Reliable hierarchical control for multicopter systems

Ngoc Thinh Nguyen

Reliable hierarchical control for multicopter systems

Contrôle hiérarchique fiable pour les systèmes multicoptères

NGUYỄN Ngọc Thịnh

PhD Thesis 2019

Laboratoire de Conception et d'Intégration des Systèmes (LCIS)
Université Grenoble Alpes, LCIS, F-26902, France

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : **AUTOMATIQUE - PRODUCTIQUE**

Arrêté ministériel : 25 mai 2016

Présentée par

NGUYỄN Ngọc Thịnh

Thèse dirigée par **Laurent LEFÈVRE**, Professeur, Grenoble INP,
co-encadrée par **Ionela PRODAN**, Maître de Conférence, Grenoble INP

préparée au sein du **Laboratoire de Conception et d'Intégration des Systèmes**
dans **l' École Doctorale Électronique, Électrotechnique, Automatique, Traitement du Signal EEATS (EEATS)**

Contrôle hiérarchique fiable pour les systèmes multicoptères

Reliable hierarchical control for multicopter systems

Thèse soutenue publiquement le **9 décembre 2019**,
devant le jury composé de :

M. Mazen Alamir
Directeur de Recherche CNRS, Institut Polytechnique de Grenoble, Président
M. Tor Arne Johansen
Professeur, Norwegian University of Science and Technology, Rapporteur
M. Fernando Lobo Pereira
Professeur, Universidade do Porto, Rapporteur
M. Julien Marzat
Ingénieur de recherche, PhD, ONERA - The French Aerospace Lab, Examinateur
Mme. Ionela Prodan
Maître de Conf., Institut Polytechnique de Grenoble, Co-Encadrante de thèse
M. Laurent Lefèvre
Professeur, Institut Polytechnique de Grenoble, Directeur de thèse

"Now we are not afraid,
Although we know there's much to fear,
We were moving mountains,
Long before we knew we could, whoa, yes."

When you believe - a song from the 1989 animation film *The Prince of Egypt*, written and composed by Stephen Schwartz.

Acknowledgement

The song *When you believe* of Stephen Schwartz, especially the part quoted in the previous page, has inspired me since I was a child. Many times in life when facing with difficulties, it has always reminded me to never give up, to believe in myself and to consist with the chosen paths. Three years conducting this PhD thesis, were such an arduous long road where using all of my efforts and even taking some breaks with the inspirational song were not enough for accomplishing it. The success of the thesis is also due to the help, support and love which I receive from a great number of people and to whom, I have always been grateful.

First of all, I would like to thank my laboratory - LCIS (Laboratoire de Conception and d'Intégration des Systemès) and Grenoble INP for all the support of funding and working facilities during my thesis. Then, I would like to send my greatest gratitude to my supervisors, Assistant Professor Ionela Prodan and Professor Laurent Lefèvre for their guidance on research as well as their patience and continuous support related to both of professional and personal perspectives. They, being as elder sister and father, have actually created a family for me - an international student living far from home. I will never forget the first time I arrived to Valence, being exhausted after 24-hour journey and I saw Professor Laurent standing there waiting for me at 1 a.m. Also, during my intern period, Assistant Professor Ionela helped me a lot on adapting to the French life by inviting me to various special occasions: the scientific meetings, the Aerospace day as well as various archaeological and historical visits. About academic aspect, they both encouraged and supported me to explore new research challenges and to pursue them until my succeeds. Spending lots of time on organizing mobility projects, they also gave me various opportunities for visiting different laboratories in Europe where I could actually merge with people in the field and extend my knowledge.

My appreciation is further sent to the other members of the committee: Directeur de Recherche CNRS Mazen Alamir, Professor Fernando Pereira, Professor Tor Arne Johansen and Dr. Julien Marzat, for their efforts on evaluating my thesis and for lots of interesting questions. I especially thank the Reviewers, Professor Fernando Pereira and Professor Tor Arne Johansen, for their careful reading and the valuable remarks which improve significantly the quality of the exposition.

As mentioned above, during my thesis, I had the chance to visit different labs and hence, to work with a large number of people. The discussions with them brought me new ideas and helped me to open questions. Therefore, I also take this opportunity to express my gratitude to all the colleagues:

- the members of my laboratory LCIS, especially of my team MACSY-COSY as well as the technical and administration offices for their support, time availability and kindness;
- Professor Florin Stoican from "Politehnica" University of Bucharest, Romania for all of his help and valuable remarks from the beginning of my thesis;

- Professor Sorin Olaru from CentraleSupélec, France for interesting discussions at various scientific meetings;

- Professor Stefan Streif from Automatic Control and System Dynamics Laboratory, Techische Universität Chemnitz, Germany, Dr. Francesca Boem from University College London, England and Dr. Esten Ingar Grøtli from SINTEF Digital, Norway for their supports in the mobility projects and the valuable cooperation experiences.

Then, I would like to dedicate my gratefulness to my mental master, Thây Quang Viên, who taught me many important life lessons and calmed me down whenever I was stressed during the thesis. Next, my warmest thanks are sent to my friends from around the world: Youness, Mehdi, Silviu, Yoann, Igyso, Thanos, Clement, anh Lai, anh Huy, anh Hung, chi Trang, Lôc and Thiên. You all have shown me the meaning of friendship and brought to my life lots of wonderful memories which I can never ever forget.

Last but not least, despite the fact that I am living abroad, my family is forever in my heart. I am sorry for making all of you worried about me during these years, especially my grandparents and parents. I am really thankful to Mai - my little sister for being brave enough and for being healthy to take care of Mom and Dad. Finally, a special thank is also dedicated to my fiancée, Trà My, for thousand miles she flew, for her consistent trust in me and for all the care she gave during the preparation of this thesis.

Thank you.

NGUYỄN Ngọc Thịnh

Notations

Operator

Notation	Description
$\dot{x}$	time derivative of x
$A^\top$	transpose matrix of A
$\mathrm{diag}\{x, y, z\}$	diagonal matrix with the diagonal elements x, y, z
$\|x\|_P$	$x^\top P x$ for a vector $x \in \mathbb{R}^n$ and a matrix $P \in \mathbb{R}^{n\times n}$
$\|x\|$	$\left[\|x_1\| \ \|x_2\| \dots \|x_n\|\right]^\top$ for a vector $x = [x_1 \ x_2 \dots x_n]^\top \in \mathbb{R}^n$
$\mathrm{eig}(A)$	eigenvalues of the square matrix A
$\mathrm{sk}(x)$	skew-symmetric matrix in $\mathbb{R}^{n\times n}$
	satisfying $\mathrm{sk}(x)v = x \times v$ for any vectors $x, v \in \mathbb{R}^n$
$\mathrm{Re}(x)$	real part of a complex number x
$\boldsymbol{I}_n$, $\mathbf{0}_n$	identity matrix and zero matrix of size $n \times n$
$\langle x, y\rangle$	x and y. E.g., $\langle x, y\rangle \le z$ means $x \le z$ and $y \le z$.
$\mathrm{sign}(x)$	sign of a real number x.
$\max(x, y)$	maximum value between two real numbers x and y.
$\mathrm{sat}(x, x_{\max})$	standard saturation function, applied for $x = [x_1, \dots, x_n]^\top \in \mathbb{R}^n$
	and its limit $x_{\max} = [x_{\max_1}, \dots, x_{\max_n}]^\top \in \mathbb{R}^n_+$.
	$\mathrm{sat}(x, x_{\max}) = [\mathrm{sat}(x_1, x_{\max_1}), \dots, \mathrm{sat}(x_n, x_{\max_n})]^\top$
	with $\mathrm{sat}(x_i, x_{\max_i}) = \mathrm{sign}(x_i)\max(\|x_i\|, x_{\max_i})$ for all $i = 1, \dots, n$.

Number set and vector space

Notation	Description
$\mathbb{R}$	set of real numbers
$\mathbb{R}_+$	set of non-negative real numbers
$\mathbb{R}^n$	vector space of n-dimension real vectors
$\mathbb{R}^{m\times n}$	matrix space of $m \times n$ real matrices

Variables regarding a general system

Notation	Description
$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$	general system with state vector $\boldsymbol{x} \in \mathbb{R}^n$ and input vector $\boldsymbol{u} \in \mathbb{R}^m$.
$(\boldsymbol{x}_e, \boldsymbol{u}_e)$	equilibrium point satisfying $f(\boldsymbol{x}_e, \boldsymbol{u}_e) = \mathbf{0}$
$\mathcal{X}$,$\mathcal{U}$	the state and input constraint sets
$\mathbf{z}$	flat output vector in $\mathbb{R}^m$, same dimension with the input vector
$\Upsilon_{\boldsymbol{x}}$, $\Upsilon_{\boldsymbol{u}}$	flatness-based representations of $\boldsymbol{x}$ and $\boldsymbol{u}$, respectively
$\boldsymbol{u}_{\text{FL}}(\imath\boldsymbol{x}, \mu)$	feedback linearization law with $\imath \in \{0, 1\}$ the state-dependence indicator
μ	virtual control input
$\mathcal{X}_{\text{FL}}$	constraint admissible set under the feedback liearization controller
$\bar{\boldsymbol{x}}_t(s), \bar{\boldsymbol{u}}_t(s)$	predicted state and input at time instant s employed in the optimization control problem (OCP) executed at time t
$\bar{\boldsymbol{x}}_t^*(s), \bar{\boldsymbol{u}}_t^*(s)$	optimal state and input solutions for the OCP at time t
$\boldsymbol{u}_{\text{loc}}$	local controller employed for designing an NMPC controller
$\ell(\boldsymbol{x}, \boldsymbol{u})$	stage cost of an NMPC controller
$F(\boldsymbol{x})$	terminal cost of an NMPC controller

Variables regarding the multicopter system

Notation	Description
m	system mass
$J \triangleq \text{diag}\{J_x, J_y, J_z\}$	inertial tensor in $\mathbb{R}^{3\times3}$
$\boldsymbol{\mathcal{G}}$	global North-East-Altitude coordinate of the TPV system
$\boldsymbol{\mathcal{B}}$	body coordinate frame attached to the TPV system
R	rotation matrix transforming the body frame $\boldsymbol{\mathcal{B}}$ to the global frame $\boldsymbol{\mathcal{G}}$
$\xi \triangleq [x\ y\ z]^\top$	position vector in $\mathbb{R}^3$ gathering three positions (x, y, z) defined in frame $\boldsymbol{\mathcal{G}}$.
$v \triangleq [v_x\ v_y\ v_z]^\top$	velocity vector in $\mathbb{R}^3$ gathering three velocities (v_x, v_y, v_z) along the three axes.
$\eta \triangleq [\phi\ \theta\ \psi]^\top$	angle vector in $\mathbb{R}^3$ gathering the roll, pitch and yaw angles.
$\omega \triangleq [\omega_x\ \omega_y\ \omega_z]^\top$	angle rate vector in $\mathbb{R}^3$ defined in frame $\boldsymbol{\mathcal{B}}$.
T	thrust magnitude in $\mathbb{R}_+$
$\tau \triangleq [\tau_x\ \tau_y\ \tau_z]^\top$	torque vector in $\mathbb{R}^3$
$\mathsf{x} \triangleq [\xi\ v\ \eta\ \omega]^\top$	state vector in $\mathbb{R}^{12}$
$\mathsf{u} \triangleq [T\ \tau_x\ \tau_y\ \tau_z]^\top$	input vector in $\mathbb{R}^4$
$\dot{\mathsf{x}} = \mathsf{f}(\mathsf{x}, \mathsf{u})$	dynamics of the thrust-propelled vehicle system
$(\mathcal{X}_{\text{MC}}$,$\mathcal{U}_{\text{MC}})$	state and input constraint sets of the multicopter system
$\mathsf{p} \triangleq [\xi\ v]^\top$	translation state vector in $\mathbb{R}^6$
$u \triangleq [T\ \phi\ \theta]$	the translation input vector in $\mathbb{R}^3$
$\dot{\mathsf{p}} = \mathsf{f}_{\mathsf{p}}(\mathsf{p}, u, \psi)$	translation sub-system
W	transformation matrix employed in (2.1.7)
ϵ	angle boundary satisfying $\lvert\phi\rvert, \lvert\theta\rvert \leq \epsilon$ from (2.2.14)
ω_b	angle rate boundary satisfying $\lvert\omega_x\rvert, \lvert\omega_y\rvert \leq \omega_b$ from (2.2.20)

Nomenclatures

UAV	Unmanned Aerial Vehicle
VTOL	Vertical Take-Off and Landing
FL	Feedback Linearization
CTC	Computed-Torque Control
MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
FTC	Fault-Tolerant Control
FDI	Fault Detection and Isolation

Contents

List of Figures

List of Tables

Chapter 1

Introduction

UAVs (Unmanned Aerial Vehicles) [Nex and Remondino, 2014, González-Jorge et al., 2017] and *multicopters* in particular (e.g., quadcopters, hexacopters as shown in Figure 1.0.1) in particular, are already impacting our society: from military applications to assessing damage, locating victims in case of natural disasters to delivering pizzas, and more [Intwala and Parikh, 2015, Mogili and Deepak, 2018, Nascimento and Saska, 2019]. In the research community, these systems are involved in a broad range of control applications including, among others, motion planning [Mellinger and Kumar, 2011, Rucco et al., 2015], control designs [Hua et al., 2009, Rucco et al., 2016, Nguyen et al., 2018b] and fault tolerant control [Freddi et al., 2010, Saied et al., 2015, Hasan and Johansen, 2018].

