ÉCOLE CENTRALE DE NANTES

MASTER CORO-IMARO
"CONTROL AND ROBOTICS"

2020 / 2021

Bibliography Report

Presented by

Elie Hatem

On January 11, 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 meanWs 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 studied to create general feasible trajectories that will allow quadrotors to perform aggressive flip maneuvers while passing through a constrained environment and while guaranteeing a satisfactory degree of robustness to the uncertainties of the dynamic model.

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 for me the confidence 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
C_{R_m}	Rolling moment coefficient
C_T	Thrust coefficient
H	Hub force
CT	Continuous time
DT	Discrete time

Abbreviations

IGE	In Ground Effect
OGE	Out of Ground Effect

List of Figures

List of Tables

Contents

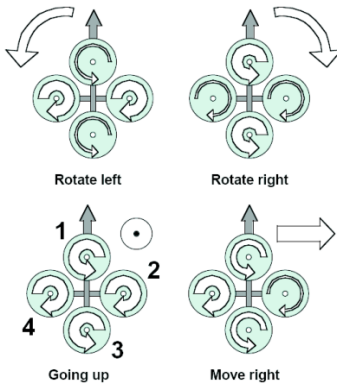
Introduction

The aim of this section is to provide a general summary of the robotic 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 it to be controller 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) Typical quadrotor configuration The width of the arrows is proportional to the angular speed of the propellers.[1]

Figure 1: A commercial quadrtotor platform, with a typical quadrotor configuration.

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.

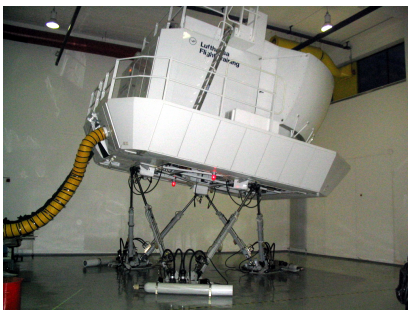
2. As quadrotors became more and more affordable and dependable, the number of quadrotors real-world applications has grown significantly. They are being used for aerial photography, agriculture, surveillance, inspection tasks, in addition to many other uses as well.

3. Quadrotors are quite agile and maneuverable during flight. Especially when compared to other types of unmanned aerial vehicles (UAVs).

However, on 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, in addition to several aerodynamic effects that will become important 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 very 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.[7]



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

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

Figure 2: Two examples of parallel robots.

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

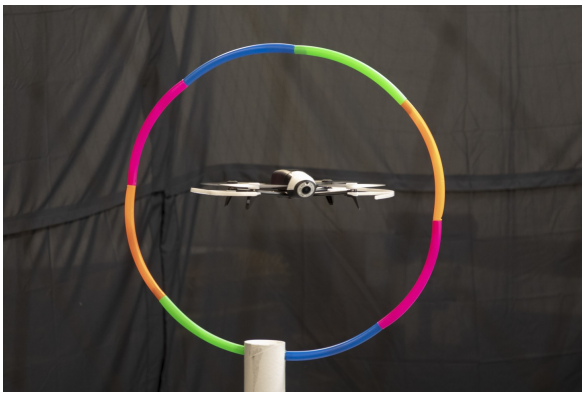
[2] https://en.wikipedia.org/wiki/Parallel_manipulator#/media/File:Prototype_robot_parall%C3%A8le_PAR4.jpg, accessed on 01/08/2021.

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 it the mechanism could even break. The main difference between serial and parallel manipulators is that singularity configurations may also appear inside the robot workspace (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 manipulator 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 must also do the flips in a constrained environment.

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



(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 also 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 goal of this chapter is to present the dynamic model of the quadrotor. The mathematical notation and the physics of the quadrotor platform are expressed using the Newton-Euler formalism. Then, the state-space model that will be coded on the controller of the quadrotor will be derived.

1.1 Concepts and Generalities

The dynamic model of the quadrotor will be derived 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.

The helicopter is a complex mechanical system, it gathers many physical effects from the domain of mechanics and aerodynamics [9]. Thus, all the significant effects including the gyroscopic effects must be considered in the modeling of the quadrotor. A small list of the most important effects that a helicopter is subject to [10] are briefly described in table 1.1:

Table 1.1: The main physical effects that the helicopter is subject to.

Effect	Source	formulation
Aerodynamic effects	Rotation of propeller Flapping of blades	$C\Omega^2$
Inertial counter torques Gravitational effect	Change in propeller spinning speed Position of the center of mass	$J\dot{\Omega}$
Gyroscopic effects	Orientation change of the rigid body Orientation change of the propeller plane	$I\theta\psi$ $J\Omega_r\theta, \phi$
Friction	All helicopter motions	$C\dot{\phi}, \dot{\theta}, \dot{\psi}$

1.2 Modelling with Euler-Lagrange Formalism

The dynamics of the rotation of a simple quadrotor are modeled using the Euler-Lagrange Formalism in this section. A fixed frame E for the world frame and body fixed frame B for the quadrotor are considered as represented in figure 1.1. The orientation of the quadrotor frame in space is provided by a rotation R from B to E, where $R \in SO3$ is a 3×3 rotation matrix.

