform, thus the position of the center of mass during the flipping maneuver is ignored. Many research works can be recognized under this context, and all of them follow the same ideology: a proper reference on attitude or angular velocity is planned and then tracked in closed-loop with a nonlinear control law. Each paper is distinguished from the others based on the strategy that is chosen to build the control law: attractive ellipsoid method [8], geometric control [60], energy-based control[61] , direct Lyapunov method [62]. Particularly, Chen et al. ([11], [63] ,[64]) generated interesting series of papers within this framework that should be mentioned because they applied the method on the same physical platform of this thesis (Crazyflie 2.0); moreover, significant theoretical contributions have been provided in these papers to greatly comprehend the issue of quadrotor flipping. Particularly, three elements have been investigated:

1. A new depiction of the flipping angle, called ϕ_g, such that a flip around a random axis $\mathbf{a}$ through the center of mass of the quadrotor can be determined. This has permitted to execute multiple flips about the body-fixed axis $\mathbf{b}_1$, and another general axis, rotated by 45° on the ($\mathbf{b}_1$,$\mathbf{b}_2$) plane.

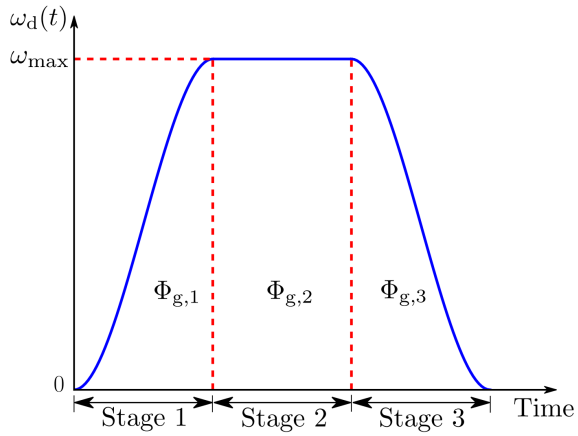
2. A meticulous angular speed planning approach using cubic splines to produce the reference signal $\boldsymbol{\omega}_d$ in such a way that the dynamics of actuation are respected by ensuring that w_{max} is never surpassed. Particularly, the reference is made out of three distinct stages, corresponding to the physical stages presented in figure 3.4a. In fact, throughout the *climb phase* a rising-speed section $\boldsymbol{\omega}_{d,1}$ is needed to reach $\boldsymbol{\omega}_{max}$, then a stage of constant-speed at $\boldsymbol{\omega}_{d,2} = \boldsymbol{\omega}_{max}$ is held while the quadrotor is flipping (*multi-flip phase*), and in the end, the angular velocity $\boldsymbol{\omega}_{d,3}$ is reduced to zero within the *descent* and *re-stabilization phase*. The use of cubic functions ensures that the angular accelerations is kept continuous during all the process. Other works, such as [8], suggested equivalent shape, however linear functions were used instead of splines. Considering that the time derivative of a linear function is just a constant, this resulted in undesirable jumps in the angular reference.
   Considering the desired number of rotations accumulated with each stage $\Phi_{g,i}1$, the time of each stage is changed based on the current flipping angle ϕ_g:

$$\boldsymbol{\omega}_d(t) = \begin{cases} \boldsymbol{\omega}_{d,1} & \text{if} \quad \phi_g \leq \Phi_{g,1} \\ \boldsymbol{\omega}_{d,2} & \text{if} \quad \Phi_{g,1} < \phi_g \leq \Phi_{g,1} + \Phi_{g,2} \\ \boldsymbol{\omega}_{d,3} & \text{if} \quad \Phi_{g,1} + \Phi_{g,2} < \phi_g \leq \Phi_{g,1} + \Phi_{g,2} + \Phi_{g,3} \end{cases} \tag{3.5}$$

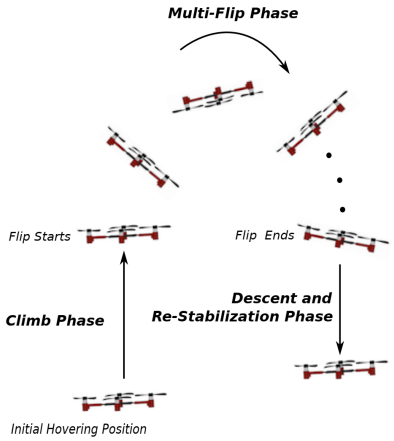
   where:

$$\Phi_{g,1} + \Phi_{g,2} + \Phi_{g,3} = 2n\pi \tag{3.6}$$

   in order to carry-out the flip.



(a) The reference of the angular velocity.



(b) The stages of the multi-flip maneuver.

Figure 3.4: The reference of the flipping speed and the different flipping phases. [11]

3. Three distinct control methods to track the reference of the angular speed, which are then compared in terms of the root mean square error. First, a linear time-invariant (LTI) has been proposed [11] as follows. Provided the error of the angular velocity $\mathbf{e}_\omega = \boldsymbol{\omega} - \omega_d \mathbf{a}$ and the error of the angular acceleration $\mathbf{e}_\alpha = F(s)\boldsymbol{\omega} - \dot{\omega}_d \mathbf{a}$, with F(s) a filter expressed as $F(s) = \lambda s(s+\lambda)^{-1}$, $\lambda > 0$, the consequence control law is:

$$\boldsymbol{\tau} = -\boldsymbol{K}_\omega \mathbf{e}_\omega - \boldsymbol{K}_\alpha \mathbf{e}_\alpha \tag{3.7}$$

In the subsequent works, a backstepping control approach [63] has been examined to solve the latency issue which is the result of the slow response of the actuators, in addition to an adaptive control law to estimate unknown parameters on the fly [64]. In the two cases, these methods achieved better performance when compared to the LTI controller. However, they were not able to successfully reduce the oscillation of the angular velocity within the descent and re-stabilization phase.

Regardless, this series of papers are by far the most successfull and comprehensive work available in literature on performing multi-flip maneuvers with quadrotors.