(a) Quadcopter

(b) Hexacopter

Figure 1.0.1: Examples of multicopters.

The multicopter (also referred as *multirotor* as in [Klausen et al., 2017, Nascimento and Saska, 2019]) is defined as a rotorcraft with more than two rotors and named accordingly to the specific number of its rotors such as quadcopter, hexacopter or octocopter as illustrated in Figure 1.0.1. Having more than two rotors, the multicopter system offers a simple flight control mechanics, i.e.: using fixed-pitch blades and varying the rotor speeds to control their thrusts and torques [Intwala and Parikh, 2015, Nascimento and Saska, 2019] (in comparison with the complex blade pitch control mechanism applied for single- and double-rotor helicopters [Hoffmann et al., 2007] or the constraints on banking and heading angle which appear at fixed-wing UAVs [Reinhardt and Johansen, 2019]). However, note that, variable-pitch blades are still considered in some works relating to the multicopters which require the vehicles to perform significantly aggressive maneuvers [Cutler and How, 2012, Pretorius and Boje, 2014].
Even though the concept of a quadcopter system appeared long time ago with its first prototype, named *Bréguet-Richet Gyroplane No.1* built in 1907 by the brothers Louis and Jacques Bréguet under the guidance of the French scientist Charles Richet [Leishman, 2002]. Yet the multicopters have only started to gain substantial attention from the research community at the beginning of the twenty-first century when the number of related works increased rapidly year over year [Nascimento and Saska, 2019]. E.g., several works like [Altug et al., 2002, Bouabdallah et al.,

2004, Cowling et al., 2007] were published in the early 2000s and then, we observe a massive number of studies being presented up to now [Freddi et al., 2010, Chamseddine et al., 2012, Bipin et al., 2014, Rucco et al., 2016, Hasan and Johansen, 2018, Nguyen et al., 2019b]. Hence, the following question arises:

Why have the multicopters become so popular in the research community ?

The answer for this question is twofold. The first part relies on their natural properties which seem to create a perfect exhibition for numerous control applications, i.e., strongly nonlinear dynamics, under-actuated configuration [Nguyen et al., 2017b], high working frequency while being limited by the low computational power of the on-board micro controllers [Nascimento and Saska, 2019], relative ease to setup and especially their capacity to perform various autonomous tasks such as navigation [Bipin et al., 2014], path following/ trajectory tracking [Roza and Maggiore, 2012, Nguyen et al., 2018b], aerial rescue [Klausen et al., 2018] and so on. The second reason is due to the widespread growth of civil applications involving the multicopters [Giones and Brem, 2017]. The global drone market is estimated to reach 4.9 billions dollars this year, 2019, and up to 14 billions over next decade according to the Reuters newspaper[1]. Multicopters are already employed all over the world for aerial camera (e.g., photography and filming), agricultural purposes [González-Jorge et al., 2017, Mogili and Deepak, 2018], emergency response, conservation, construction planning and aerial package delivery. For example, in 2014, graduate student Alec Momont of Delft University of Technology has designed an unmanned mini aeroplane that can quickly deliver a defibrillator to where it is needed[2]. In 2017, Land Rover partnered with the Austrian Red Cross to design a special operations vehicle with a roof-mounted, thermal imaging drone which can securely land atop the vehicle while moving. This, so called, "Project Hero", hopes to save lives by speeding up response times[3]. Similarly, Zipline (one of the most prominent venture-backed medical delivery companies) has launched delivery drones in order to provide people in rural areas throughout Africa with instant access to urgent medicines[4]. Regarding the commercial transportation purposes, the octocopter of UPS shown

(a) UPS package delivery using an octocopter (Photo: https://www.ups.com/)

(b) "Prime Air" octocopter (Photo: https://www.amazon.com/)

Figure 1.0.2: Octocopters employed for aerial package delivery.

in Figure 1.0.2a is able to autonomously deliver a 4.5-kg package and come back to the delivery truck within a 30-minute fly time while the "Prime Air" octocopter of Amazon as shown in Figure 1.0.2b is anticipated to be able to lift a 25-kg package [Cheung et al., 2017]. These examples further encourage the research community to seek and open similar problems, e.g., optimal control design for a camera multicopter [Engelhardt et al., 2016], autonomous landing

[1] https://www.reuters.com/article/us-usa-security-drones/global-drone-market-estimated-to-reach-14-billion-over-next-decade-study-idUSKCN1UC2MU
[2] https://www.tudelft.nl/en/2014/tu-delft/tu-delfts-ambulance-drone-drastically-increases-chances-of-survival-of-cardiac-arrest-patients/
[3] https://www.cbinsights.com/research/drone-impact-society-uav/
[4] https://flyzipline.com/

on a moving target [Borowczyk et al., 2017], 3D (three-dimensional) structure inspection by using a camera multicopter [Eudes et al., 2018,Stoican et al., 2019], control design for multicopter with suspended load [Klausen et al., 2017] and so on.

Hence, being motivated by the expanding use of multicopters as well as the existing challenges in the research community when tackling the control of such systems, we provide in this thesis our original findings related to the hierarchical control of multicopter systems. In this sense, a literature review is provided in the following.

1.1 Hierarchical control for multicopters

As mentioned above, research and industrial communities have shown an increasing interest in multicopter aerial vehicles, e.g., tricopter, quadcopter, hexacopter since they manifest strong nonlinearities as well as being naturally coupling and having underactuated dynamics, i.e., having six degrees of freedom (to fully describe the movement of a rigid body in three-dimensional space) with only four inputs (the upward thrust and three angle torques) while their relative ease of fabrication results in low-cost experimental platforms. The systems are also subject to many operating and possibly non-convex or nonholonomic constraints [Nguyen et al., 2017b], hence, being a real challenge to control.

In the literature, the solutions can be classified into two control design perspectives, the first is to apply only one control layer which considers the whole multicopter dynamics and provides the four inputs at once [Cowling et al., 2007,Chang and Eun, 2014] and the other is to take advantage of the naturally coupling multicopter dynamics by employing a cascade control design [Hua et al., 2009, Freddi et al., 2011, Formentin and Lovera, 2011, Nguyen et al., 2017b] . The latter case is referred to as a hierarchical two-layer control scheme, shown in Figure 1.1.1 where the position controller at high level tracks the reference trajectory by providing the necessary thrust force and the desired angles while the attitude controller at low level stabilizes the vehicle's attitude at the desired set-points [Hua et al., 2009, Freddi et al., 2011]. Our works within this thesis also follow the hierarchical control direction due to several reasons presented hereinafter. Each control layer considers the translation and rotation dynamics of the multicopter system separately, hence, reducing the complexity of the control designs. Furthermore, the stability of the whole scheme is facilitated by the theory of singular perturbation, also referred as the time-scale separation principle [Zagaris et al., 2004]. The design only needs to ensure that the low control layer is exponentially stable and that the low level bandwidth is higher than the one of the high level dynamics, thus, the high level controller can be designed ignoring the low level loop [Nascimento and Saska, 2019] and furthermore, can be easily applied on commercial drone platforms since most of them already have their built-in attitude controllers running at significantly higher frequency [Intwala and Parikh, 2015, Giernacki et al., 2017].

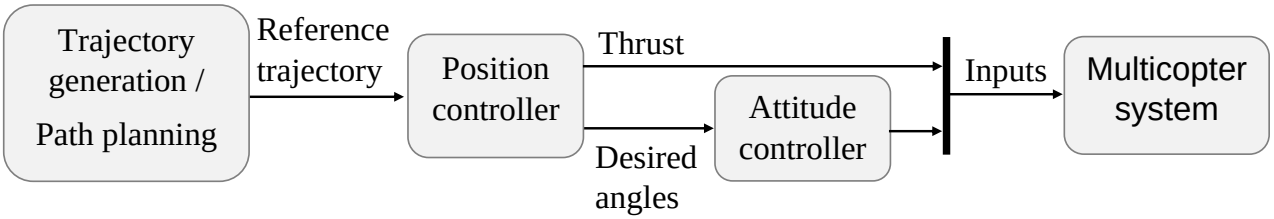


Figure 1.1.1: General hierarchical control scheme for a multicopter system.

The general control scheme for the multicopter illustrated in Figure 1.1.1 is also similar

to various control applications on different UAV systems [Prodan et al., 2013, Rucco et al., 2015] where the first step is to generate a reference trajectory (or path) which is subject to requirements defined by users (e.g., passing through way-points, pointing to targets, etc) as well as operating constraints of the systems (e.g., respecting constraints on states and inputs) awhich is subsequently tracked by the UAV by using tracking feedback controllers (e.g., the two-layer controller for the multicopter's case). Note that, many studies also consider only parts of the scheme, e.g. works like [Chamseddine et al., 2012, Eliker et al., 2018, Nguyen et al., 2018a] concentrate on the trajectory generation problems while works like [Formentin and Lovera, 2011, Cao and Lynch, 2016, Nguyen et al., 2019b] propose different tracking control designs (while assuming the reference to be already given). In the following, we present in sequence the state of the art in the two processes: trajectory generation and tracking control design for the multicopters and also for UAVs in general.

1.1.1 Reference trajectory generation

The reference trajectory generation problem is crucial for any realistic multicopter applications since these systems are challenging due to their strongly nonlinear dynamics and various constraints on states and inputs. Hence, an infeasible trajectory (in the sense of one which does not respect the internal dynamics or disregards the constraints, e.g., a step reference) causes much difficulty for the later tracking process and probably leads to poor control performance or even to instability [Chamseddine et al., 2012, Prodan et al., 2013, Klausen et al., 2017]. It is worthwhile to mention that for the multicopter case, angles are expressed as second order derivatives of position (i.e., accelerations) and torques as fourth order derivatives. Hence, a trajectory which appears to be reasonable in its position, can still be extremely challenging to track.
In the literature, such a feasible trajectory for multicopter systems is usually obtained by solving an optimization problem subject to constraints (e.g., passing through waypoints, speed/angle limitations, and the like). This process can be done either off-line or online depending on the application purposes and in general, an online algorithm takes into account less constraints than the off-line one [Hehn and D'Andrea, 2015, Rousseau et al., 2019] (further information will be provided hereinafter). We may classify existing approaches as follows:

1. **From path to trajectory**:
   The steps to follow in this approach are: generate a suitable geometric path (e.g. passing through way-points, having collision-free behavior) and parametrize it in terms of time. Then, we scale the time while checking the imposed constraints until obtaining a feasible trajectory [Hoffmann et al., 2008, Mellinger and Kumar, 2011, Bipin et al., 2015, Rousseau et al., 2019]. Both the path and the time functions need to be parametrized by using specific splines: e.g. B-splines [Piegl and Tiller, 1995] as employed in [Hoffmann et al., 2008, Bipin et al., 2015, Rousseau et al., 2019] or piecewise polynomial functions as employed in [Mellinger and Kumar, 2011]. The scaling can be done by solving an optimization problem as in [Rousseau et al., 2019] or by iterating the tuning process [Mellinger and Kumar, 2011].