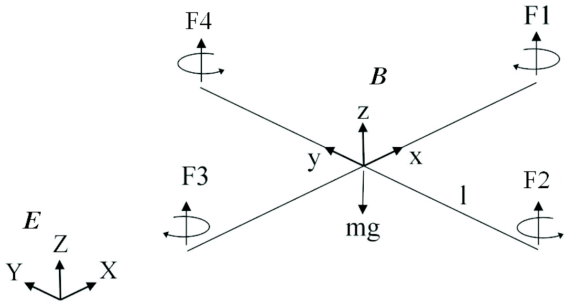


Figure 1.1: Simplified coordinate system of a quadrotor. [1]

1.2.1 Kinematics

For any point of the body frame of the quadrotor expressed in the fixed world frame, the following can be written (c: cos, s: sin):

$$\begin{cases} r_X = (c\psi c\theta)x + (c\psi s\theta s\phi - s\psi c\phi)y + (c\psi s\theta c\phi + s\psi s\phi)z \\ r_Y = (s\psi c\theta)x + (s\psi s\theta s\phi + c\psi c\phi)y + (s\psi s\theta c\phi - c\psi s\phi)z \\ r_Z = (-s\theta)x + (c\theta s\phi)y + (c\theta c\phi)z \end{cases} \tag{1.1}$$

Thus, the velocities can be derived by differentiation 1.1, and the squared magnitude of the squared velocity can be expressed as follows for any point:

$$v^2 = v_X^2 + v_Y^2 + v_Z^2 \tag{1.2}$$

1.2.2 Energy

Assuming that the matrix of inertia is diagonal, then from equation 1.2, the expression of the kinetics energy can be calculated:

$$T = \frac{1}{2}I_{xx}(\dot{\phi} - \dot{\psi}s\theta)^2 + \frac{1}{2}I_{yy}(\dot{\theta}c\phi + \dot{\psi}s\phi c\theta)^2 + \frac{1}{2}I_{zz}(\dot{\theta}s\phi - \dot{\psi}c\phi)^2 \tag{1.3}$$

Using the formula of the potential energy, equation 1.3 can be expressed in the fixed world frame as:

$$V = \int xdm(x)(-gs\theta) + \int ydm(y)(gs\phi c\theta) + \int zdm(z)(gc\phi c\theta) \tag{1.4}$$

1.2.3 Equation of Motion

By using the Euler-Lagrange formalism:

$$L = T - V \quad , \quad \Gamma_i = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} \tag{1.5}$$

Where L Γ_i and $\dot{q}_i$ are the Lagrangian, the generalized forces and the generalized coordinates respectively. Thus, the equations of motion can be expressed as follows:

$$\begin{cases} I_{xx}\ddot{\phi} = \dot{\theta}\dot{\psi}(I_{yy} - I_{zz}) \\ I_{yy}\ddot{\theta} = \dot{\phi}\dot{\psi}(I_{zz} - I_{xx}) \\ I_{zz}\ddot{\psi} = \dot{\phi}\dot{\theta}(I_{xx} - I_{yy}) \end{cases} \tag{1.6}$$

Moreover, the torques that are nonconservative and acting on the quadrotor, are due to two different causes. First, it is due to thrust of each rotor pairs in figure 1.1:

$$\begin{cases} \tau_x = bl(\Omega_4^2 - \Omega_2^2) \\ \tau_y = bl(\Omega_3^2 - \Omega_1^2) \\ \tau_z = bl(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \end{cases} \tag{1.7}$$

Second, it is also due to the gyroscopic effect which is the result of the rotation of the propellers:

$$\begin{cases} \tau'_x = J_r\omega_y(\Omega_1 + \Omega_3 - \Omega_2 - \Omega_4) \\ \tau'_y = J_r\omega_x(\Omega_2 + \Omega_4 - \Omega_1 - \Omega_3) \end{cases} \tag{1.8}$$

1.2.4 The Derived Dynamic Model

The dynamic model of the quadrotor which describes the rotations of roll, pitch and yaw consists of three terms:

1. The actuator torques.
2. The gyroscopic effects that are due to the rotation of the rigid body.
3. The gyroscopic effects that are due to rotation of the propeller that is coupled with the rotation of the body.

Thus, the dynamic model of the quadrotor is:

$$\begin{cases} I_{xx}\ddot{\phi} = \dot{\theta}\dot{\psi}(I_{yy} - I_{zz}) - J\dot{\theta}\Omega_r + \tau_x \\ I_{yy}\ddot{\theta} = \dot{\theta}\dot{\psi}(I_{zz} - I_{xx}) + J\dot{\phi}\Omega_r + \tau_y \\ I_{zz}\ddot{\psi} = \dot{\phi}\dot{\theta}(I_{xx} - I_{yy}) + \tau_z \end{cases} \tag{1.9}$$

1.2.5 Rotor Dynamics

DC motors are used to drive the rotors of a quadrotor. So, the established equations of a DC motor are the following:

$$\begin{cases} L\frac{di}{dt} = u - R_{mot}i - k_e\omega_m \\ \\ J_m\frac{d\omega_m}{dt} = \tau_m - \tau_d \end{cases} \tag{1.10}$$

Since small motors are used in which they also have little inductance, then the second order equation of the DC motor dynamics is given by:

$$J_m\frac{d\omega_m}{dt} = -\frac{k_m^2}{R_{mot}}\omega_m - \tau_d + \frac{k_m}{R_{mot}}u \tag{1.11}$$

When the gearbox and the propeller models are introduced, then equation 1.12 becomes:

$$\begin{cases} \dot{\omega}_m = -\frac{1}{\tau}\omega_m - \frac{d}{\eta r^3 J_t}\omega_m^2 + \frac{1}{k_m\tau}u \\ \\ \frac{1}{\tau} = \frac{k_m^2}{RJ_t} \end{cases} \tag{1.12}$$

Moreover, linearization of equation 1.12 can be done around an operation point $\dot{\omega}_0$ to the form $\dot{\omega}_m = -A\omega_m + Bu + C$ with:

$$A = \left(\frac{1}{\tau} + \frac{2d\omega_0}{\eta r^3 J_t}\right),\ B = \left(\frac{1}{k_m\tau}\right),\ C = \left(\frac{d\omega_0^2}{\eta r^3 J_t}\right) \tag{1.13}$$

1.3 Modeling with Newton-Euler Formalism

The model above was derived in succession as shown in papers [11, 12, 13] . The dynamic equations below include rolling moments R_m, hub forces H and various aerodynamic effects. Thus, this is a more realistic dynamic model, especially when the quadrotor flies in a forward manner. With the previous versions of the dynamic model, it was required to to moderately tune the control parameters in order to have experiments that are successful.