3.3 Shortcomings of Current Strategies

The control approaches for multi-flip maneuvers that have been presented in section 2.3 demonstrate that the implementation of flipping maneuvers is currently addressed as a more complicated version of the classical attitude control problem. But, no attention has been given on the consequent Cartesian path of the center of mass of the quadrotor, nor on an accurate depiction of the climb phase. In fact, within the first stage, the maximum thrust is established in order to acquire sufficient height. However, none of the works presented above deals with comprehensive altitude analysis in order to give an approximation of the necessary duration for the climb phase. This is mainly done heuristically in experiments. This problem, combined with the fact that the attention is put only on attitude planning and control, results in a totally erratic trajectory for the quadrotor. Thus, it is important to be able to predict the consequent trajectory of the quadrotor during a flipping maneuver. Because, this will help to keep the quadrotor safe, particularly in complex outdoor settings.

Chapter 4

Trajectory optimization

In order to understand what trajectory optimization means, an example of a moving satellite between two planets can taken. The term *trajectory* is used to depict the path that the satellite takes between the two planets. Generally, this path would incorporate both the state (position and velocity) and control (thrust) in function of time. the term *trajectory optimization* alludes to a set of approaches that can be utilized to identify the best trajectory option, generally by choosing the inputs to the system, also known as *controls*, as functions of time.

4.1 The trajectory optimization problem

4.1.1 Formulation of The Optimization Problem

The are numerous ways to formulate a trajectory optimization problem([65], [66] [67]). The main focus will be on single-phase continuous-time trajectory optimization problems. This means that the dynamics of the system are continuous along the full trajectory. Thus, multi-phase methods are outside the scope of this paper and interested readers are referred to [67] for a more general framework.
Generally, an objective function includes two distinct terms: a boundary objective J and a path integral about the full trajectory, with the integrand w . A problem that is containing both terms is considered to be in *Bolza form*. Moreover, a problem only having the integral term is considered to be in *Lagrange form*, and a problem only having the boundary term is considered to be in *Mayer form*.

$$\min_{t_0,t_F,\boldsymbol{x}(t),\boldsymbol{u}(t)} \underbrace{J\big(t_0,t_f,\boldsymbol{x}(t_0),\boldsymbol{x}(t_F)\big)}_{\text{Mayer Term}} + \underbrace{\int_{t_0}^{t_F} w(\tau,\boldsymbol{x}(\tau),\boldsymbol{x}(\tau),\boldsymbol{u}(\tau))d\tau}_{\text{Lagrange Term}} \tag{4.1}$$

In optimization, the term *decision variable* is used to depict the variables that will be adjusted through the optimizer to minimize the objective function. As an example, the decision variables in equation (4.1) are the initial and final time (t_0, t_f) in addition to the trajectories of the state and the control input $(\boldsymbol{x}(t), \boldsymbol{u}(t))$. Numerous limits and constraints are imposed on the optimization problem at hand, which are detailed in the following equations (4.2-4.9).
The first, and possibly the most crucial of all the imposed constraints is the system dynamics, which are generally nonlinear and depict how the system varies with time.

$$\dot{\boldsymbol{x}} = \boldsymbol{f}\big(t,\boldsymbol{x}(t),\boldsymbol{u}(t)\big) \qquad \textbf{system dynamics} \tag{4.2}$$

Then, there is the path constraint, which implements limitations along the trajectory. As an example, a path constraint can be utilized to maintain the foot of a humanoid robot above the ground in the course of it performing a step.

$$\boldsymbol{h}\big(t, \boldsymbol{x}(t), \boldsymbol{u}(t)\big) \leq 0 \qquad \textbf{path constraint} \tag{4.3}$$

Another crucial type of constraint is a nonlinear boundary constraint, which imposes limitations of the initial and final state of the system. As an example, this type of constraint can be utilized to make sure that the step of a humanoid robot is periodic.

$$\boldsymbol{g}\big(t_0, t_F, \boldsymbol{x}(t_0), \boldsymbol{x}(t_F)\big) \leq 0 \qquad \textbf{boundary constraint} \tag{4.4}$$

Typically, there are constant limits on the state and the control. For instant, a 6-degrees of freedom (6-DOF) serial anthropomorphic robot could have limitations on the angles, angular rates, and torques that could be applied throughout the entire trajectory.

$$\boldsymbol{x}_{low} \leq \boldsymbol{x}(t) \leq \boldsymbol{x}_{upp} \qquad \textbf{path bound on state} \tag{4.5}$$
$$\boldsymbol{u}_{low} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{upp} \qquad \textbf{path bound on control} \tag{4.6}$$

Lastly, it is generally crucial to incorporate particular limitations on the initial time t_0, the final time t_F and the state $\boldsymbol{x}(t)$. These types of constraints can be used to make sure that the solution to a trajectory planning problem attains the goal inside some desirable time window, alternatively it attains some target region in state space.

$$t_{low} \leq t_0 < t_F \leq t_{upp} \qquad \textbf{bounds on initial and final time} \tag{4.7}$$
$$\boldsymbol{x}_{0,low} \leq t_0 < \boldsymbol{x}(t_0) \leq \boldsymbol{x}_{0,upp} \qquad \textbf{bound on initial state} \tag{4.8}$$
$$\boldsymbol{x}_{F,low} \leq t_0 < \boldsymbol{x}(t_F) \leq \boldsymbol{x}_{F,upp} \qquad \textbf{bound on final state} \tag{4.9}$$

4.1.2 Direct Collocation Method

The majority of the methods for solving trajectory optimization problem can be categorized as *direct* or *indirect* methods. In this paper, the main focus will be on direct methods, however a brief overview will on indirect collocation methods will be provided. The main idea of a direct method is that it discretizes the trajectory optimization problem and converts it into a nonlinear program. This conversion procedure is named *transcription*. This is why the direct collocation methods are also known as *direct direct transcription methods.* Generally, direct transcription methods are capable of discretizing a continuous trajectory optimization problem by estimating all the continuous functions of the problem as polynomial splines. A *spline* is a function which consists of a series of polynomial segments. Polynomials are utilized since they have two crucial characteristics:

1. They can be described by using a small and finite group of coefficients.
2. It is simple to calculate integrals and derivatives of polynomials in function of these coefficients.