2. **Trajectory parametrization with predefined time range**:
   We parametrize the trajectory by using specific splines without pre-planning a path. Then, construct an optimization problem in terms of the trajectory, in which, the internal dynamics as well as all the imposed constraints (or only part of them if some simplifications are required for real-time generation [Hehn and D'Andrea, 2015]) are taken into account. The cost is to minimize certain objectives such as, curve length by using B-spline parametrization [Stoican et al., 2017, Nguyen et al., 2018b], fuel consumption by using Laguerre and

Chebyshev polynomials [Cowling et al., 2010] or also final time instant by using a pice-wise function defined by the authors of [Hehn and D'Andrea, 2015]. Note that, the choice of the parametrization facilitates the construction of a specific cost function, hence, an inappropriate choice complicates the problem without any qualitative gain.

3. **Trajectory generation using the MPC approach**:
   This approach solves an MPC (Model Predictive Control) optimization problem for generating a point-to-point trajectory. The cost is chosen such as to minimize the distance from the initial state to the final target and the constraints consist of the internal dynamics and further operating constraints on states and inputs. If the MPC problem is infeasible to solve which may be due to the conservativeness of the constraints, the method requires the user to increase the prediction horizon (i.e., the time required for reaching the target) in order to obtain a solution. Examples for this trajectory generation method can be found in [Singh and Fuller, 2001, Mueller and D'Andrea, 2013, Engelhardt et al., 2016].

Note that depending on the complexity of the established algorithms, the process may need to be solved off-line [Stoican et al., 2015, Stoican et al., 2017, Cowling et al., 2007] or can be done on-line [Prodan et al., 2013, Mellinger and Kumar, 2011, Hehn and D'Andrea, 2011, Hehn and D'Andrea, 2015]. Some differences between the two approaches are given as follows:

- When solving an off-line trajectory generation (almost) all of the system's state and input constraints are taken into account since the restrictions on the computation time are usually less demanding [Cowling et al., 2007].

- When solving on-line the planning problem, only some constraints (e.g., passing-through way-points [Prodan et al., 2013, Mellinger and Kumar, 2011], maximum thrust force [Hehn and D'Andrea, 2011, Hehn and D'Andrea, 2015]) can be considered and after finishing the process, the rest of the constraints (e.g., rotors speed bounds) are checked. If they are not validated, some adjustments are made (e.g., reducing the maximum thrust, relaxing the time constraints) and the algorithms iteratively solve the optimization problems [Mellinger and Kumar, 2011, Hehn and D'Andrea, 2011, Hehn and D'Andrea, 2015]:

From the above summary, it can be observed that all the existing methods are required to check the constraints on states and inputs as well as the internal dynamics of the systems. This is not an easy task since the considered dynamics are strongly nonlinear and in high dimensional spaces. An appropriate tool for tackling the constrained trajectory generation problem is represented by *differential flatness* [Fliess et al., 1995]. Recently, many works related to the trajectory generation for the UAVs have made use of this concept. For example, by using flatness, [Cowling et al., 2007] generates a reference trajectory with minimum fuel consumption and [Bouktir et al., 2008, Chamseddine et al., 2012] address the minimum-time trajectory generation problems. Also, [Mellinger and Kumar, 2011] considers a trajectory passing through way-points and staying within the defined safety corridors. By further combining B-spline parametrization [Piegl and Tiller, 1995] with the flatness properties, [Prodan et al., 2013] obtains a trajectory passing through way-points with constant velocity, [Stoican et al., 2017, Nguyen et al., 2018b] can generate a trajectory with minimum-length subject to various constraints and [Rousseau et al., 2019] achieves a minimum-time trajectory with corridor constraints. Hence, we will briefly explain in the following what is *differential flatness* and why it is appropriate for works related to trajectory generation and control designs for the UAVs and multicopters systems.

Differential flatness: A generic system is called *differentially flat* if there exists a *flat output* defined in terms of states and inputs of the system such that, the states and inputs can

be expressed algebraically by using only the flat output and a finite number of its higher-order derivatives (the reader is referred to Definition 2.2.1 for a more detailed version). It is interesting that, by coincidence, the flat outputs of the general fixed-wing UAV and the multicopter systems are also their controlled outputs, i.e., the 2D position for the fixed-wing UAV [Prodan et al., 2013] as well as the 3D position and the yaw angle (direction angle) in case of the multicopter system [Chamseddine et al., 2012, Nguyen et al., 2018b] (the multicopter's case will be discussed further in Chapter 2 of the thesis). Therefore, whether the notion of flatness is explicitly mentioned or not, a great part of the existing works on the trajectory generation for the UAVs employ all or some of the flatness properties within their algorithms. For example, works like [Hehn and D'Andrea, 2011, Hehn and D'Andrea, 2015] do not specifically mention the use of flatness, but they make use of an input thrust of the quadcopter system which is given in terms of its 3D position (i.e., part of the flat output) and up to its second derivatives, hence, being exactly a flatness-based representation. We emphasize this since there exists a searching procedure introduced in [Lévine, 2011] which can find the flat output and the corresponding flatness-based representations of a generic, flat, system, hence, reducing time and improving the compactness when formulating the problems.
The widespread use of the differential flatness and its huge impact on the trajectory generation problems is due to the fact that the flat output characterizations allow us to mathematically formulate a reference path that validates specific objectives (i.e., passing through a priori given way-points, fuel consumption minimization, time-to-target minimization and state/input constraints satisfaction) [Cowling et al., 2007, Chamseddine et al., 2012, Prodan et al., 2013, Stoican et al., 2017]. One more advantage of constructing the flatness-based representations is that they can further provide us with some robustness properties (e.g., maintaining the angular constraints irrespective of the changes in the predefined yaw angle [Nguyen et al., 2018a]) and also a clear path for feedback linearization [Hagenmeyer and Delaleau, 2003]. Therefore, by exploiting the differential flatness property of the systems, the challenging tasks associated to trajectory generation and feedback linearization control design (with the added possibility of deriving robustness properties) can be alleviated.
Recalling the reference trajectory classification delineated at the beginning of this subsection, we briefly summarize next some parametrizations usually employed in the literature for the first two methods.

Polynomials parametrization: With their strengths already validated through a massive number of robotics applications during recent years [Chand and Doty, 1985, Ata, 2007], using the polynomials appears as a natural choice for solving the trajectory generation problem. The main advantage is that the polynomials provide familiar formulations with clear interpretation. Works like [Kaminer et al., 2006, Cowling et al., 2010, Mellinger and Kumar, 2011] firstly define the trajectory as standard polynomials in terms of a virtual time variable and then, parametrize the variable as another polynomial of the real time. The approach succeeds in dealing with various convoluted constraints and conditions such as passing through way-points, collision-free and changing time coordinates for multiple UAVs [Kaminer et al., 2006] as well as staying within safety corridors [Mellinger and Kumar, 2011]. Furthermore, [Cowling et al., 2010] presents a thorough comparison between using Chebyshev, Laguerre functions [Fahroo and Ross, 2002, Valencia-Palomo and Rossiter, 2010] and the polynomial parametrization within the trajectory generation problem for a quadcopter system in which, the latter appears as the best candidate. However, a draw back of the polynomials is that their degree increases accordingly to the number of constraints [Cowling et al., 2010] and the coefficients (i.e., the tuning parameters) do not provide any direct and illustrative meaning. Therefore, for more recent applications, another parametrization with more advantages (namely, B-spline functions) is replacing the classical

polynomials.

B-splines parametrization: B-splines [Piegl and Tiller, 1995] are involved in plenty of studies on the trajectory generation for UAVs and multicopters [Hoffmann et al., 2008, Prodan et al., 2013, Bipin et al., 2014, Nguyen et al., 2018b, Rousseau et al., 2019]. A B-spline curve is parameterized by a set of control points and a knot vector which defines the moments when the curve switches between two polynomial basis functions. The two most interesting properties of the B-spline curve are given as follows:

1) the curve always lies within the convex hull of the control points;

2) a derivative of the B-spline function can be expressed in terms of a linear combination of the basis functions [Suryawan et al., 2011, Mercy et al., 2017].

The first property inspires various geometrical applications on the UAVs and multicopters, e.g., [Stoican et al., 2017] and [Rousseau et al., 2019] generate the reference trajectories constrained to pass through (or near) predefined way-points and to stay within a polytope and a cylindrical flight corridor, respectively. Next, the linear derivative relation helps to greatly reduce the complexity when formulating the constraints or the cost functions which employ the high-order derivatives of the trajectory. For example, [Stoican et al., 2015] and [Nguyen et al., 2018b] consider the problem of minimizing the lengths of the reference trajectories for the UAV and the multicopter, respectively, in which, the cost functions given in terms of the velocity are transformed into a quadratic function of the control points, and hence, become easy to solve for its minimum value. It is worth mentioning that most of the works employ a uniform B-spline which requires the knot vector (i.e., defining the switching moments between different basis functions) to be equally distributed [Hoffmann et al., 2008, Prodan et al., 2013, Stoican et al., 2015, Stoican et al., 2017] and this causes difficulty when considering the minimum-time trajectory generation problem. Therefore, by considering non-uniform B-spline curves, the works in [Bipin et al., 2015, Rousseau et al., 2019] can mitigate the issue and succeed in minimizing the time for reaching their targets.

MPC (Model Predictive Control): MPC (or also referred as receding horizon control) is a control strategy in which, an optimization problem, subject to system dynamics, state and input constraints and initial condition, is solved at each time step to obtain an optimal control sequence for a finite number of future steps (the so called prediction horizon). From the sequence, the first input value is applied to the system. At the next time step, the newly obtained state is introduced into the optimization problem as the new initial condition. The process is repeated to create the closed-loop controlled system [Alamir and Bornard, 1994, Mayne et al., 2000, Grüne and Pannek, 2011]. The strategy is well-known for its capability of handling explictly constraints while having a structural design. The cost function of the optimization problem is usually designed to minimize the tracking error (e.g. being a quadratic form in terms of the error) for the standard tracking control design or even to maximize the economic benefits for the, so called, economic MPC controller [Rawlings et al., 2012]. Due to its implicit control law resulted from solving the optimization problem, typical issues on designing the MPC scheme are closed-loop stability, feasibility of the solution and also computation burden [Mayne et al., 2000].
In order to apply the MPC method within the trajectory generation problem, besides the initial condition, we also constrain the final predicted state to arrive to the predefined ending point. The optimization problem is solved once and the optimal state sequence is used as the reference trajectory. The method requires to tune the prediction horizon such that the problem is feasible. Note that the approach also allows to reconfigure the trajectory by simply introducing a new initial condition to the MPC scheme [Hehn and D'Andrea, 2015].

Next, we summarize the existing approaches on designing the tracking controllers for the multicopters. The main sources of the study are the comprehensive bibliography reviews [Hua et al., 2013] and [Nascimento and Saska, 2019].