The dynamic model expressed in the Newton-Euler formalism of a rigid body that is subject to external forces acting on the center of mass [14] is expressed as follows:

$$\begin{bmatrix} mI_{3\times3} & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} \dot{V} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} \omega \times mV \\ \omega \times I\omega \end{bmatrix} = \begin{bmatrix} F \\ \tau \end{bmatrix} \tag{1.14}$$

Considering a fixed world frame E and a fixed body frame B on the quadrotor as shown in figure 1.2. Then, by the use of the Euler angles, the orientation of the rigid body of the quadrotor in space is expressed by a rotation R from B to E, where $R \in SO3$ is a rotation matrix.

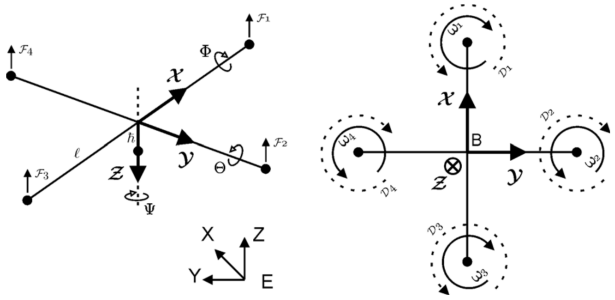


Figure 1.2: Detailed coordinate system showing the forces with the roll, pitch and yaw angles (left) and the moments (right) applied on a quadrotor. [1]

1.3.1 Aerodynamic Forces and Moments

The aerodynamic forces and moments are computed using a mix of blade element and momentum theory [15]. This is based off of the work of Gary Fay during the project of the Mesicopter [16]. For a simpler readings of the equations provided below, some symbols are recalled:

σ: solidity ration	λ: inflow ratio
a: lift slope	v: induced velocity
μ: rotor advance ration	ρ: air density

Thrust Force

The thrust force is due to all the vertical forces that the blade elements are subject to.

$$\begin{cases} T = C_T \rho A (\Omega R_{rad})^2 \\ \\ \frac{C_T}{\sigma a} = (\frac{1}{6} + \frac{1}{4}\mu^2)\theta_0 - (1+\mu^2)\frac{\theta_{tw}}{8} - \frac{1}{5}\lambda \end{cases} \tag{1.15}$$

Hub Force

The hub force is due to all the horizontal forces that the blade elements are subject to.

$$\begin{cases} H = C_H \rho A (\Omega R_{rad})^2 \\ \\ \frac{C_H}{\sigma a} = (\frac{1}{4a}\mu\overline{C_d} + \frac{1}{4}\lambda\mu(\theta_0 - \frac{\theta_{tw}}{2}) \end{cases} \tag{1.16}$$

Drag Moment

The drag moment about the rotor shaft is due to the aerodynamic forces that the blade elements are subject to. The horizontal forces that are acting on the rotor are multiplied by the moment arm and integrated over the rotor. The drag moment gives the required power to spin the rotor.

$$\begin{cases} Q = C_Q \rho A (\Omega R_{rad})^2 R_{rad} \\ \\ \frac{C_Q}{\sigma a} = \frac{1}{8a}(1+\mu^2)\overline{C_d} + \lambda(\frac{1}{6}\theta_0 - \frac{1}{8}\theta_{tw} - \frac{1}{4}\lambda) \end{cases} \tag{1.17}$$

Rolling moment

The rolling moment of a propeller occurs when the blade that is advancing is producing more lift than the blade that is retreating in forwarding flight. It is the integration of the lift of every single section that is acting at a given radius over the entire rotor. The reader should notice that the rolling moment is not the same as the propeller radius, or the overall rolling moment which is due to other effects or the rotation matrix R. So, there should not be any confusion.

$$\begin{cases} R_m = C_{R_m}\rho A(\Omega R_{rad})^2 R_{rad} \\ \\ \frac{C_{R_m}}{\sigma a} = -\mu(\frac{1}{6}\theta_0 - \frac{1}{8}\theta_{tw} - \frac{1}{8}\lambda) \end{cases} \tag{1.18}$$

Ground Effect

When operating near the ground (at a height equivalent to half the diameter of the rotor), helicopters experience thrust augmentation which is caused by greater efficiency of the rotor.This is linked to a decrease in the velocity of induced airflow. Moreover, this is called Ground Effect. Different approaches to deal with this effect can be found in literature, for example, adaptive techniques can be used [17]. However, the objective is to find a model that is simple and mainly captures the change in the velocity of the induced inflow. Cheeseman [18] states (reached from the images method [19]) that if the power is constant ($T_{OGE}v_{i,OGE} = T_{IGE}v_{i,IGE}$), the generated vecolity at the center of the rotor by its imageis $\delta v_i = Av_i/16\pi z^2$. Cheesman acquired the relation (1.19) by using the assumption that both v_i and δv_i are constant over disk, which results in $v_{i,IGE} = v_i - \delta v_i$.

$$\frac{T_{IGE}}{T_{OGE}} = \frac{1}{1 - \frac{R_{rad}^2}{16z^2}} \tag{1.19}$$

An alternative way to move forward is to consider that the inflow ration of the inflow ratio is $\lambda_{IGE} = (v_{i,OGE} - \delta v_i - \dot{z})/\Omega R_{rad}$, where the change of the velocity of the induced inflow is $\delta v_i = v_i/(4z/R_{rad})^2$.Then, the thrust coefficient (1.15) IGE can be rewritten as:

$$\begin{cases} T_{IGE} = C_T^{IGE}\rho A(\Omega R_{rad})^2 \\ \\ \frac{C_T^{IGE}}{\sigma a} = \frac{C_T^{OGE}}{\sigma a} + \frac{\delta v_i}{4\Omega R_{rad}} \end{cases} \tag{1.20}$$

1.3.2 General Moments and Forces

The motion of the quadrotor is the result of several forces and moments that are originating from different physical effects [1]. In this model, the following effects are considered (with c: cos, s:sin).