Throughout this paper, two direct collocation methods will be discussed in detail: *trapezoidal* collocation and *Hermite-Simpson* collocation. Moreover, other direct collocation methods will be briefly covered, such as the direct single shooting method, the direct multiple shooting method and the orthogonal collocation method.

4.1.3 Nonlinear Programming

The majority of the direct collocation methods transcribe a continuous-time trajectory optimization problem to a nonlinear problem. A non-linear program is a specific term provided to a constrained parameter optimization problem that contains nonlinear elements in either the objective function or the constraint function. A general expression for a nonlinear program is provided below:

$$\begin{aligned} \min_{\boldsymbol{z}} \quad & J(\boldsymbol{z}) \\ \text{s.t.} \quad & \boldsymbol{f}(\boldsymbol{z}) = \boldsymbol{0}, \\ & \boldsymbol{g}(\boldsymbol{z}) \leq \boldsymbol{0}, \\ & \boldsymbol{z}_{low} \leq \boldsymbol{z} \leq \boldsymbol{z}_{upp} \end{aligned} \tag{4.10}$$

Sometimes, a direct collocation method can result in a linear or quadratic program instead of a non-linear program. This occurs if the constraints (including the dynamics of the system) are linear and the objective function is linear (which results in a linear program) or quadratic (which results in a quadratic program). Linear and quadratic programs are both far easier to tackle than nonlinear programs, rendering them attractive for real-time applications, specifically in robotics.

4.2 Trapezoidal Collocation Method

Trapezoidal collocation operates by transcribing a continuous-trajectory optimization into a non-linear program. This is performed by utilizing trapezoidal quadrature, which is also named the *trapezoidal rule* for integration, to transcribe every continuous element of the problem into a discrete approximation. In this section, the trapezoidal method will be thoroughly examined by demonstrating how the transformation is achieve for each element of a trajectory optimization problem.

4.2.1 Integrals

Frequently, there are integral expressions in trajectory optimization problems. Generally, they are located in the objective function, however, it is also possible that they could be located in the constraints too. The aim is to approximate the continuous integral $\int w(\cdot)dt$ as a sum $\sum c_k w_k$. The main idea is that the summation just necessitates the value of the integrand $w(t_k) = w_k$ at the collocation points t_k about the trajectory. This approximation is performed by applying the trapezoid rule for integration between each pair of consecutive collocation points, which produces the equation below, with $h_k = t_{k+1} - t_k$. [68]

$$\int_{t_0}^{t_F} w\big(\tau, \boldsymbol{x}(\tau), \boldsymbol{u}(\tau)\big) d\tau \approx \sum_{k=0}^{N-1} \frac{1}{2} h_k \cdot (w_k + w_{k+1}) \tag{4.11}$$

4.2.2 System Dynamics

One of the main features of direct collocation method is that it depicts the system dynamics as a set of constraints, referred to as *collocation constraints*. For trapezoidal collocation, the collocation constraints are built by expressing the dynamics of the system in integral form and then approximation of that integral is performed by using trapezoidal quadrature [68].

$$\dot{\boldsymbol{x}} = \boldsymbol{f}$$

$$\int_{t_k}^{t_{k+1}} \dot{\boldsymbol{x}} dt = \int_{t_k}^{t_{k+1}} \boldsymbol{f} dt$$

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}_k \approx \frac{1}{2} h_k \cdot (\boldsymbol{f}_{k+1} + \boldsymbol{f}_k)$$

This approximation is then implemented between each pair of collocation points:

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}_k = \frac{1}{2} h_k \cdot (\boldsymbol{f}_{k+1} + \boldsymbol{f}_k) \qquad k \in \ldots (N-1) \tag{4.12}$$

It should be noted that $\boldsymbol{x}_k$ is a decision variable in the nonlinear program, whereas $\boldsymbol{f}_k = \boldsymbol{f}(t_k, \boldsymbol{x}_k, \boldsymbol{u}_k)$ is the outcome of evaluating the dynamics of the system at every collocation point.

4.2.3 Constraints

Besides the collocation constraints, which implement the dynamics of the system, there could also be limits on the state and the control, path constraints and boundary constraints. All of these constraints are managed by implementing them at particular collocation points. For instance, simple constraints on state and control can be approximated as follows:

$$\boldsymbol{x} < \boldsymbol{0} \qquad \rightarrow \qquad \boldsymbol{x}_k < \boldsymbol{0} \qquad \forall k \tag{4.13}$$

$$\boldsymbol{u} < \boldsymbol{0} \qquad \rightarrow \qquad \boldsymbol{u}_k < \boldsymbol{0} \qquad \forall k \tag{4.14}$$

Path constraints are managed in the same way:

$$\boldsymbol{g}(t, \boldsymbol{x}, \boldsymbol{u}) < \boldsymbol{0} \qquad \rightarrow \qquad \boldsymbol{g}(t_k, \boldsymbol{x}_k, \boldsymbol{u}_k) < \boldsymbol{0} \qquad \forall k \tag{4.15}$$

Moreover, boundary constraint are also enforced at the first and last collocation points:

$$\boldsymbol{h}(t_0, \boldsymbol{x}(t_0), \boldsymbol{u}(t_0)) < \boldsymbol{0} \qquad \rightarrow \qquad \boldsymbol{h}(t_0, \boldsymbol{x}_0, \boldsymbol{u}_0) < \boldsymbol{0} \qquad \forall k \tag{4.16}$$

When dealing with constraints, the following points must be takes into consideration:

- Trajectory optimization problems with path constraints are likely to be considerably harder to solve than trajectory optimization problems without path constraints. The details are beyond the scope of this paper. However, they are properly explained by Betts [68].
- The limits of the trajectories are always considered as collocation points in trapezoidal collocation. There exist different methods for which the trajectory limits are not considered as collocation points. For these methods, special attention must be given when boundary constraints are being handled ([69, 70].