1.1.2 Tracking control architectures

In Figure 1.1.1, we do not explicitly define the outputs of the attitude controller nor the inputs of the multicopter system since they are very diverse for various applications. E.g., some platforms receive thrust and angles as their inputs as in [Giernacki et al., 2017, Nguyen et al., 2018b] while others can admit thrust and angle rates as used in [Hehn and D'Andrea, 2015]. Therefore, we classify existing works on the multicopter control by the inputs admitted by the employed platforms. The classification is summarized in Figure 1.1.2 which should be read from up to down. We emphasize that accepting the inputs of the platforms also implies a certain

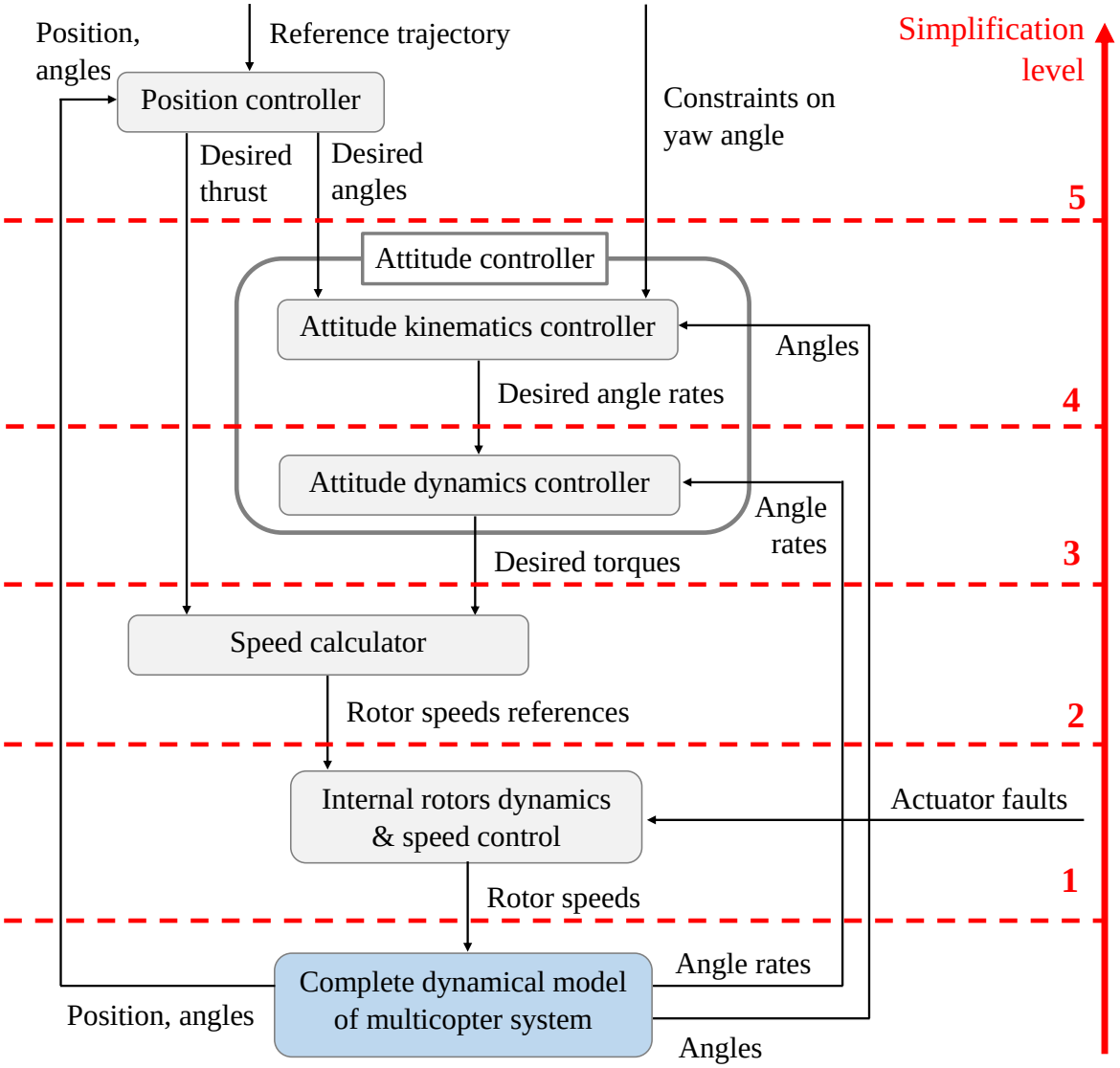


Figure 1.1.2: Classification of tracking control designs for the multicopters.

simplification level w.r.t. the full dynamical model of the multicopter represented by the blue

box at the end of Figure 1.1.2. For example, at the simplification level 1 corresponding to the less simplified multicopter dynamics employed in the control scheme, the inputs are defined as its actual rotor speeds which are different from the rotor speeds references resulted from the whole control scheme [Jiang et al., 2016, Borisov et al., 2017]. The internal rotor dynamics and even actuator faults (if applicable) act as the uncertainties affecting the system. In contrast, at the highest simplification level 5, the employed platforms receives the desired values of thrust and angles as its inputs [Giernacki et al., 2017, Nguyen et al., 2018b, Eudes et al., 2018]. (i.e., the platforms already have their built-in controllers which control the rotors to stabilize the platform at these desired inputs), and hence, works classified at this level consider only the translation dynamics of the multicopter system while neglecting the rest of the scheme. More precisely, the existing works are classified as follows:

- **Level 1:** As mentioned above, the works at this level distinguish the actual rotor speeds and their references provided by the designed controller. Hence, they can take into account the rotor dynamics (e.g., can be explicitly given as in [Bouabdallah et al., 2004] but mostly being simplified as saturation effects [Jiang et al., 2016, Borisov et al., 2017] or low-pass filters [Shao et al., 2018]) and actuator faults such as loss of efficiency [Chamseddine et al., 2012, Avram et al., 2017], complete loss of rotor(s) [Mueller and D'Andrea, 2014, Falconí et al., 2016, Sun et al., 2018] and stuck rotor faults [Freddi et al., 2011, Nguyen et al., 2017b]. Hence, this level includes the works on FTC (Fault Tolerant Control) designs and/or on the designs of the related fault diagnosis modules [Hasan and Johansen, 2018]. For the control design part, a two-layer hierarchical control scheme is usually employed. The *position controller* at the high level provides the desired thrust and the desired angles for the controller at the low level, called *attitude controller*, to track. Then, the attitude controller gives the desired torques which are combined with the desired thrust in order to calculate the rotating speed references for the rotors. Note that, as aforementioned, the actual rotor speeds (i.e., the inputs of the multicopter system) track these references but are subject to the rotor dynamics and faults.

- **Level 2:** The multicopter system at this level admits its inputs as the rotor speeds under actuator saturation constraints only (i.e. still under the simplified rotor dynamics but not subject to fault(s)) [Chen and Huzmezan, 2003, Monteiro et al., 2016, Convens et al., 2017, Wang et al., 2017]. For guaranteeing these constraints, it is necessary to design the aforementioned two-layer hierarchical control scheme subject to polytopic constraints on the desired thrust and torques due to the use of the *control allocation* (i.e., the linear relation between the desired thrust, torques and the rotor speed references). A natural solution is to employ the (Nonlinear) MPC controllers as proposed in [Monteiro et al., 2016, Wang et al., 2017]. Other methods can be considered as $\mathcal{H} - \infty$ as proposed [Chen and Huzmezan, 2003] or a sophisticated Lyapunov control design as given in [Convens et al., 2017].

- **Level 3:** This level contains most of the existing works related to the control designs of the multicopters. At this level, the inputs of the system are considered as either the rotor speeds references without constraints (which are equivalent to the desired thrust, torques without constraints due to their linear relation described by the *control allocation*) as in [Hua et al., 2009, Formentin and Lovera, 2011, Mellinger and Kumar, 2011, Roza and Maggiore, 2012, Klausen et al., 2014, Nguyen et al., 2017b] or with more difficulty, the desired thrust and torques subject to their saturation constraints [Limaverde Filho et al., 2016, Lu et al., 2017, Nguyen et al., 2017a]. The first class represents for the nominal designs of the aforementioned hierarchical control scheme where the main goal is to establish

the stability of the whole scheme by designing the position and attitude controllers while no input constraints are taken into account. This freedom facilitates various sophisticated control applications such as Lyapunov-based design [Hua et al., 2009, Mellinger and Kumar, 2011], flatness-based control [Formentin and Lovera, 2011, Nguyen et al., 2017b], output feedback linearization control [Roza and Maggiore, 2012], Model Predictive Control [Engelhardt et al., 2016, Zanelli et al., 2018] and so on. For the second class, in which the desired thrust and torques are subject to the saturation constraints, the related works usually apply a hierarchical optimization-based controller for counteracting these problems. E.g., [Lu et al., 2017] designs an MPC scheme at the high control level and an $\mathcal{H} - \infty$ controller with a disturbance observer at the low control level. In a similar fashion, [Nguyen et al., 2017a] employs an MPC position controller for guaranteeing the constraint on thrust and next, uses a combination of computed-torque control [Craig, 2005] and MPC for ensuring the torque constraints.

- **Level 4:** In this category, the multicopters are considered to admit the desired thrust and three angle rates as their input [Hehn and D'Andrea, 2011, Mueller and D'Andrea, 2013, Ha et al., 2014, Hehn and D'Andrea, 2015]. Therefore, at this level, besides the translation dynamics, only the attitude kinematics (i.e., describing the dynamical relation between the attitude and the angle rates) are considered and as a result, it is more convenient to control both of them at once. For example, works from the group of D'Andrea Raffaello [Hehn and D'Andrea, 2011, Mueller and D'Andrea, 2013, Hehn and D'Andrea, 2015] solve one optimization problem (being similar to a MPC approach [Mayne et al., 2000]) subject to constraints on states and inputs of the system (considered decoupled, for simplification) in real time. The first optimal input values are applied to the multicopter and then, the algorithm re-plans the entire trajectory from the present state. Similarly, [Ha et al., 2014] constructs only one passivity-based adaptive backstepping controller for tackling the trajectory tracking problem of the multicopter dynamics described by this simplification level.

- **Level 5:** There exist some multicopter platforms which already have their built-in controller to stabilize the system at the desired values of thrust and angles [Giernacki et al., 2017, Nguyen et al., 2018b, Eudes et al., 2018]. Therefore, using these platforms allows the researcher to concentrate only on developing the position controller [Roza and Maggiore, 2014, Nguyen et al., 2018b, Nguyen et al., 2019b]. Note that, even though we are classifying this category at the highest simplification level of the dynamics, it does not imply that the position control designs of these works are simpler than those of the other levels, but probably vice versa. E.g., [Roza and Maggiore, 2014] proposes the general principles for designing the position controller with almost global stability guarantee, then, apply the theorem on three different control designs. [Nguyen et al., 2018b] and [Nguyen et al., 2019b] follow the aforementioned theorem and construct a flatness-based feedback linearization controller and an NMPC (Nonlinear MPC) controller with guaranteed stability, respectively.

By this summary, we have classified the works related to the multicopter control designs in the literature based on the complexity of the employed dynamical models (see again Figure 1.1.2). Their corresponding main difficulties as well as the typically employed solutions are also discussed. In the following, we provide some recent updates on designing the position and the attitude controllers.

Position control designs:

The position control problem of the multicopters has been studied with vigour in the literature recently [Nascimento and Saska, 2019]. The position controller is responsible for controlling the outer loop which provides the desired thrust and the desired angles for the low control level to follow. The goals of the control designs can be for trajectory tracking [Hoffmann et al., 2008, Nguyen et al., 2018b], for target tracking (i.e., stabilizing or hovering at a desired point) [Choi and Ahn, 2014, Kuantama et al., 2018, Nguyen et al., 2019b] and even only for altitude tracking [Muñoz et al., 2017]. Over two hundreds different works are summarized and classified in the bibliography review [Nascimento and Saska, 2019] which show a diversity of the employed control methods: adaptive control [Borisov et al., 2017], backstepping control [Ha et al., 2014], differential flatness-based [Formentin and Lovera, 2011, Nguyen et al., 2017b], Lyapunov-based control [Hua et al., 2009, Convens et al., 2017], NMPC [Merabti et al., 2015, Nguyen et al., 2019b] and so on. Among them, we find out interest in the method of feedback linearization control based on differential flatness [Hagenmeyer and Delaleau, 2003] which not only benefits from the existing flatness representation of the multicopter system (c.f. the use of flatness on the trajectory generation as detailed in Section 1.1.1) but also provides several robustness properties on the roll, pitch angles under undefined yaw angle as detailed in [Nguyen et al., 2018b]. Theses properties are useful when considering actuator faults (c.f. Level 1 of the classification given at the beginning of this section) which usually lead to uncontrolled yaw motion [Freddi et al., 2011, Nguyen et al., 2017b]. Especially, this control design facilitates the construction of an invariant set which can be employed to guarantee the stability of an NMPC controller as introduced in [Nguyen et al., 2019b].

Attitude control designs:

Recalling the classification given in Figure 1.1.2, we consider that the attitude control problem is to stabilize the whole rotation dynamics at the desired angles with the inputs defined as the desired torques. Our definition is different from the one of the bibliography review [Nascimento and Saska, 2019] where the authors define the attitude control problem as only the attitude kinematics controller shown in Figure 1.1.2. To the best of our knowledge, this covers most of the works related to attitude control in the literature while the works considering the angle rates as the multicopter inputs (i.e., belonging to the simplification level 4 of our classification given in Figure 1.1.2) are in minority.
The attitude control designs gained a substantial interest in the research community in the last 6 years [Nascimento and Saska, 2019] and various control methods have been applied in the literature: PID [Bolandi et al., 2013, Yu et al., 2015], dynamics inversion control (e.g., computed-torque control) [Nguyen et al., 2017b, Das et al., 2009], Lyapunov-based control [Mellinger and Kumar, 2011], quaternion-based control [Do, 2015, Liu et al., 2015], (Nonlinear) MPC [Alexis et al., 2014, Nguyen et al., 2017a] and so on. Beside the common goal of stabilizing the attitude, there also exist several interesting directions as follows:

- robust control designs with disturbance rejection. E.g.: [Lu et al., 2017] combines disturbance observer-based control and $\mathcal{H} - \infty$ control in order to construct a composite hierarchical anti-disturbance controller. Also, [Lotufo et al., 2019] makes use of active disturbance rejection control and embedded model control to design its robust attitude controller against multiple disturbance types.