Rolling Moments

body gyro effect	$\dot{\theta}\dot{\psi}(I_{yy} - I_{zz})$
propeller gyro effect	$J_r\dot{\theta}\Omega_r$
pitch actuators action	$l(-T_2 + T_4)$
hub moment due to forward flight	$h(\sum_{i=1}^{4} H_{yi})$
rolling moment due to forward flight	$(-1)^{i+1}\sum_{i=1}^{4} R_{mxi}$

Pitching Moments

body gyro effect	$\dot{\phi}\dot{\psi}(I_{zz} - I_{xx})$
propeller gyro effect	$J_r\dot{\phi}\Omega_r$
pitch actuators action	$l(T_1 - T_3)$
hub moment due to forward flight	$h(\sum_{i=1}^{4} H_{xi})$
rolling moment due to side-ward flight	$(-1)^{i+1}\sum_{i=1}^{4} R_{myi}$

Yawing Moments

body gyro effect	$\dot{\theta}\dot{\phi}(I_{xx} - I_{yy})$
inertial counter-torque	$J_r\dot{\Omega}_r$
counter-torque unbalance	$(-1)^{i}\sum_{i=1}^{4} Q_i$
hub force unbalance in forward flight	$l(H_{x2} - H_{x4})$
hub force unbalance in sideward flight	$l(-H_{y1} + H_{y3})$

Forces Along z Axis

actuators action	$c\psi c\phi(\sum_{i=1}^{4} T_i$
weight	mg

Forces Along x Axis

actuators action	$(s\psi s\phi + c\psi s\theta c\phi)(\sum_{i=1}^{4} T_i)$
hub force in x axis	$-\sum_{i=1}^{4} H_{xi}$
friction	$\frac{1}{2} C_x A_c \rho \dot{x} \lvert\dot{x}\rvert$

Forces Along y Axis

actuators action	$(-c\psi s\phi + s\psi s\theta c\phi)(\sum_{i=1}^{4} T_i)$
hub force in y axis	$-\sum_{i=1}^{4} H_{yi}$
friction	$\frac{1}{2} C_y A_c \rho \dot{y} \lvert\dot{y}\rvert$

1.3.3 Equations of Motion

The equations of motion are derived from (1.14) in addition to all the forces and the moments that were listed in subsection 1.3.2.

$$\begin{cases} I_{xx}\ddot{\phi} = \dot{\theta}\dot{\psi}(I_{yy} - I_{zz}) + J_r\dot{\theta}\Omega_r + l(-T_2 + T_4) - h(\sum_{i=1}^{4} H_{yi}) + (-1)^{i+1}\sum_{i=1}^{4} R_{mxi} \\ I_{yy}\ddot{\theta} = \dot{\phi}\dot{\psi}(I_{zz} - I_{xx}) - J_r\dot{\phi}\Omega_r + l(T_1 - T_3) + h(\sum_{i=1}^{4} H_{xi}) + (-1)^{i+1}\sum_{i=1}^{4} R_{mxi} \\ I_{zz}\ddot{\psi} = \dot{\phi}\dot{\psi}(I_{xx} - I_{yy}) + J_r\dot{\Omega}_r + (-1)^{i}\sum_{i=1}^{4} Q_i + l(H_{x2} - H_{x4}) + l(-H_{y1} + H_{y3}) \\ m\ddot{z} = mg - (c\psi c\phi)\sum_{i=1}^{4} T_i \\ m\ddot{x} = (s\psi s\phi + c\psi s\theta c\phi)\sum_{i=1}^{4} T_i - \sum_{i=1}^{4} H_{xi} - \frac{1}{2} C_x A_c \rho \dot{x} \lvert\dot{x}\rvert \\ m\ddot{y} = (-c\psi s\phi + s\psi s\theta c\phi)\sum_{i=1}^{4} T_i - \sum_{i=1}^{4} H_{yi} - \frac{1}{2} C_y A_c \rho \dot{y} \lvert\dot{y}\rvert \end{cases} \tag{1.21}$$

1.4 State-Space Model

The model 1.21 that was developed in subsecton 1.3.3 expresses 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 and the hub forces and rolling moments are neglected. 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:

$$\mathbf{x} = \begin{bmatrix} \phi & \dot{\phi} & \theta & \dot{\theta} & \psi & \dot{\psi} & z & \dot{z} & x & \dot{x} & y & \dot{y} \end{bmatrix}^\intercal \tag{1.22}$$

With,

$$\begin{aligned} x_1 &= \phi & \qquad x_7 &= z \\ x_2 &= \dot{x}_1 = \dot{\phi} & x_8 &= \dot{x}_7 = \dot{z} \\ x_3 &= \theta & x_9 &= x \\ x_4 &= \dot{x}_3 = \dot{\theta} & x_{10} &= \dot{x}_9 = \dot{x} \\ x_5 &= \psi & x_{11} &= y \\ x_6 &= \dot{x}_5 = \dot{\psi} & x_{12} &= \dot{x}_{11} = \dot{y} \end{aligned} \tag{1.23}$$

Moreover, the control input vector has the following form:

$$\mathbf{u} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^\intercal \tag{1.24}$$

Where the control inputs are mapped by:

$$\begin{cases} u_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ u_2 = b(-\Omega_2^2 + \Omega_4^2) \\ u_3 = b(\Omega_1^2 - \Omega^3) \\ u_4 = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{cases} \tag{1.25}$$

The transformation matrix between the rate 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:

$$(\dot{\phi}, \dot{\theta}, \dot{\psi}) \approx (p, q, r) \tag{1.26}$$

Simulation tests have demonstrated that this assumption is reasonable [1].