4.2.4 Interpolation

Trapezoidal collocation operates by approximating the trajectory of the dynamics of the system and the control trajectory as piece-wise linear functions, which are also called *linear splines*, shown in figure 4.1. When building a spline, the term *knot point* is utilized to represent every point that combines two polynomial segments. The knot points related to the spline are coincident in the case of trapezoidal collocation.

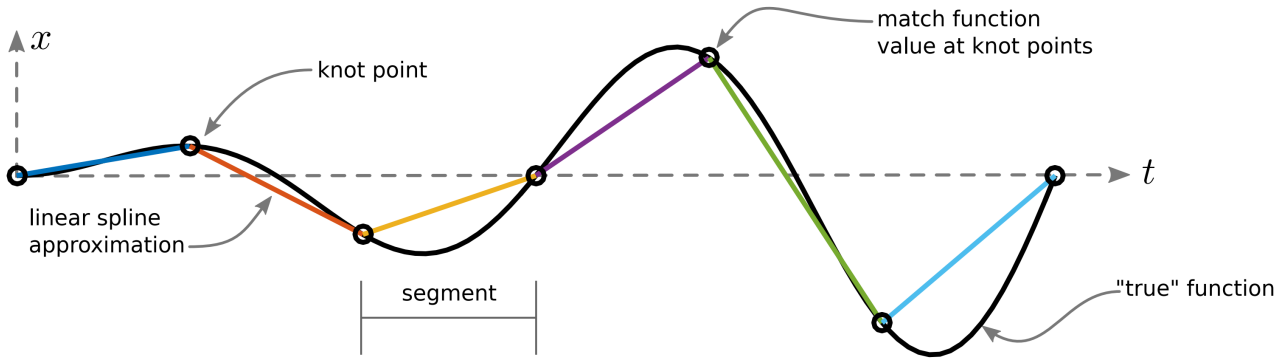


Figure 4.1: Function approximation using a linear spline [12].

First, the control strategy must be constructed, which is a simple spline that is linear. Since both the time and control are known at each knot point, then it is easy to derive the expression for $\boldsymbol{u}$ on the interval $t \in [t_k, t_{k+1}]$. In order to keep the math readable,τ and h_k are defined such that:

$$\tau = t - t_k \qquad\qquad h_k = t_{k+1} - t_k$$

Thus, the expression of $\boldsymbol{u}$ is the following:

$$\boldsymbol{u}(t) \approx \boldsymbol{u}_k + \frac{\tau}{h_k}(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k) \tag{4.17}$$

The state trajectory is expressed by a *quadratic spline* (a piece-wise quadratic function). This may seem perplexing, however it follows directly from the collocation equations 4.12. The trapezoidal collocation equations are accurate when the dynamics of the system vary linearly between any two knot points, which is a fact that is used to approximate the dynamics of the system over a single segment $t \in [t_k, t_{k+1}]$ as expressed below.

$$\boldsymbol{f}(t) = \dot{\boldsymbol{x}}(t) \approx \boldsymbol{f}_k + \frac{\tau}{h_k}(\boldsymbol{f}_{k+1} - \boldsymbol{f}_k) \tag{4.18}$$

However, what is interesting is $\boldsymbol{x}$ and not $\dot{\boldsymbol{x}}$, thus equation (4.18) must be integrated on both sides in order to obtain a quadratic expression for the state.

$$\boldsymbol{x}(t) = \int \dot{\boldsymbol{x}}(t)d\tau \approx \boldsymbol{c} + \boldsymbol{f}_k\tau + \frac{\tau^2}{2h_k}(\boldsymbol{f}_{k+1} - \boldsymbol{f}_k) \tag{4.19}$$

Then, the constant of integration $\boldsymbol{c}$ can be computed by utilizing the value of the state at the boundary $\tau = 0$ in order to get the final expression of the state.

$$\boldsymbol{x}(t) \approx \boldsymbol{x}_k + \boldsymbol{f}_k\tau + \frac{\tau^2}{2h_k}(\boldsymbol{f}_{k+1} - \boldsymbol{f}_k) \tag{4.20}$$

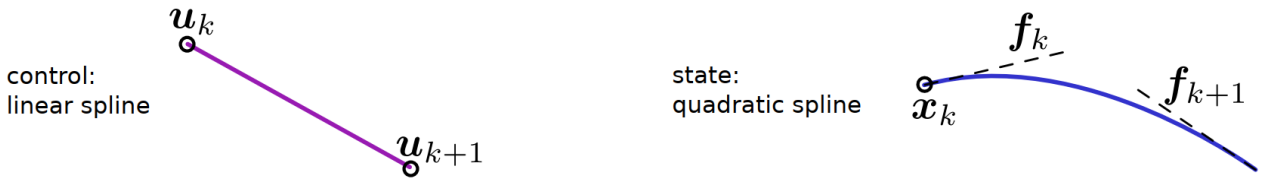


Figure 4.2: Representation of the linear and quadratic spline segments that are utilizied to approximate the control and state trajectories for trapezoidal collocation.

Figure 4.2 represents how a linear control segment and quadratic state segment are built. The spline equations (4.17) and (4.20) are particularly for trapezoidal collocation, since there is a one-to-one correlation between the collocation equations and the interpolation spline. Generally, if the control is a spline of order n, then the state is depicted by a spline of order $n+1$ [68].

4.3 Hermite-Simpson Collocation Method

The Hermite-Simpson collocation is comparable to the trapezoidal collocation, however it offers a solution that is higher-order accurate. This is due to the fact that trapezoidal collocation approximates the objective function and the dynamics of the system as piece-wise quadratic functions, as shown in figure 4.3. An added benefit of the Hermite-Simpson collocation method is that the trajectory is a cubic Hermite spline, which has a first-order derivative which is continuous.