- FTC designs for counteracting specific actuator faults (i.e., employed for the works at the simplification level 1 of our classification): loss of efficiency [Chamseddine et al., 2012,

Avram et al., 2017], complete loss of rotor(s) [Mueller and D'Andrea, 2014, Falconí et al., 2016, Sun et al., 2018] and stuck rotor faults [Freddi et al., 2011, Nguyen et al., 2017b].

The foregoing literature review demonstrates the variety and vastness of the existing approaches on trajectory planning and control designs for the UAVs and the multicopters in particular. It also proves the tremendous interest in drone applications (e.g., applied control designs, mapping, navigation, aerial photography), shared by both research and industrial communities. However, we find out that there are still several open problems in the literature which, if being solved, will strongly improve and expand the use of the multicopters. These questions will be addressed in the following section.

1.1.3 Open problems

Problem 1. [Constraints validation under change of the predefined direction angle] Usually, a reference trajectory for a multicopter system is generated with a desired direction (yaw) angle [Cowling et al., 2007, Lu et al., 2017, Abadi et al., 2019]. However, changing of the predefined direction angle during flight is necessary and unavoidable for some realistic and common applications such as aerial photography [Engelhardt et al., 2016] or consideration of faults [Chen and Jiang, 2005, Freddi et al., 2011]. Therefore, the a priori computed flight path may become suboptimal or even infeasible and, hence, impossible to follow/track. Consequently, it is worth investigating on effective states/inputs constraints for the multicopter system which do not depend on a predefined yaw trajectory, and hence, can be satisfied even under change of the direction angle during flight.

Problem 2. [NMPC design with stability guarantee and fast response requirement] Let us consider various existing NMPC designs for a multicopter system [Zanelli et al., 2018, Limaverde Filho et al., 2016, Mueller and D'Andrea, 2013] and also for a fixed-wing UAV system [Prodan et al., 2013, Gros et al., 2012]. It is well-known that the common problems for an optimization-based controller are its heavy computational burden and the difficulties in guaranteeing the feasibility of the solution as well as the closed-loop stability (even for nominal consideration) due to its implicit control law [Alamir and Bornard, 1994, Mayne et al., 2000, Alamir and Murilo, 2008]. These issues are even more critical for aerial vehicles since they requires not only rapid but also accurate responses. The foregoing works on NMPC designs for UAVs concentrate only on reducing the computation time but disregard the stability and the feasibility problems, hence, possibly yielding an infeasible solution or even leading to instability. Thus, designing an NMPC scheme which not only requires fast computation time but also guarantees the closed-loop stability as well as the feasibility still remains an open and valuable question.

Problem 3. [NMPC design with semi-global stability guarantee] C.f. Problem 2, there exists a region of attraction for any NMPC design with guaranteed stability [Alamir and Bornard, 1994, Mayne et al., 2000, Grüne and Pannek, 2011]. It is also trivial that increasing the prediction horizon length of the NMPC controller will enlarge this region. However, the prediction horizon has its upper limit due to the effect it has on computation time, hence, limiting the region of attraction around the desired equilibrium. Thus, an NMPC design with guaranteed stability whose region of attraction is easy to tune such that it covers a compact set containing feasible initial states of the system even with a fixed value of the prediction horizon (so called semi-global stability) is new to the state of the art.

Problem 4. [Functioning under stuck actuator fault and constraints] In the literature, an unique stuck rotor fault is considered for the spacecrafts [Yang and Lum, 2003, Marzat et al.,

2012, Jiang et al., 2016] and for the quadcopter system [Freddi et al., 2011, Nguyen et al., 2017b]. Specially, for more information, once being stuck, the faulty actuators keep rotating at a constant speed regardless of the variation of the control inputs. Thus, under a unique stuck rotor fault, the quadcopter system not only loses one degree of freedom in its control ability but also gains persistent external disturbances [Chen and Jiang, 2005]. The aforementioned existing works all assume that the fault diagnosis module is already installed to detect, isolate, and identify the fault parameters, i.e., the stuck rotor and its associated speed, and then, propose the control reconfiguration designs and several related analysis. However, due to the complexity of the faulty system, these works do not consider any actuator constraints. This may lead to harmful saturation effects under real applications and as a result, the desired control properties (e.g., stability, performance) cannot be held. Therefore, improving the works on counteracting the stuck actuator fault for the multicopter as in [Freddi et al., 2011, Nguyen et al., 2017b] is also an interesting question.

1.2 Thesis orientation

The bibliography review detailed in Section 1.1 highlights the fact that the motion planning problem for a multicopter system shows a great deal of variety in the employed dynamical models, control architectures and design methods. Thus, in this thesis, we limit ourselves to the most general control architecture for a standard multicopter system which is illustrated at the simplification level 3 in Figure 1.1.2. The scheme consists of a trajectory planner, a two-layer hierarchical control scheme in consideration of the constraints on states and inputs (i.e., thrust and three angle torques). Bear in mind that without specifying the multicopter's type, we do not have the rotor configuration (i.e., number and arrangement), hence, we cannot obtain the rotor speeds (i.e., required for applying the simplification levels 1 and 2 in Figure 1.1.2).

The control approach considered within the thesis is to generate off-line a feasible reference trajectory with respect to the nominal functioning of the multicopter system and then, to design the tracking hierarchical controller. Regarding the trajectory generation, the reference has to pass through way-points, to satisfy the system constraints and to take into account that the yaw motion can be modified during flight due to intentional change for aerial camera/filming applications or due to faults. We exploit the differential flatness properties of the multicopter system for seeking a solution to Problem 1, i.e., maintaining the constraints satisfactions (i.e., the feasibility w.r.t. to the system) of the reference trajectory under change of the predefined yaw angle.

Next, for the tracking control design, we exploit the properties of a feedback linearization controller via flatness at the high level and a computed-torque controller at the low level. We first deal with the trajectory tracking problem under these feedback linearization controllers. Then, in order to fulfill various state and input constraints of the nonlinear multicopter dynamics, NMPC appears as a suitable candidate. Noticing Problems 2 and 3, we also analyze the stability, feasibility and computation burden problems affecting the NMPC controller.

Furthermore, we concentrate on a particular type of multicopter, the quadcopter, to propose solutions for Problem 4 which addresses questions on a reliable control design for such systems under a stuck actuator fault.

1.3 Thesis contributions

This thesis addresses the multicopter hovering, trajectory generation and tracking problems through a coherent hierarchical control framework exploiting the properties of differential flatness, feedback linearization, and in particular computed-torque control, and NMPC design with stability guarantees.
More precisely, the trajectory generation problem is solved by combining the differential flatness, B-spline parametrization and the robust angular constraint formulations proposed in [Nguyen et al., 2018a]. The resulted trajectory achieves the minimum length curve and validates the system constraints even under mismatches on the desired direction angle tracking. The approach is validated over a generic indoor trajectory generation application [Nguyen et al., 2018b] and a building inspection scenario where the vehicle's direction is varying over time for pointing the mounted camera towards the building [Stoican and Prodan and Grøtli and Nguyen, 2019]. Then, the flatness representation of the system and the angular constraint formulations designed in [Nguyen et al., 2018a] further facilitate a feedback linearization (FL) design for the position controller at the high control level with saturation guarantees. An unstable mode of this controller is also addressed which can lead to instability in case of large altitude error even under nominal functioning. The controller is validated through simulation and experimental tests over the Crazyflie 2.0 quadcopter platform [Nguyen et al., 2018b]. It is worth mentioning that our method is tested independently through numerical experiments in the Gazebo software by [Silano et al., 2019], which highlight its effectiveness.

Furthermore, a hierarchical optimization-based control scheme is first introduced in [Nguyen et al., 2017a] (where linear MPC-based controllers are employed at both the two control layers) and is significantly developed in the thesis by using different NMPC (Nonlinear Model Predictive Control) designs with guaranteed stability. More precisely, for the position controller at high level we employ:

1. NMPC design with terminal constraints: The design is an extension of our work in [Nguyen et al., 2019b] which includes an invariant set under the aforementioned FL controller and standard qudratic stage and terminal costs. The tuning parameters are chosen according to the existing NMPC design principles in order to establish the (locally) asympotic stability of the closed-loop system. Then, by exploiting the special construction of the admissible set, we are able to enlarge infinitely the terminal region (if not restricted otherwise by state constraints), hence, also increasing the domain of attraction unlimitedly and finally, achieving the semi-global stability property of the design.

2. NMPC design using a terminal relaxed invariant set [Nguyen et al., TSMC]: The design is constructed similarly to the previous one but using a δ-invariant set (with δ the sampling time of the NMPC controller) as its terminal region. The set allows the state to escape from itself but guarantees that it will come back at predefined periodic time instants and always stay within the admissible set. Due to this relaxation, the terminal region can be obtained through simple box-type constraints. This reduces the complexity of the NMPC optimization problem (i.e., less computation burden) and ensures the closed-loop stability.

3. NMPC design without terminal constraints [Nguyen et al., 2020c]: This design ensures the closed-loop stability by defining an appropriate prediction horizon length corresponding to a specific domain of attraction. A thorough analysis is conducted by applying the existing design procedure to the FL controller which provides the shortest required length as 150 steps in comparison with thousands steps obtained when using a standard linear controller.

Next, for the attitude controller at the low control level, we present a general construction of an admissible invariant set under the classical CTC (Computed-Torque Control) law of a "computed-torque like" system and employ it as the terminal region within an NMPC design for guaranteeing the closed-loop stability. Next, the approach is applied to design the NMPC attitude controller with further improvements on the parameter calculation process. All the proposed approaches are validated through extensive simulations and comparison with the existing designs in the literature while the NMPC position controllers are also experimentally tested over the Crazyflie quadcopter platform.

Furthermore, a reliable hierarchical control scheme is designed for the case of a quadcotper under a unique stuck rotor fault. We extend our work in [Nguyen et al., 2017b] by developing an under-fault functioning mode for the aforementioned NMPC attitude controller and the corresponding fault diagnosis module which allows to analyze the faulty quadcopter system in a similar fashion to the nominal case, hence, still guaranteeing the tracking capability. The proposed FTC (Fault Tolerant Control) scheme shows its effectiveness through various simulations.

Finally, this thesis can be placed in a line of research which includes the (relatively recent) results in [Formentin and Lovera, 2011, Freddi et al., 2011, Prodan et al., 2013, Hehn and D'Andrea, 2015, Engelhardt et al., 2016, Zanelli et al., 2018]. With respect to their results, we significantly improve the control methods despite the fact that they can share similar tools. More precisely, the FL controller (i.e., obtained via flatness as aforementioned) is also applied in [Formentin and Lovera, 2011, Freddi et al., 2011] but these works have some major drawbacks: no input constraints are employed and furthermore, the authors do not notice the unstable mode which is discovered in [Nguyen et al., 2018b]. Therefore, under realistic applications with essential actuator limits, their designs will obviously lose the feedback linearization properties and probably fail into instability. Furthermore, [Engelhardt et al., 2016, Zanelli et al., 2018] also designs NMPC controllers for the multicopter system but these works do not treat thoroughly the stability issues. Unlike those, we propose three different original approaches to stabilize the NMPC designs for the multicopter which are validated in simulation and experiment.

In the sequel, we provide both the list of publications accepted/submitted to various conferences and journals, and the presentations given by the candidate at various scientific days or within various mobility projects of the co-supervisor.