From equations (1.21),(1.22),(1.24), the following expression is obtained after simplification:

$$f(\mathbf{x},\mathbf{u}) = \begin{pmatrix} \dot{\phi} \\ \dot{\theta}\dot{\psi}a_1 + \dot{\theta}a_2\Omega_r + b_1u_2 \\ \dot{\theta} \\ \dot{\phi}\dot{\psi}a_3 - \dot{\phi}a_4\Omega_r + b_2u_3 \\ \dot{\psi} \\ \dot{\theta}\dot{\psi}a_5 + b_3u_4 \\ \dot{z} \\ g - (\cos\phi\cos\theta)\frac{1}{m}u_1 \\ \dot{x} \\ u_x\frac{1}{m}u_1 \\ \dot{y} \\ u_y\frac{1}{m}u_1 \end{pmatrix} \tag{1.27}$$

With,

$$\begin{array}{l|l} a_1 = (I_{yy} - I_{zz})/I_{xx} & b_1 = l/I_{xx} \\ a_2 = J_r/I_{xx} & b_2 = l/I_{yy} \\ a_3 = (I_{zz} - I_{xx})/I_{yy} & b_3 = l/I_{zz} \\ a_4 = J_r/I_{yy} & \\ a_5 = (I_{xx} - I_{yy})/I_{zz} & \end{array} \tag{1.28}$$

$$\begin{aligned} u_x &= (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi) \\ u_y &= (\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi) \end{aligned} \tag{1.29}$$

It is important to note that that the angles and the derivatives of the angles do not depend on the components of the translation in the system represented by equation (1.27). Contrarily, the translation components depend of the angles. So, the system represented by equation(1.27) can be depicted as two subsystems, the angle subsystem and the translation subsystem as shown in figure

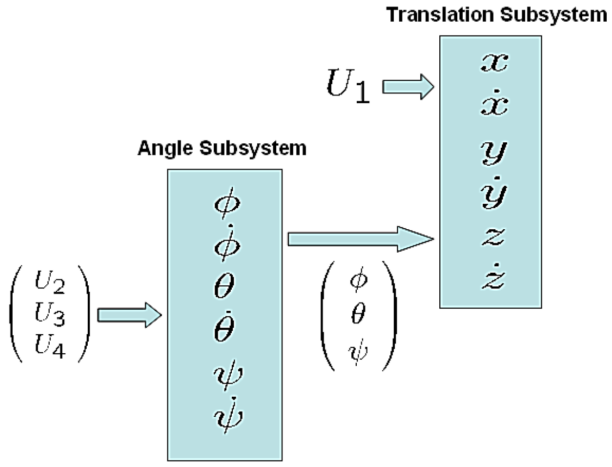


Figure 1.3: Link between the rotation and the translation subsystems.[1]

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 non-linear 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 trajecing 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. Thius 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 ithas 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 1.25. 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.

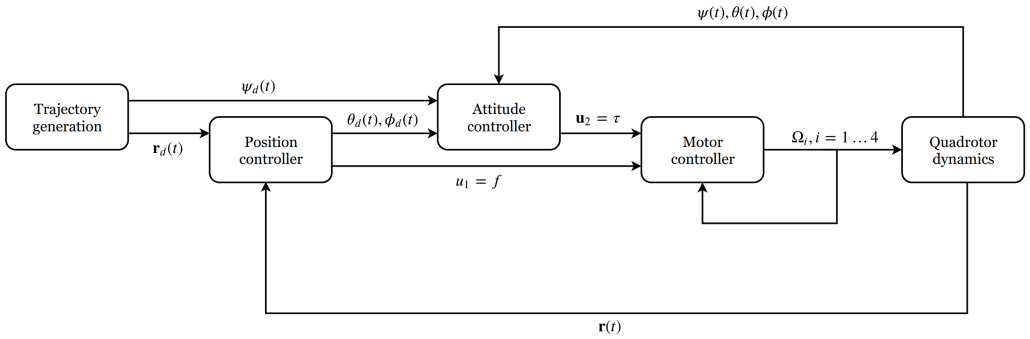


Figure 2.1: General control architecture of a quadrotor.

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 $\boldsymbol{U} = \overline{\mathbf{U}}$ 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 areound 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 notion 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 bibliography and interested readers are referred to [28] for 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 [2]:

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 aa *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. Moreover, 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 $V(x)$ 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 semidefinite* (lpsd) id 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 imaging 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.

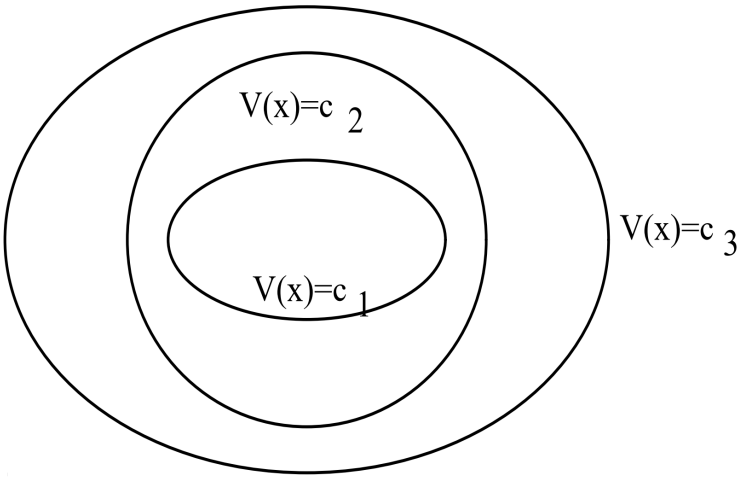


Figure 2.2: Diffrenet values for a Lyapunov function, where $c_1 < c_2 < c_3$. [2]

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 [2].