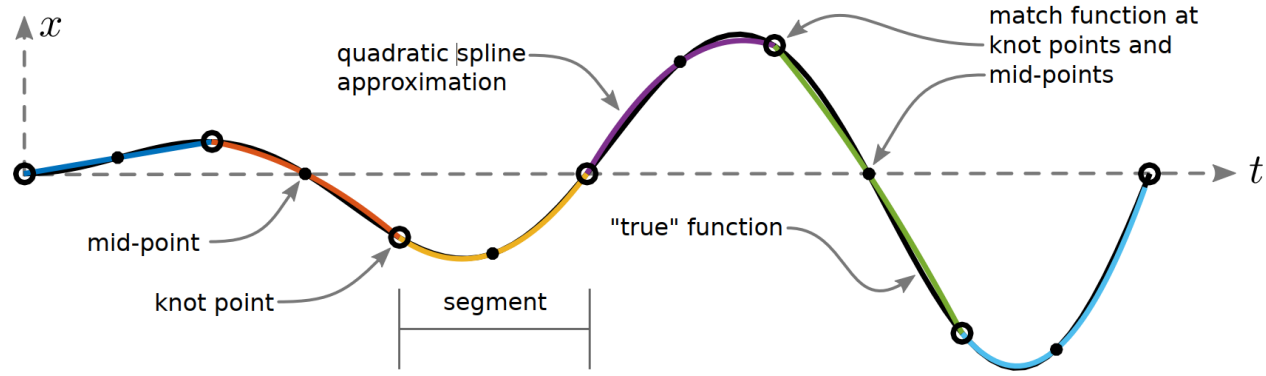


Figure 4.3: Function approximation using a quadratic spline. [12].

It can be observed that this approximation is significantly more accurate than the linear spline in figure 4.1, for the same number of segments.

4.3.1 Integrals

Integral expressions are regular in trajectory optimization problems, specifically in the objective function. The Hermite-Simpson collocation provide an approximation of the integrals by utilizing the Simpson quadrature. The Simpson quadrature, also called *Simpson's rule* for integration, which operates by approximating the integrand of the integral as a piece-wise quadratic function. This approximation of the integrand is provided below.

$$\int_{t_0}^{t_f} w(\tau)d\tau \quad \approx \quad \sum_{k=0}^{N-1} \frac{h_k}{6}(w_k + 4w_{k+\frac{1}{2}} + w_{k+1})$$

4.3.2 System Dynamics

Regardless of the collocation method that is being used, the *collocation constraints* are the group of constraints that are built to approximate the dynamics of the system. In the Hermite-Simpson collocation method, these constraints are built by writing the system dynamics in integral form: the change in state between any two knot points t_k must be equal to the integral of the dynamics of the system $\boldsymbol{f}(\cdot)$ between those points.

$$\dot{\boldsymbol{x}} = \boldsymbol{f} \tag{4.21}$$

$$\int_{t_k}^{t_{k+1}} \dot{\boldsymbol{x}} = \int_{t_k}^{t_k+1} \boldsymbol{f} dt \tag{4.22}$$

The transcription from continuous dynamics to a group of collocation equations takes place when the continuous integral is approximated in 4.22 with the Simpson quadrature and then implemented between every pair of knot points.

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}_k = \frac{1}{6} h_k (\boldsymbol{f}_k + 4\boldsymbol{f}_{k+\frac{1}{2}} + \boldsymbol{f}_{k+1}) \tag{4.23}$$

In fact, for the Hermite-Simpson collocation, a second collocation equation is required in addition to equation (4.23), in order to impose the dynamics of the system. This is due to fact that the dynamics at the mid-point of segment $\boldsymbol{f}_{k+\frac{1}{2}}$ are a function of the state $\boldsymbol{x}_{k+\frac{1}{2}}$, which is not known *a priori*. Thus, the state of the system can be computed at the mid-point by building an interpolant for the state trajectory and then assessing it at the mid-point of the interval.

$$\boldsymbol{x}_{k+\frac{1}{2}} = \frac{1}{2}(\boldsymbol{x}_k + \boldsymbol{x}_{k+1}) + \frac{h_k}{8}(\boldsymbol{f}_k - \boldsymbol{f}_{k+1}) \tag{4.24}$$

This second collocation equation (4.24) is particuar because it can be computed explicitly in function of the state at the knot points. So, it is possible to merge both equations (4.23) and(4.24) into just one complicated collocation constraint. When the transcription of the dynamics of the system is performed using only this single resulting collocation constraint, then the resulting formulation is said to be in *compressed form*. A different approach is to build an additional decision variable for the state at the mid-point $\boldsymbol{x}_{k+\frac{1}{2}}$, and the to use both equation (4.23) and (4.24) as constraint equations. When the collocation equations are are formed by utilizing this these two constraint equations, then this pair of equations is considered to be in *separated form*.

There are numerous trade-offs between the separated and the compressed forms of Hermite-Simpson collocation, which are explained in detail in [68]. The general rule of thumb is that the separated form is superior when working with a smaller number of segments, while the compressed form is superior when the number of segments is great. Both constraint equations (4.23) and (4.24) can be found in the book of Netts [68].

4.3.3 Constraints

Additionally to collocation constraints, which implement the dynamics of the system, there might also be limits on the state and control, path constraints and boundary constraints. These constraints are all handled by imposing them at particular collocation points. For instance, simple limits on the state of the system and the control are approximated as follows:

$$\boldsymbol{x} < \boldsymbol{0} \qquad \rightarrow \qquad \begin{cases} \boldsymbol{x}_k < \boldsymbol{0} \\ \boldsymbol{x}_{k+\frac{1}{2}} < \boldsymbol{0} \end{cases} \tag{4.25}$$

$$\boldsymbol{u} < \mathbf{0} \qquad \rightarrow \qquad \begin{cases} \boldsymbol{u}_k < \mathbf{0} \\ \boldsymbol{u}_{k+\frac{1}{2}} < \mathbf{0} \end{cases} \tag{4.26}$$

Path constraints are managed tackled in a similar manner: they are implemented at all the collocation points as shown below.

$$\boldsymbol{g}(t, \boldsymbol{x}, \boldsymbol{u}) < \mathbf{0} \qquad \rightarrow \qquad \begin{cases} \boldsymbol{g}(t_k, \boldsymbol{x}_k, \boldsymbol{u}_k) < \mathbf{0} \\ \boldsymbol{g}(t_{k+\frac{1}{2}}, \boldsymbol{x}_{k+\frac{1}{2}}, \boldsymbol{u}_{k+\frac{1}{2}}) < \mathbf{0} \end{cases} \tag{4.27}$$