Journal articles:

[1] **N.T. Nguyen**, I. Prodan, L. Lefèvre, "Stability guarantees for translational thrust-propelled vehicles dynamics through NMPC designs". *IEEE Transactions on Control Systems Technology*, 2020. Article in Press, pp. 1–13. DOI: https://doi.org/10.1109/TCST.2020.2974146.
[2] **N.T. Nguyen**, I. Prodan, L. Lefèvre, "Flat trajectory design and tracking with saturation guarantees: a nano-drone application". *International Journal of Control*, 2018. Article in Press, pp. 1–14. DOI: https://doi.org/10.1080/00207179.2018.1502474.
[3] **N.T. Nguyen**, I. Prodan, L. Lefèvre, "Stabilising VTOL system through an NMPC design with a relaxed terminal region". Submitted to the *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019.
[4] **N.T. Nguyen**, I. Prodan, L. Lefèvre, "NMPC design with semi-global stability guarantee for thrust-propelled vehicles dynamics". Submitted to the *IEEE Control Systems Letters*, 2020.

Conference papers:

[1] **N.T. Nguyen**, I. Prodan, L. Lefèvre, "Multicopter attitude control through NMPC design with guaranteed stability". Accepted for publication at the 2020 IFAC World Congress, held in

Berlin, July, 2020.
[2] **N.T. Nguyen**, I. Prodan, F. Petzke, S. Streif, L. Lefèvre, "Hierarchical control of a quadcopter under stuck actuator fault". Accepted for publication at the 2020 IFAC World Congress.
[3] H.T. Nguyen, **N.T. Nguyen**, I. Prodan, "Trajectory tracking for a quadcopter under a quaternion representation". Accepted for publication at the 2020 IFAC World Congress.
[4] **N.T. Nguyen**, I. Prodan, L. Lefèvre, "On the use of a computed-torque control law for the terminal region of an NMPC scheme", in Proc. of the *2019 American Control Conference (ACC'19)*, pp. 1008–1013, held in Philadelphia, USA, 2019.
[5] **N.T. Nguyen**, I. Prodan, L. Lefèvre, "A stabilizing NMPC design for thrust-propelled vehicles dynamics via feedback linearization", in Proc. of the *2019 American Control Conference (ACC'19)*, pp. 2909–2914, held in Philadelphia, USA, 2019.
[6] F. Stoican, I. Prodan, E.I. Grøtli, **N.T. Nguyen**, "Inspection Trajectory Planning for 3D Structures under a Mixed-Integer Framework", in Proc. of the *2019 IEEE International Conference on Control & Automation (ICCA'19)*, pp. 1349–1354, held in Edinburgh, Scotland, 2019.
[7] **N.T. Nguyen**, I. Prodan, L. Lefèvre, "NMPC design with invariance induced by a computed-torque control law". Abstract publication at the *6th IFAC Conference on Nonlinear Model Predictive Control (NMPC'18)*, held in Madison, Wisconsin, USA, 2018.
[8] **N.T. Nguyen**, I. Prodan, L. Lefèvre, "Effective angular constrained trajectory generation for thrust-propelled vehicles", in Proc. of the *2018 European Control Conference (ECC'18)*, pp. 1833–1838, held in Limassol, Cyprus, 2018.
[9] **N.T. Nguyen**, I. Prodan, F. Stoican, L. Lefèvre, "Reliable nonlinear control for quadcopter trajectory tracking through differential flatness", in Proc. of the *20th IFAC World Congress*, held in Toulouse, France, *IFAC-PapersOnline*, vol. 50, no. 1, pp. 6971–6976, 2017.
[10] **N.T. Nguyen**, I. Prodan, L. Lefèvre, "Multi-layer optimization-based control design for quadcopter trajectory tracking", in Proc. of the *25th Mediterranean Conference on Control and Automation (MED'17)*, pp. 601–606, held in Valletta, Malta, 2017.

Presentations:

1. *"Reliable motion planning strategies for nonlinear dynamics"*, presentation at Institute for Robotics and Cognitive Systems, Lübeck, Germany, on August 15th, 2019.
2. *"Trajectory generation and tracking for a quadcopter system"*, presentation at Department of Electronic & Electrical Engineering, of the University College London, London, United Kingdom, on May 8th, 2019.
3. *"On the use of a computed-torque control law for the terminal region of an NMPC scheme"*, presentation at the laboratory of Automatic Control and System Dynamics, at Technische Universität (TU), Chemnitz, Germany, on February 14th, 2019.
4. *"Flat trajectory design and tracking with saturation guarantees: a nano-drone application"*, presentation at the laboratory of Automatic Control and System Dynamics, at Technische Universität (TU), Chemnitz, Germany, on January 16th, 2019.
5. *"An NMPC design for stabilizing thrust-propelled underactuated vehicles"*, presentation at the Réunion du GT–CPNL (Groupe de Travail – Commande Prédictive Nonlinéaire), at ONERA (The French Aerospace Lab), Châtillon, France, on June 4th, 2018.
6. *"Multi-layer optimization-based control design for quadcopter trajectory tracking"*, open invited track for GDR MACS Young PhD researchers at 20th IFAC World Congress, Toulouse, France, on July 11th, 2017.
7. *"Reliable motion planning strategies for nonlinear dynamics under uncertainties"*, presentation at Norwegian University of Science and Technology (NTNU), Trondheim, Norway, on June 9th, 2017.

1.4 Organization of the manuscript

This thesis is organized into five chapters excluding this introduction:

Chapter 2 presents the global view of the multicopter control problem. It details the dynamical model, the off-line trajectory generation process and the general ideas on hierarchical control design of the system. For illustration, we also introduce two control design candidates, i.e., the feedback linearization control via flatness for the position controller at the high level and the computed-torque attitude controller at the low level with their advantages/disadvantages. Then, the trajectory generation results are shown for an indoor trajectory generation application and a building inspection scenario. The indoor trajectory is then employed for conducting simulation and experimental tests of the tracking control designs.

Chapter 3 addresses the use of a computed-torque control law within an NMPC framework for a particular class of systems, the "computed-torque like" systems. We firstly recapitulate the existing principles of designing NMPC with terminal stabilizing constraints and apply them to the computed-torque controller. We propose an ellipsoid input constraint admissible set obtained by using Taylor's approximation theory and a condition on choosing the control gain such that the set also becomes positive invariant. The bound of the computed-torque controller within the set is constructed in terms of the state. All the aforementioned elements are gathered to guarantee the stability of an NMPC design using the admissible invariant set as its terminal constraint set. Next, we apply the method to design an NMPC attitude controller for stabilizing the rotation dynamics of the multicopter system. Several elements are modified to adapt to the fast changes of the desired angle. The proposed method is validated through simulations and comparison with the quasi-infinite horizon NMPC design.

Chapter 4 proposes three different NMPC designs for the position controller. The first design makes use of a terminal invariant set under the feedback linearization controller obtained via flatness (provided in Chapter 2). We show that the set can be extended to cover an arbitrarily defined compact set containing all the feasible initial states, and by this, the design achieves its semi-global stability property. The second NMPC design has its terminal constraint set given as a relaxed invariant set under the aforementioned feedback linearization controller. The set is obtained in a simple formulation - linear box-type constraints, hence, reducing the complexity as well as the computation time while still guaranteeing the stability. The final NMPC scheme does not use terminal stabilizing constraints but only requires an appropriate prediction horizon length. We demonstrate that using the feedback linearization controller reduces the required prediction horizon length in comparison with using a standard linear controller. The contributions are validated through proofs of concepts, various examples and extensive simulations. The controllers are also experimentally tested over a real nano-quadcopter platform.

Chapter 5 addresses a FTC (Fault Tolerant Control) design for a quadcopter system under stuck rotor fault. The reconfiguration part consists of the NMPC attitude controller introduced in Chapter 3 and the speed calculator which provides the rotor speeds references to the rotors. A fault diagnosis module is also proposed to identify the faulty rotor and estimate the stuck speed. Simulation results show the effectiveness of the whole scheme.

Chapter 6 concludes the thesis and discusses future directions.

The foregoing outline is illustrated in Figure 1.4.1 where the arrows show the dependencies between chapters.

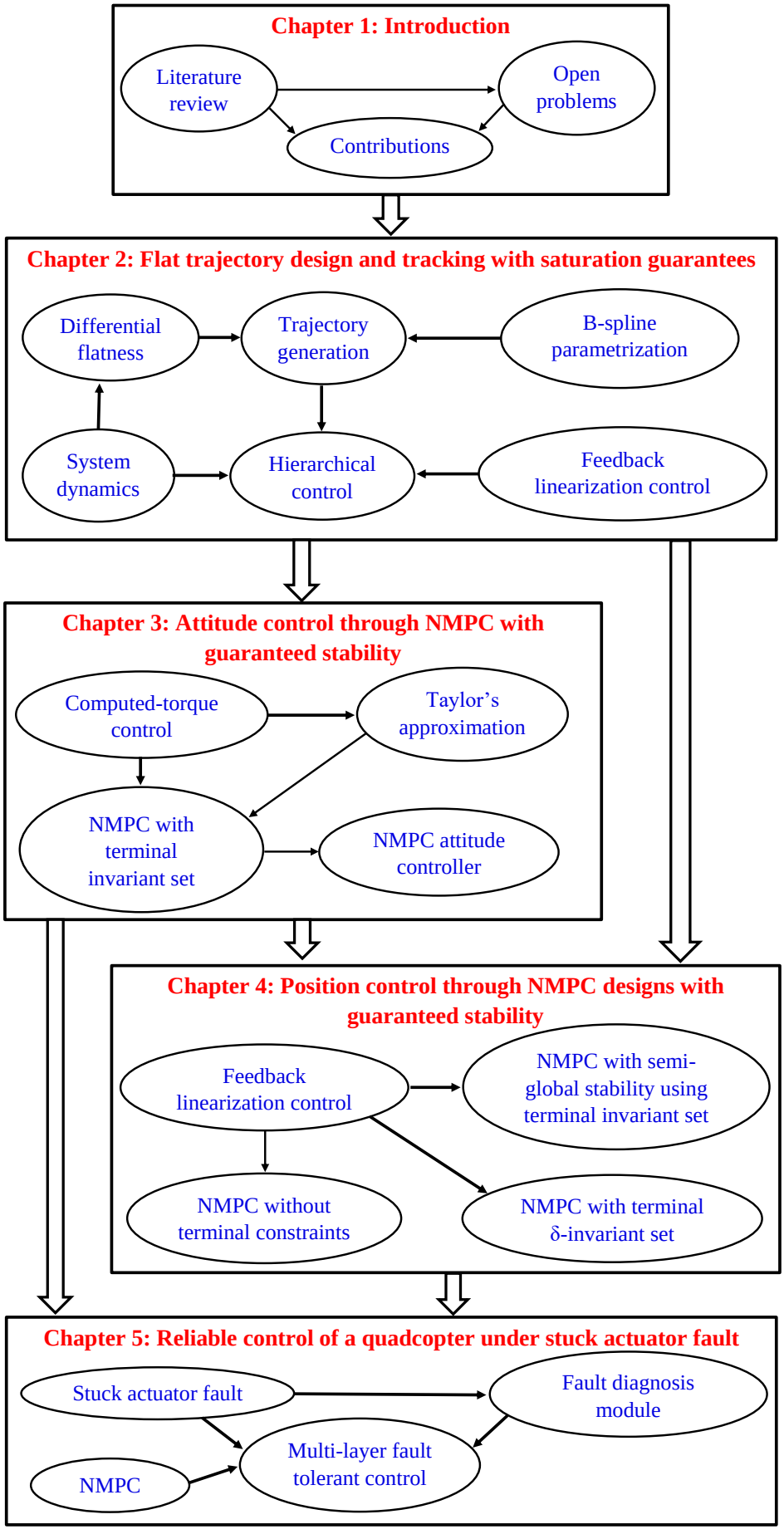


Figure 1.4.1: The organization of the thesis.