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[2].

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 [2].

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 error 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"* [3] 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 instant 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 $\mathbf{x}(k+1)$.

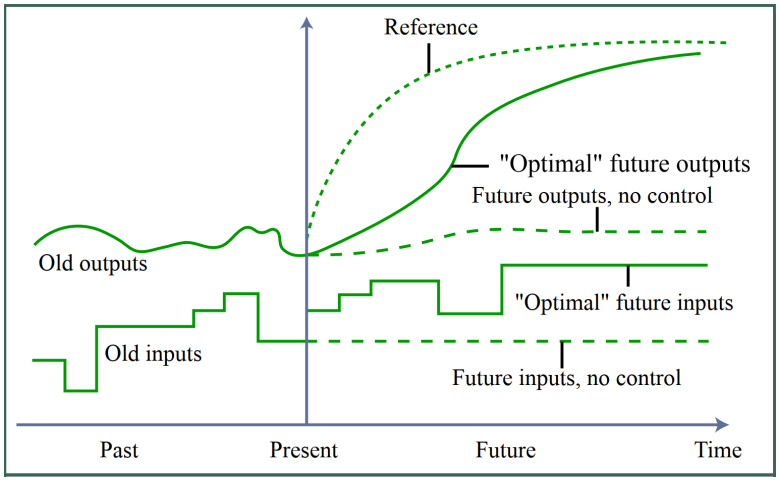


Figure 2.3: Basic idea of MPC [3].

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 instant.
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.[6]

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 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 [31].

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 [31].

Prediction horizon N The prediction horizon N (sometimes referred to with the variable p [31], 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 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 [31].

Control horizon m The control horizon is the set of future actions which will lead to the predicted plant output. It represent the number of control moves until time step m. After the first m steps, the remain 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 [31],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. If not, 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 used soft constraints on the outputs and to avoid having hard constraints on both the inputs and the rate of change of the inputs [32].

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 outs can be weighted relative to each other. It is also possible to adjust relative weights within the input rates and the outputs [33].

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 $\mathbf{z}$ or $\mathbf{u}$, then the problem falls back to the discrete LQR problem [34]. Moreover, when limits are added on $\mathbf{x}$ 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 useless if the model used for the computations is expected to be erroneous or there are distrubances applied on the system, since only the first element the optimized control sequence only will be implemented. This is why MPC is designed by using a small N.

Typical problem statement For a finite N and $F = 0$ 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 (2.14) 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 (2.14) 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 $\mathbf{x}(k)^\intercal H_1 \mathbf{x}(k)$ 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.

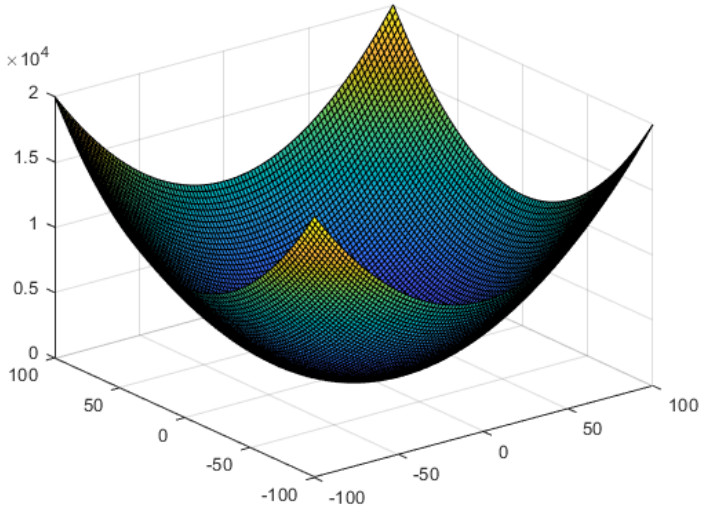


Figure 2.4: Representation of a convex function plotted using MATLAB.

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].

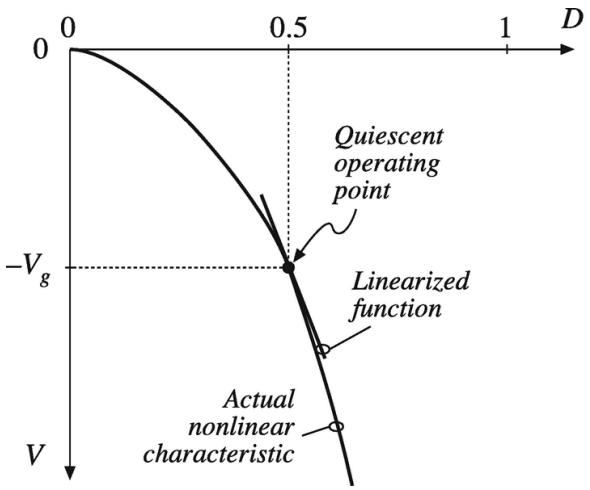


Figure 2.5: Example of the linearization of a nonlinear function around an operating point [4].

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 between all the different methods mentioned earlier, since it used 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.

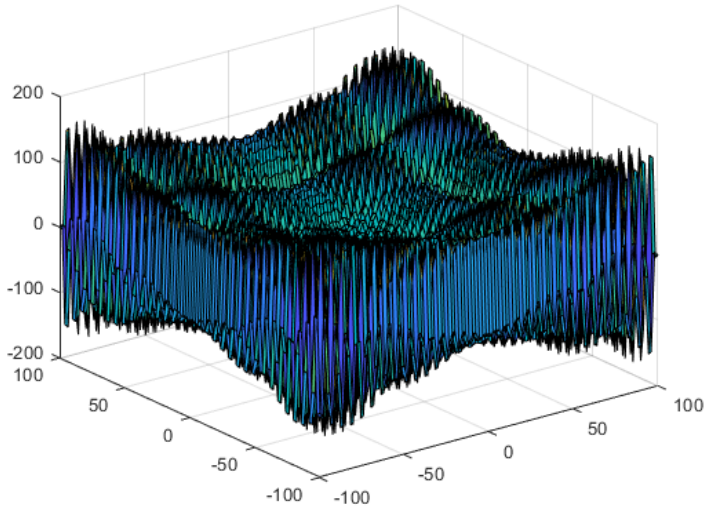


Figure 2.6: Representation of a non-convex function plotted using MATLAB.