Boundary constraint are also implemented at the first and last knot points:

$$\boldsymbol{h}\big(t_0, \boldsymbol{x}(t_0), \boldsymbol{u}(t_0)\big) < \mathbf{0} \qquad \rightarrow \qquad \boldsymbol{h}(t_0, \boldsymbol{x}_0, \boldsymbol{u}_0) < \mathbf{0} \tag{4.28}$$

Similarly to the case of trapezoidal collocation, trajectory optimization problems with path constraints always happen to be much more difficult to solve than trajectory optimization problems that do not have path constraints [68]. In addition, the boundaries of the trajectory are always collocation points. There exist some methods for which the trajectory boundaries are not collocation points. So, special care must be taked when handling boundary constraints for these methods [69, 70]

4.3.4 Interpolation

After solving the nonlinear program, the value of the state of the system and the control trajectories at each collocation point are now know. The subsequent step is to build a continuous trajectory to interpolate the solution between the collocation points. Similarly to what was done in trapezoidal collocation, a polynomial interpolant, which is derived from the collocation equations will be used.

Figure 4.4: Representation of the quadratic and cubic spline segments that are utilized to approximate the control and state trajectories for Hermite–Simpson collocation.

Hermite-Simpson collocation operates by using the Simpson quadrature in order to approximate every segment of the trajectory. The Simpson quadrature utilizes a quadratic segment, fitted through three equally spaced points in order to approximate the intergrand [12]. The general equation for quadratic interpolation is provided in *Numerical Recipes in C* [71], and it is demonstrated below for a curve $\boldsymbol{u}$(t) that passes through 3 points: $(t_A, \boldsymbol{u}_A)$, $(t_B, \boldsymbol{u}_B)$ and $(t_C, \boldsymbol{u}_C)$

$$\boldsymbol{u}(t) = \frac{(t-t_B)(t-t_C)}{(t_A-t_B)(t_A-t_C)}\boldsymbol{u}_A + \frac{(t-t_A)(t-t_C)}{(t_B-t_A)(t_B-t_C)}\boldsymbol{u}_B + \frac{(t-t_A)(t-t_B)}{(t_C-t_A)(t_C-t_B)}\boldsymbol{u}_C \tag{4.29}$$

For this specific case, this equation can be greatly simplified, because the points are uniformly distributed. Then, the points $k, k+\frac{1}{2}$ and k+1 can be used in place of A, B and C respectively. The next step is to recall from previous sections that $h_k = t_{k+1} - t_k$, and $t_{k+\frac{1}{2}} = \frac{1}{2}(t_k + t_{k+1})$. And, $\tau = t - t_k$. After doing the proper substitutions and performing some algebra, the following simplified equation for interpolating the control trajectory is the following:

$$\boldsymbol{u}(t) = \frac{2}{h_k^2}(\tau - \frac{h_k}{2})(\tau - h_k)\boldsymbol{u}_k - \frac{4}{h_k^2}(\tau)(\tau - h_k)\boldsymbol{u}_{k+\frac{1}{2}} + \frac{2}{h_k^2}(\tau)(\tau - \frac{h_k}{2})\boldsymbol{u}_{k+1} \tag{4.30}$$

Hermite-Simpson collocation also describes the dynamics of the system $\boldsymbol{f}(\cdot) = \dot{\boldsymbol{x}}$ by utilizing quadratic polynomials over each segment. Thus, the quadratic interpolation formula that was developed for the control trajectory can also be implemented to the dynamics of the system.

$$\boldsymbol{f}(t) = \dot{\boldsymbol{x}} = \frac{2}{h_k^2}(\tau - \frac{h_k}{2})(\tau - h_k)\boldsymbol{f}_k - \frac{4}{h_k^2}(\tau)(\tau - h_k)\boldsymbol{f}_{k+\frac{1}{2}} + \frac{2}{h_k^2}(\tau)(\tau - \frac{h_k}{2})\boldsymbol{f}_{k+1} \tag{4.31}$$

Generally, the main interest is to obtain an expression for the state trajectory $\boldsymbol{x}(t)$ instead of its derivative $\dot{\boldsymbol{x}}(t)$. In order to obtain the state trajectory, equation 4.31 will be integrated, after arranging the expressing for it to be in standard polynomial form, the following expression is then obtained.

$$\boldsymbol{x}(t) = \int \dot{\boldsymbol{x}}(t) = \int \left[\boldsymbol{f}_k + \left(-3\boldsymbol{f}_k + 4\boldsymbol{f}_{k+\frac{1}{2}} - \boldsymbol{f}_{k+1}\right)\left(\frac{\tau}{h_k}\right) + \left(2\boldsymbol{f}_k - 4\boldsymbol{f}_{k+\frac{1}{2}} + 2\boldsymbol{f}_{k+1}\right)\left(\frac{\tau}{h_k}\right)^2\right] dt \tag{4.32}$$

The integral can be calculated by using basic calculus. Then, the constant of integration is solved by using the boundary condition $\boldsymbol{x}(t_k) = \boldsymbol{x}_k$. The obtained expression is provided below. Moreover, this expression allows the interpolation of the state trajectory.

$$\boldsymbol{x}(t) = \boldsymbol{x}_k + \boldsymbol{f}_k\left(\frac{\tau}{h_k}\right) + \frac{1}{2}\left(-3\boldsymbol{f}_k + 4\boldsymbol{f}_{k+\frac{1}{2}} - \boldsymbol{f}_{k+1}\right)\left(\frac{\tau}{h_k}\right)^2 + \frac{1}{3}\left(2\boldsymbol{f}_k - 4\boldsymbol{f}_{k+\frac{1}{2}} + 2\boldsymbol{f}_{k+1}\right)\left(\frac{\tau}{h_k}\right)^3 \tag{4.33}$$

The interpolant of the state and control trajectories are represented in figure 4.4.