Chapter 2

Flat trajectory design and tracking with saturation guarantees

Let us recall the whole control process for a multicopter system given in Figure 1.1.1 in which, a two-step control approach is employed: off-line trajectory generation and online tracking design, as similar to various existing works in the literature [Mellinger and Kumar, 2011, Roza and Maggiore, 2012, Prodan et al., 2013, Zhao and Go, 2014]. The reference trajectory has to take into account nonlinear dynamics with various operating constraints of the system. Constrained trajectory generation problems for aerial vehicles are intensively studied in the literature [Chamseddine et al., 2012, Prodan et al., 2013, Stoican et al., 2017] which recommends a feasible and appropriate solution, i.e, employing differential flatness [Fliess et al., 1995] for formulating the optimal trajectory generation problems taking into account the system constraints as considered in [Mellinger and Kumar, 2011, Hehn and D'Andrea, 2011, Prodan et al., 2013, Hehn and D'Andrea, 2015, Nguyen et al., 2018b]. The flat output characterizations allow us to mathematically formulate a reference path that tracks specific objectives (i.e., passing through a priori given way-points, consumption minimization, state/input constraints satisfaction) [Stoican et al., 2017, Stoican et al., 2015, Prodan et al., 2013]. The next step is to parametrize the flat outputs by employing a specific parametrization [Cowling et al., 2007, Bipin et al., 2014, Mueller and D'Andrea, 2013, Engelhardt et al., 2016, Lu et al., 2017], e.g., user-defined piecewise functions [Mellinger and Kumar, 2011], Laguerre polynomials [Cowling et al., 2007], B-splines functions [Lu et al., 2017]. Then, the optimization problems are solved by using various existing solvers (with the caveat that the problem is often nonlinear and nonconvex and, thus, requires specialized tools). In this thesis, we limit ourselves to the off-line trajectory generation problem of the multicopter system. When solving an off-line trajectory generation (almost) all system's state and input constraints are taken into account since the restrictions on the computation time are usually less demanding [Cowling et al., 2007], hence, allowing us to fully exploit the capability of the system.
Next, for the online tracking mechanism, within the two layers of the hierarchical control scheme chosen by the aforementioned reasons, there are various existing control designs such as classical PID control [Rivera and Sawodny, 2010], LQR (Linear-quadratic regulator) control [Kuantama et al., 2018], adaptive control [Ha et al., 2014], feedback linearization-based control [Formentin and Lovera, 2011, Nguyen et al., 2017b] and optimization-based control [Mueller and D'Andrea, 2013, Nguyen et al., 2017a, Zanelli et al., 2018, Nguyen et al., 2019b]. Beside the natural capability of fulfilling the system constraints of the optimization-based control strategy [Mayne et al., 2000], most of the other approaches do not consider constraint validation, not even saturation input constraints (e.g., [Rivera and Sawodny, 2010, Formentin and Lovera, 2011, Freddi et al., 2011]). This is apologized by the complexity of the control designs taking into account

constraints but bounding the inputs changes the system's expected response and probably leads to undesired consequences (see, e.g., the anti-windup issue [Wu and Lu, 2004]). We would like to emphasize the work done in [Cao and Lynch, 2016] where the authors design a feedback control scheme which provides bounded thrust, roll and pitch angles based on the body frame (BF) consideration. However, it requires a convoluted transformation of the reference from the inertial frame (IF) to the BF since the reference trajectory is usually designed based on the IF, e.g. passing through way-points which are determined w.r.t. the IF. Moreover, when considering the IF representation of the system, [Cao and Lynch, 2016] also state that it is impossible to impose bounds on these two angles separately due to the existence of the yaw angle in the roll and pitch angles. Thus, the work raises the following question:

> Is it really impossible to design a non-optimization-based control taking into account the input constraints from the more simpler viewpoint - the IF frame ?

The answer for this question is given in this chapter through an original feedback linearization control design published in [Nguyen et al., 2018b] which employs the differential flatness representation of the multicopter system to derive the control law with saturation guarantees. By exploiting the robustness properties of the flatness representation of the system, the input constraints are ensured even under change of the predefined direction (yaw) angle. More precisely, the contributions of this chapter are:

i) propose several reliable constraint formulations which is robust under change of the predefined yaw angle value [Nguyen et al., 2018a]. The formulations are constructed based on the differential flatness representation of the multicopter system and are employed for both trajectory generation and tracking control design processes.

ii) formulate the constrained trajectory generation problem with the cost minimizing the path length. The approach is validated through two applications on standard way-point passing scenario [Nguyen et al., 2018b] and building inspection problem [Stoican et al., 2019].

iii) propose a feedback linearization position controller (at high control level) facilitated by nested control design which provides bounded inputs (thrust, roll and pitch angles) [Nguyen et al., 2018b].

iv) propose a modification on the original nested control design [Teel, 1992] which allows the system to have larger saturation limits. The limits now vary accordingly to the a priori given references, thus, are able to exploit more capability of the system than the standard fixed limits [Nguyen et al., 2018b].

v) propose a condition for choosing the angle bounds employed in the control design based on the a priori given trajectory and a condition for ensuring the existence of all the related parameters consisting of the reference trajectory, the control design and the input saturation limits. Thus, we create an unified design scheme for trajectory generation and tracking with bounded thrust and bounded angles while respecting the physical constraints of the system [Nguyen et al., 2018b].

vi) validate the control method through simulation and experimental testing over the nano quadcopter Crazyflie 2.0 platform [Giernacki et al., 2017].

The chapter is organized as follows. Section 2.1 recapitulates the dynamical model of a standard multicopter system. Next, the differential flatness representation and the trajectory generation problem is detailed in Section 2.2. Then, Section 2.3 presents the hierarchical control

scheme and the control problems within each layer. In Section 2.4, the feedback linearization control candidate for the high control level is presented. Section 2.5 shows the validation of the proposed approaches under simulation and experiment by using a real nano-quadcopter platform.

2.1 Dynamical models of multicopters

This section recapitulates the dynamical models of a standard multicopter system using fixed pitch propellers (i.e., producing only upward thrust, compared with variable pitch propeller [Cutler et al., 2011]) as given in [Roza and Maggiore, 2014, Formentin and Lovera, 2011, Zhao and Go, 2014]. The multicopter operates in two coordinate systems: the global frame $\mathcal{G}$ fixed to the ground and the body frame $\mathcal{B}$ attached to the vehicle as shown in Figure 2.1.1.

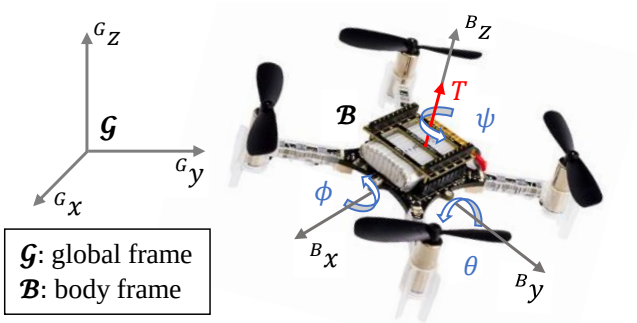


Figure 2.1.1: A multicopter vehicle and its coordinate systems.

It manifests an underactuated, highly coupled and nonlinear dynamical model as illustrated in Figure 2.1.2. The inputs $\mathsf{u} \triangleq [T\ \tau]^\top$ consist of the thrust, $T \in \mathbb{R}_+$, and three torques gathered in $\tau \in \mathbb{R}^3$. The outputs are the three positions along the three x, y, z axis of the global frame $\mathcal{G}$ gathered in $\xi \triangleq [x\ y\ z]^\top$ and the roll, pitch, yaw angles gathered in $\eta \triangleq [\phi\ \theta\ \psi]^\top$ whose directions are defined by the three blue arrows in Figure 2.1.2.

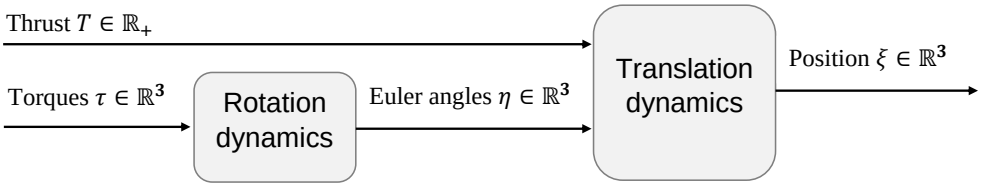


Figure 2.1.2: Coupled dynamical scheme of a standard multicopter system.

Next, to derive the dynamical model of the multicopter system using the Euler–Lagrange approach, we consider the multicopter model as a rigid body as usually asummed in various references [Roza and Maggiore, 2014, Formentin and Lovera, 2011, Hasan and Johansen, 2018].

Rotation dynamics of the multicopter system:
We firstly address the rotation dynamics of the multicopter system which is influenced by the

gyroscopic effect [Chaturvedi et al., 2011] and the actuator torques τ, hence, being given by[1]:

$$J\dot{\omega} = -\omega \times (J\omega) + \tau, \tag{2.1.1}$$

where $\omega \triangleq [\omega_x \ \omega_y \ \omega_z]^\top \in \mathbb{R}^3$ represents the angle rates in frame $\boldsymbol{\mathcal{B}}$ and $J \triangleq \text{diag}\{J_x, J_y, J_z\} \in \mathbb{R}^{3\times3}$ stands for the inertia tensor of the multicopter system.
Then, the angle rates ω affects the rotation motion of the multicopter system via the following relation:

$$\dot{\mathsf{R}} = \mathsf{R}\,\text{sk}(\omega), \tag{2.1.2}$$

with $\mathsf{R} \in \mathcal{SO}(3)$, the rotation matrix and $\text{sk}(\omega)$ the skew-symmetric matrix of ω satisfying $\text{sk}(\omega)p = \omega \times p$ for any vector $p \in \mathbb{R}^3$. An usual explicit formulation of $\text{sk}(\omega)$ is as follows [Chaturvedi et al., 2011, Hehn and D'Andrea, 2015, Do, 2015]:

$$\text{sk}(\omega) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{2.1.3}$$

Since $\mathsf{R}^{-1} = \mathsf{R}^\top$ due to the properties of the group $\mathcal{SO}(3)$, (2.1.2) leads to:

$$\text{sk}(\omega) = \mathsf{R}^\top\dot{\mathsf{R}}. \tag{2.1.4}$$

As usually found in the works related to aerial applications [Freddi et al., 2011, Formentin and Lovera, 2011, Nguyen et al., 2017b, Hasan and Johansen, 2018], we employ the roll-pitch-yaw XYZ rotation sequence to transform the body frame $\boldsymbol{\mathcal{B}}$ to the global frame $\boldsymbol{\mathcal{G}}$ whose rotation matrix R is defined as:

$$\mathsf{R} = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}, \tag{2.1.5}$$

with ϕ, θ, ψ the roll, pitch, yaw angles illustrated in Figure 2.1.1. Then, by introducing (2.1.5) to (2.1.4), we have that:

$$\text{sk}(\omega) = \begin{bmatrix} 0 & -\dot{\psi}\cos\phi\cos\theta + \dot{\theta}\sin\phi & \dot{\psi}\sin\phi\cos\theta + \dot{\theta}\cos\phi \\ \dot{\psi}\cos\phi\cos\theta - \dot{\theta}\sin\phi & 0 & -\dot{\phi} + \dot{\psi}\sin\theta \\ -\dot{\psi}\sin\phi\cos\theta - \dot{\theta}\cos\phi & \dot{\phi} - \dot{\psi}\sin\theta & 0 \end{bmatrix}. \tag{2.1.6}$$

This leads to the following formulation of ω in (2.1.1) which is equivalent to (2.1.4) but simpler:

$$\omega = W\dot{\eta}, \tag{2.1.7}$$

with $\dot{\eta} \triangleq [\dot{\phi} \ \dot{\theta} \ \dot{\psi}]^\top$ and the matrix W defined as:

$$W = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix}. \tag{2.1.8}$$

Translation dynamics of the multicopter system:
Next, the translation dynamics of the multicopter system is affected by the thrust acting along the ${}^Bz$ axis of frame $\boldsymbol{\mathcal{B}}$ (as shown by the red arrow in Figure 2.1.1) and the gravity in the reverse

[1] further details on the rotation motion of a rigid body can be found in [Chaturvedi et al., 2011, Formentin and Lovera, 2011, Zhao and Go, 2014, Nguyen et al., 2017b]

${}^{G}z$ axis of frame $\boldsymbol{\mathcal{G}}$. Thus, by applying Newton's second law of motion, the translation dynamics are given by:

$$\ddot{\xi} = -g\mathsf{e}_3 + \frac{T}{m}\mathsf{R}\mathsf{e}_3, \tag{2.1.9}$$

in which $T \in \mathbb{R}_+$ is the magnitude of the thrust, $\mathsf{e}_3 \triangleq [0\ 0\ 1]^\top$ denotes the unit vector pointing along the z axis, g stands for the gravitational acceleration and m is the system mass. Thus, by combining the rotation and translation subsystems (2.1.1)–(2.1.9), the entire twelve-state, four-input dynamical model of the multicopter system (c.f. Figure 2.1.2) is given by:

$$\begin{aligned} \dot{\mathsf{x}} &= \mathsf{f}(\mathsf{x}, \mathsf{u}), \\ \mathsf{y} &= [x\ y\ z\ \psi]^\top, \end{aligned} \tag{2.1.10}$$

where the state vector is $\mathsf{x} \triangleq [\xi\ v\ \eta\ \omega]^\top \in \mathbb{R}^{12}$ with $v \triangleq \dot{\xi} \in \mathbb{R}^3$ the velocity vector, the input vector is $\mathsf{u} \triangleq [T\ \tau]^\top \in \mathbb{R}^4$ and the output vector $\mathsf{y} \in \mathbb{R}^4$ consists of the 3D position ξ and the yaw angle ψ. The function $\mathsf{f}(\mathsf{x}, \mathsf{u})$ gathers the appropriate elements taken from (2.1.1)–(2.1.9) and is given by:

$$\mathsf{f}(\mathsf{x}, \mathsf{u}) = \begin{bmatrix} v \\ -g\mathsf{e}_3 + \mathsf{R}\mathsf{e}_3 T/m \\ \overline{W}\omega \\ J^{-1}(-\omega \times (J\omega) + \tau) \end{bmatrix}, \tag{2.1.11}$$

in which, $\dot{\eta} = \overline{W}\omega$ is the inversion of (2.1.7) with $\overline{W} = W^{-1}$.