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 important. 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 will 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.

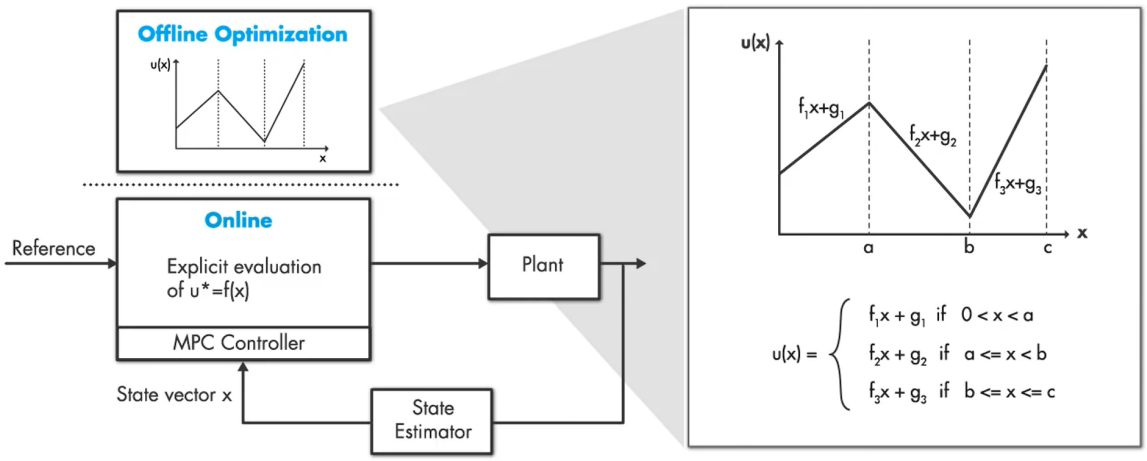


Figure 2.7: Example of an Explicit MPC controller applied on a one-dimensional system [5].

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.

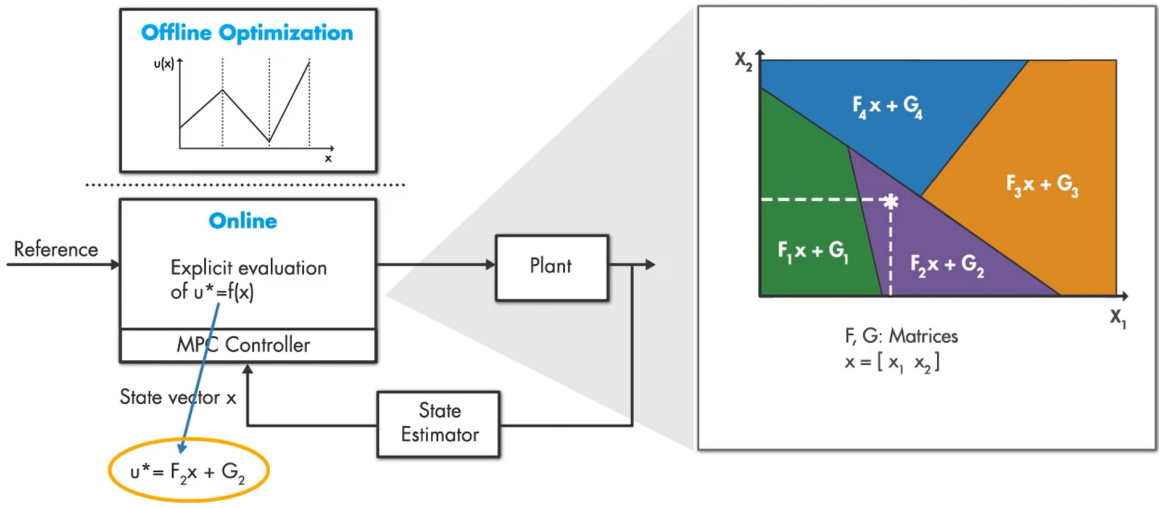


Figure 2.8: Example of an Explicit MPC controller applied on a two-dimensional system [5].

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, 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 as an example, it is taken as 5.

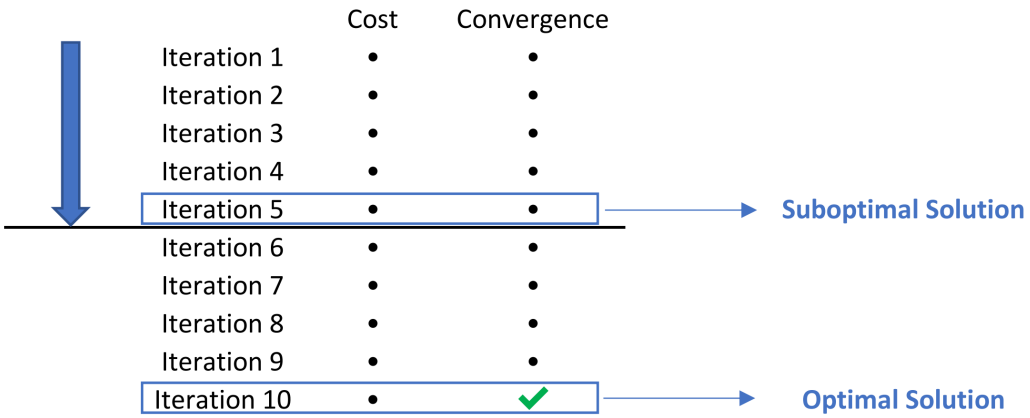


Figure 2.9: Example showing how the suboptimal solution is found when the maximum number of iterations is set to 5.