Conclusion

As can be deduced from the previous chapters, performing multi-flip maneuvers with quadrotors is not an easy task to handle. There are several aspects that must be studied and tuned in order to properly control the quadrotor and perform the multi-flip maneuver. First, the dynamic model of the quadrotor must be very thorough and robust in order to have a good representation of the system in simulations and to reduce the errors which result from aerodynamic effects that are usually hard to properly estimate. But, on the other hand, the complexity of the dynamics of the quadrotor must be reduced and simplified in order to satisfy the real-time limitations of the embedded control loop. This is why the very first step to do in this master thesis is to perform the identification of the base dynamic parameters for the quadrotor, since some of the standard parameters of the system have no effect on the dynamic model. So, these irrelevant dynamic parameters can be eliminated and other parameters can also be grouped together, which will greatly reduce the computational cost, regardless of which controller will be used. Furthermore, several control methods which can be used during this master thesis have been discussed. However, the most interesting control method to be used is model predictive control (MPC), since MPC is a closed-loop controller which will be able to account for modeling errors and disturbances on the system. In addition, MPC will solve an optimization problem at each time step in order to select the optimal control. However, MPC controllers must be properly tuned in order to be able to solve the optimization problem at each time step in real time. This is why simulations are required, since they will be very useful for tuning the control parameters in order to find the best balance between system stability, performance and robustness. Finally, the trajectory optimization method, which is an open-loop optimization method that was proven to be successful by Marino [56] for a single-flip maneuver, can be revisited for further experimentation and to add the supplementary constraint of performing the multi-flip maneuvers in constrained environments.

Planning The milestones for this master thesis are organized as follows:

- Perform the identification of the base dynamic parameters of the quadrotor that will be used for the experiments, namely the *"Crazyflie 2.0"*.
- Design a model predictive controller and tune the parameters in a way that the controller can be used on the Crazyflie 2.0 in real-time applications.
- Perform simulations using ROS and Gazebo to make sure that the controller is working properly.
- Perform real experimentation in the LS2N lab.
- Compare the performance of MPC to the performance of trajectory optimization techniques (Closed-loop Vs Open-loop).

Bibliography

[1] I. Fantoni, "Lecture notes in modelling of terrestrial and aerial vehicles," December 2016.

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

[3] Mohammed Dahleh, Munther A. Dahleh, and George Verghese, *Lectures on Dynamic Systems and Control*. MIT, 2011.

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

[5] R. W. Erickson and D. Maksimović, *AC Equivalent Circuit Modeling*. Cham: Springer International Publishing, 2020, pp. 215–275. [Online]. Available: https://doi.org/10.1007/978-3-030-43881-4_7

[6] MathWorks, "How to run mpc faster," https://www.mathworks.com/videos/understanding-model-predictive-control-part-5-how-to-run-mpc-faster-1533108818950.html, 2018, from the Model Predictive Control Toolbox.

[7] R. DeCarlo and S. Zak, "A quick introduction to sliding mode control and its applications 1," 2008.

[8] F. Oliva-Palomo, A. Sanchez-Orta, P. Castillo, and H. Alazki, "Nonlinear ellipsoid based attitude control for aggressive trajectories in a quadrotor: closed-loop multiflips implementation," *Control Engineering Practice*, pp. 150–161, 2018.

[9] J. H. Gillula, H. Huang, M. P. Vitus, and C. J. Tomlin, "Design and analysis of hybrid systems, with applications to robotic aerial vehicles," *Robotics Research*, pp. 139–149, 2011.

[10] S. Lupashin and R. D'Andrea, "Adaptive fast open-loop maneuvers for quadrocopters," *Autonomous Robots*, pp. 89–102, 2012.

[11] Y. Chen and N. O. P´erez-Arancibia, "Generation and real-time implementation of high-speed controlled maneuvers using an autonomous 19-gram quadrotor," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3204–3211, 2016.

[12] M. Kelly, "An introduction to trajectory optimization, how to do your own direct collocation," *SIAM*, pp. 1–30, 2017.

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

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

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

[16] P. C. Goldstein, H., and J. Safko, *Classical Mechanics.* London: Adisson Wesly Series in Physics, Adison-Wesley,U.S.A. World Scientific, 1980.

[17] S. Houston, *Bramwell's Helicopter Dynamics-second edition A.R.S. Bramwell.* Cambridge University Press, 2001, vol. 105, no. 1051.

[18] R. M. et al., "A mathematical introduction to robotic manipulation," *CRC*, 1994.

[19] B. Etkin and L. Reid, *Dynamics of Flight : Stability and Control.* John Wiley and Sons, 1996.

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

[21] 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.

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

[23] 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.

[24] 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.

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

[26] 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.

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

[28] Klaus Schmitt, "Periodic solutions of nonlinear differential systems," *Journal of Mathematical Analysis and Applications*, pp. 174 – 182, 1972.

[29] D. Melchor-Aguilar, "On the lyapunov's indirect method for scalar differential-difference equations," *IFAC Proceedings Volumes*, vol. 37, no. 21, pp. 175 – 180, 2004, 2nd IFAC Symposium on System Structure and Control, Oaxaca, Mexico, December 8-10, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667017304640

[30] 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.

[31] 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.

[32] ——, "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.

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

[34] 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.

[35] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789 – 814, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109899002149

[36] A. Bemporad, L. Chisci, and E. Mosca, "On the stabilizing property of siorhc," *Automatica*, vol. 30, no. 12, pp. 2013 – 2015, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0005109894900647

[37] J. Zeman and B. Rohá I'-Ilkiv, "Robust model predictive control of linear time invariant system with disturbances," *IFAC Proceedings Volumes*, vol. 36, no. 18, pp. 325 – 332, 2003, 2nd IFAC Conference on Control Systems Design (CSD '03), Bratislava, Slovak Republic, 7-10 September 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S147466701734689X

[38] M. Bujarbaruah, X. Zhang, U. Rosolia, and F. Borrelli, "Adaptive mpc for iterative tasks," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 6322–6327.