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 (MPC) which is written in Python language.
- **Control Toolbox**[4]: it is an efficient library for control, estimation, optimization and motion planning in robotics problem that is written in C++ language.

[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 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 following book presents a very modern overview of the most promising current line of theoretical and practical research in the domain.

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

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 controller 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 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 being an adequately large positive constant.

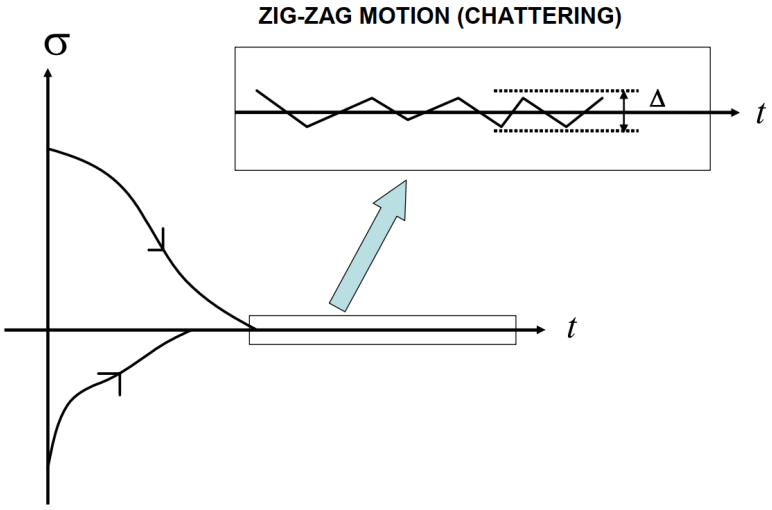


Figure 2.10: Typical evolution of σ starting from different initial conditions.

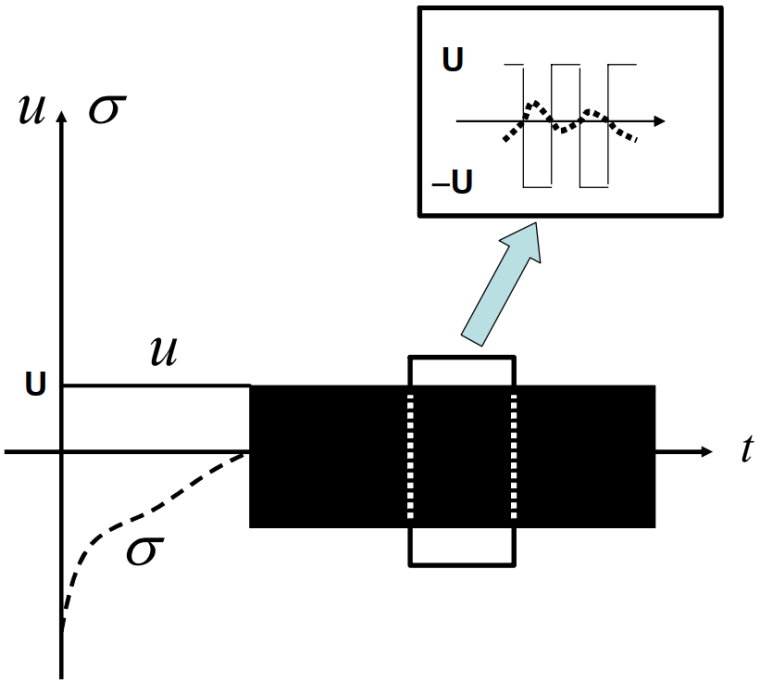


Figure 2.11: Typical evolution of the control signal u (σ is represented by dashed lines).

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" implementation (where PWM control signals are usually used) but cause 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.

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

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.

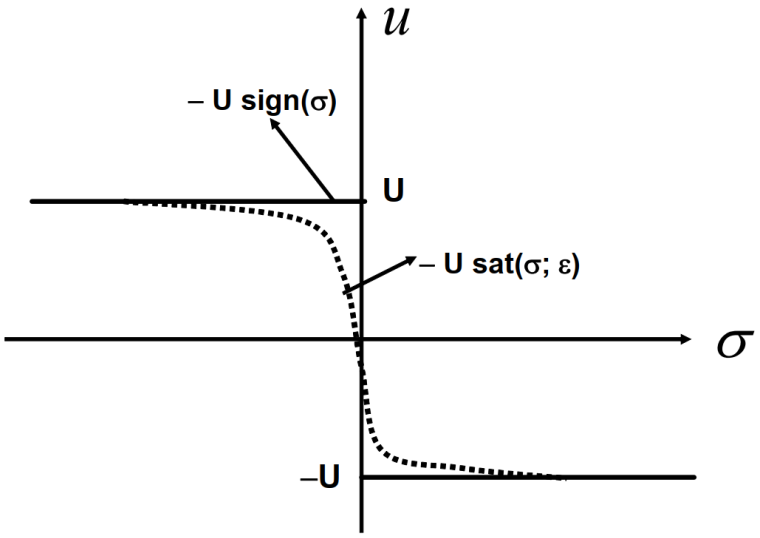


Figure 2.12: Smooth approximations of sliding mode control.

2.3.5 Other types of control

Chapter 3

Multi-flips maneuver with quadrotors

Chapter 4

Trajectory optimization

Conclusion

Bibliography

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

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

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

[4] 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

[5] 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.

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

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

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

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

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

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

[12] 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.

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

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

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

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

[17] 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.

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

[19] 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.

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

[49] 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

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