[39] L. Cui, L. Chen, and D. Duan, "Gain-scheduling model predictive control for unmanned airship with lpv system description," *Journal of Systems Engineering and Electronics*, vol. 26, no. 5, pp. 1043–1051, 2015.

[40] Naoyuki Hara and Akira Kojima, "Reduced order model predictive control - an approach based on system decomposition -," in *SICE Annual Conference 2007*, 2007, pp. 2226–2229.

[41] P. Bemporad, *Explicit Model Predictive Control*. London: Springer London, 2013, pp. 1–9.

[42] S. Hovland and J. Gravdahl, "Complexity reduction in explicit mpc through model reduction," vol. 17, 07 2008.

[43] T. Mumcu and K. Gulez, "Suboptimal solutions for time varying time delayed mpc controllers," *Electronics and Electrical Engineering*, vol. 20, 02 2014.

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

[45] 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.

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

[47] 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.

[48] 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.

[49] A. Levant, "Sliding order and sliding accuracy in sliding mode control," *International Journal of Control, 58(6), 1993*, pp. 1247–1263, 1993.

[50] G. Bartolini, A. Ferrara, and A. L. E. Usai, "On second order sliding mode controller" in variable structure systems, sliding mode and nonlinear control," *Springer Lecture Notes in Control and Information Sciences*, vol. 247, pp. 1247–1263, 1999.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] M. Orsingher, "Making aggressive maneuvers with drones thanks to parallel singularity crossing approaches." Master's thesis, 2019.

[56] A. Marino, "Robust and feasible trajectory generation of aggressive maneuvers for quadrotors," Master's thesis, 2020.

[57] J. H. Gillula, H. Huang, M. P. Vitus, and C. J. Tomlin, "Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice," *2010 IEEE international conference on Robotics and Automation (ICRA)*, pp. 1649–1654, 2010.

[58] S. Lupashin, A. Sch¨ollig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1642–1648, 2010.

[59] S. Lupashin and R. D'Andrea, "Adaptive open-loop aerobatic maneuvers for quadrocopters," *IFAC Proceedings Volumes*, pp. 2600–2606, 2011.

[60] T. Lee, M. Leoky, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," *49th IEEE Conference on Decision and Control (CDC)*, pp. 5420–5425, 2010.

[61] A. A. El-Badawy and M. A. Bakr, "Quadcopter aggressive maneuvers along singular configurations: An energy-quaternion based approach," *Journal of Control Science and Engineering*, 2016.

[62] L. Wang and J. Su, "Switching control of attitude tracking on a quadrotor uav for large-angle rotational maneuvers," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2907–2912, 2014.

[63] Y. Chen and N. O. P´erez-Arancibia, "Lyapunov-based controller synthesis and stability analysis for the execution of high-speed multi-flip quadrotor maneuvers," *2017 American Control Conference (ACC)*, pp. 3599–3606, 2017.

[64] Y. Chen and N. O. Perez-Arancibia, "Nonlinear adaptive control of quadrotor multiflipping maneuvers in the presence of time-varying torque latency," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–9, 2018.

[65] J. T. Betts, "A survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, pp. 1–56, 1998.

[66] M. A. Patterson and A. V. Rao, "Gpops ii : A matlab software for solving multiple-phase optimal control problems using hp adaptive gaussian quadrature collocation methods and spa and rse nonlinear programming," pp. 1–41, 2013.

[67] A. Rao, "A survey of numerical methods for optimal control," *Advances in the Astronautical Sciences*, pp. 497–528, 2009.

[68] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Siam, Philadelphia, PA,: Springer, 2010.

[69] D. a. Benson, G. T. Huntington, T. P. Thorvaldsen, and A. V. Rao, "Direct trajectory optimization and costate estimation via an orthogonal collocation method," *Journal of Guidance, Control, and Dynamics, 29*, pp. 1435–1440, 2006.

[70] D. Garg, M. Patterson, W. W. Hager, A. V. Rao, D. a. Benson, and G. T. Huntington, "A unified framework for the numerical solution of optimal control problems using pseudospectral methods," *Automatica, 46*, p. 1843–1851, 2010.

[71] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C - 2nd edition*. Cambridge, GB: Cambridge University Press, 1992.

[72] P. Mullhaupt, "Analysis and control of underactuated mechanical nonminimum-phase systems," Ph.D. dissertation, EPFL, 1999.

[73] S. B. et al, "Design and control of an indoor micro quadrotor," *Proc. (IEEE) International Conference on Robotics and Automation (ICRA'04)*, 2004.

[74] S. Bouabdallah and R. Siegwart, "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor," *Proc. (IEEE) International Conference on Robotics and Automation (ICRA'05)*, 2005.

[75] S.Bouabdallah and R. Siegwart, "Towards intelligent miniature flying robots," *Proc. of Field and Service Robotics*, 2005.

[76] J. G. Leishman, "Principles of helicopter aerodynamics," *Principles of Helicopter Aerodynamics*.

[77] J. Leishman, "Principles of helicopter aerodynamics," *Cambridge University Press*.

[78] N. G. et al., "Control laws for the tele operation of an unmanned aerial vehicle known as an x4-flyer," *roc. (IEEE) International Conference on Intelligent Robots (IROS'06)*, 2006.

[79] I. Cheeseman and W. Bennett, "The effect of the ground on a helicopter rotor in forward flight," *Aeronautical Research Council, no. 3021*, 1957.

[80] D. Griffiths and J. Leishman, "A study of dual-rotor interference and ground effect using a free-vortex wake model," *Proc. of the 58th Annual Forum of the Ameriacan Helicopter Society*, 2002.

[81] R. Findeisen and F. Allgöwer, "An introduction to nonlinear model predictive control," 01 2002.

[82] B. Bequette, *Process Control: Modeling, Design, and Simulation*, ser. Prentice-Hall international series in the physical and chemical engineering sciences. Prentice Hall PTR, 2003. [Online]. Available: https://books.google.com.lb/books?id=PdjHYm5e9d4C

[83] A. Grancharova and T. A. Johansen, *Explicit MPC of Constrained Nonlinear Systems with Quantized Inputs*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 111–125.