State and input constraints of the multicopter system:
Furthermore, the state and input constraints of the multicopter system (2.1.10) are given as follows:

$$\mathsf{x} \in \mathcal{X}_{\text{MC}} = \left\{ \mathsf{x} \in \mathbb{R}^{12} \,\middle|\, \langle |\phi|, |\theta| \rangle \leq \epsilon_{\max},\ \langle |\omega_x|, |\omega_y| \rangle \leq \omega_{\max} \right\}, \tag{2.1.12}$$

$$\mathsf{u} \in \mathcal{U}_{\text{MC}} = \left\{ \mathsf{u} \in \mathbb{R}^{4} \,\middle|\, |\tau| \leq \tau_{\max},\ 0 \leq T \leq T_{\max} \right\}, \tag{2.1.13}$$

with $\epsilon_{\max} \in (0, \pi/2)$ the defined maximum angle value, $\omega_{\max} \in \mathbb{R}_+$ the maximum angle rate, $T_{\max} > 0$ the thrust maximum limit and $\tau_{\max} \in \mathbb{R}^3_+$ gathering the maximum values of the three torques on the three axes of the multicopter system. Note that, we use in (2.1.12) the notation $\langle |\phi|, |\theta| \rangle \leq \epsilon_{\max}$ to denote "$|\phi| \leq \epsilon_{\max}$ and $|\theta| \leq \epsilon_{\max}$". A similar reasoning is also applied for $\langle |\omega_x|, |\omega_y| \rangle \leq \omega_{\max}$. Imposing the condition on $\epsilon_{\max} < \pi/2$ is not only to avoid singularities in (2.1.8) but also to constrain the multicopter system not to tilt up to the perpendicular orientation which causes the loss of upward thrust and makes the system unable to resist the gravity.

Remark 2.1.1. First, note that in (2.1.12), no constraints on the positions (x, y, z) and the yaw angle ψ contribute to the shape of the set $\mathcal{X}_{\text{MC}}$ since (x, y, z, ψ) will appear as the decision parameters in the reference trajectory generation problem. Depending on the control application, position and yaw angle constraints will be considered, e.g., by passing through way-points or maintaining a fixed direction.
Second, note that, the roll, pitch angles (ϕ, θ) and the two angle rates (ω_x, ω_y) as in (2.1.12) can be constrained by different limits for each variable. However, it is customary to impose the constraints as in (2.1.12) due to the symmetry of the multicopter system [Hehn and D'Andrea, 2015, Lu et al., 2017, Prach and Kayacan, 2018]. In any case, different bounds for these parameters can be easily imposed at the price of a more convoluted notation which offers no further qualitative gain.

Lastly, bear in mind that the considered inputs: the thrust T and the torques τ as in (2.1.10) are actually resulted from the rotor configuration corresponding to the real multicopter system (depending on the number of rotors and their positioning, etc) [Intwala and Parikh, 2015]. Physically, the rotors speeds have bounds which either translate into additional constraints on the thrust and torques as in (2.1.13) (i.e., popular configurations are those of quadcopters, hexacopters). The quadcopter system case will be presented in Section 5.2.1 where the constraints on four rotor speeds are transformed into those on the thrust and torques even under the fault of one rotor. □

In the next section, we present the method for generating off-line a reference trajectory for the multicopter system (2.1.10) which takes into account the internal dynamics (2.1.1)-(2.1.9), the system constraints (2.1.12)-(2.1.13) and additional user-defined constraints such as passing through some way-points. Furthermore, the chosen method allows to generate an optimal trajectory which minimizes certain objectives such as curve length or acceleration.

2.2 Off-line constrained trajectory generation

This section presents an approach for tackling the off-line trajectory generation problem of the multicopter system. The trajectory has to be *feasible* which means that it respects the internal dynamics (2.1.1)-(2.1.9) and the state and input constraints (2.1.12)-(2.1.13). Furthermore, the trajectory also takes into account various user-defined constraints such as way-points passing and initial boundary conditions. For solving these problems, in the following, we firstly present the differential flatness representation [Fliess et al., 1995, Lévine, 2011] of the multicopter system. Then, we introduce several robust constraints formulations which are derived from the mathematical properties of the flatness. Next, the cost and the constraints are recasted in a B-splines parametrization [Piegl and Tiller, 1995] in order to generate an optimal reference trajectory which minimizes the path length as similarly considered in [Stoican et al., 2015, Stoican et al., 2017] and satisfies the imposed constraints.

2.2.1 Differential flatness properties of the multicopter system

Among various approaches for optimal trajectory generation for multicopter systems, a popular solution is the use of flat output characterizations [Fliess et al., 1995]. There is a number of works like [Chamseddine et al., 2012, Hehn and D'Andrea, 2015, Stoican et al., 2015, Stoican et al., 2017, Eliker et al., 2018] which employ differential flatness for dealing with the problems of constrained trajectory generation for aerial vehicles in general and the multicopter system in particular. Important features of flatness are well-know, as for example, it takes explicitly into account the internal dynamics of the systems and (with some difficulty) state and input constraints. The flatness property of the multicopter system is well-known in the literature and has been employed in various works related to trajectory generation and control design of the system like [Rivera and Sawodny, 2010, Formentin and Lovera, 2011, Bipin et al., 2014, Nguyen et al., 2017b, Nguyen et al., 2018b] . Therefore, we briefly recapitulate the results hereinafter.

Definition 2.2.1 ([Fliess et al., 1995]). *Consider the nonlinear system in general form:*

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \tag{2.2.1}$$

with the state $\boldsymbol{x}(t) \in \mathbb{R}^n$ and input vectors $\boldsymbol{u}(t) \in \mathbb{R}^m$. The system (2.2.1), satisfying the controllability property, is differentially flat if there exists a flat output $\mathbf{z}(t) \in \mathbb{R}^m$ (i.e., required

to have the same dimension with the input $\boldsymbol{u}(t)$*) [Fliess et al., 1995]:*

$$\mathbf{z}(t) = \Upsilon(\boldsymbol{x}(t), \boldsymbol{u}(t), \dot{\boldsymbol{u}}(t), \dots, \boldsymbol{u}^{(q)}(t)), \tag{2.2.2}$$

such that the states and inputs can be algebraically expressed in terms of $\mathbf{z}(t)$ *and a finite number of its higher-order derivatives:*

$$\boldsymbol{x}(t) = \Upsilon_x(\mathbf{z}(t), \dot{\mathbf{z}}(t), \cdots, \mathbf{z}^{(q)}(t)), \tag{2.2.3a}$$
$$\boldsymbol{u}(t) = \Upsilon_u(\mathbf{z}(t), \dot{\mathbf{z}}(t), \cdots, \mathbf{z}^{(q+1)}(t)). \tag{2.2.3b}$$

In Definition 2.2.1, the derivatives are taken in terms of the time t, hence, denoted by $\dot{\boldsymbol{x}}(t)$ for the state $\boldsymbol{x}(t)$ as in (2.2.1). However, throughout most of the thesis, the time notation (t) will be discarded whenever the meaning is straightforward from the context.

Proposition 2.2.2. *The multicopter system* $\dot{\mathsf{x}} = \mathsf{f}(\mathsf{x}, \mathsf{u})$ *defined by* (2.1.10) *is differentially flat with the associated flat output taken as the output of the system, i.e.,* $\mathsf{y} = [x\ y\ z\ \psi]^\top$ *from* (2.1.10).

Proof. By using the output y as the flat output, the first condition (2.2.2) follows immediately from the definition of the flat output (2.2.4) while the second condition (2.2.3) is verified constructively. Let denote the flat output $\mathbf{z}$ as follows:

$$\mathbf{z} = \begin{bmatrix} x\ y\ z\ \psi \end{bmatrix}^\top. \tag{2.2.4}$$

Let us consider the states $\mathsf{x} = [\xi\ v\ \eta\ \omega]^\top$ defined in (2.1.10). Beside the clear expression of the position ξ and the yaw angle ψ in terms of the flat output $\mathbf{z}$, it is also trivial to have the velocity $v = \dot{\xi}$ expressed in terms of $\dot{\mathbf{z}}$. Next, by considering the translation dynamics (2.1.9), the roll, pitch angles are given by:

$$\phi = \Upsilon_\phi(\ddot{\xi}, \psi) \triangleq \arcsin\left(\Phi_x \sin\psi - \Phi_y \cos\psi\right), \tag{2.2.5}$$
$$\theta = \Upsilon_\theta(\ddot{\xi}, \psi) \triangleq \arctan\left(\Theta_x \cos\psi + \Theta_y \sin\psi\right), \tag{2.2.6}$$

in which, $\Phi_x, \Phi_y, \Theta_x, \Theta_y$ are defined as:

$$\Phi_x = \frac{\ddot{x}}{\sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} + g)^2}}, \tag{2.2.7a}$$
$$\Phi_y = \frac{\ddot{y}}{\sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} + g)^2}}, \tag{2.2.7b}$$
$$\Theta_x = \frac{\ddot{x}}{\ddot{z} + g}, \tag{2.2.7c}$$
$$\Theta_y = \frac{\ddot{y}}{\ddot{z} + g}, \tag{2.2.7d}$$

with g the gravitational acceleration from (2.1.9). Next, from (2.1.7), the angular velocity vector ω is expressed as a function of the angle vector η and its derivative $\dot{\eta}$, i.e., $\omega = W\dot{\eta}$. Thus, by introducing (2.2.5)-(2.2.6), the angular velocities $\omega \triangleq [\omega_x\ \omega_y\ \omega_z]^\top$ are expressed as the function of $\mathbf{z}$ and its derivatives up to the third-order as follows:

$$\omega_x = \Upsilon_{\omega_x}(\ddot{\xi}, \dddot{\xi}, \psi) \triangleq \frac{\dot{\Phi}_x \sin\psi - \dot{\Phi}_y \cos\psi}{\cos\phi}, \tag{2.2.8}$$
$$\omega_y = \Upsilon_{\omega_y}(\ddot{\xi}, \dddot{\xi}, \psi) \triangleq \cos\phi \cos^2\theta \left(\dot{\Theta}_x \cos\psi + \dot{\Theta}_y \sin\psi\right), \tag{2.2.9}$$
$$\omega_z = \Upsilon_{\omega_z}(\ddot{\xi}, \dddot{\xi}, \psi, \dot{\psi}) \triangleq -\sin\phi \cos^2\theta \left(\dot{\Theta}_x \cos\psi + \dot{\Theta}_y \sin\psi\right) + \frac{\cos\theta}{\cos\phi}\dot{\psi}, \tag{2.2.10}$$

in which, we emphasize that $\dot{\Phi}_x \triangleq \dfrac{d\Phi_x}{dt}$ is a function of the translation acceleration $\ddot{\xi}$ and the jerk $\dddot{\xi}$. Similar expressions are obtained for $\dot{\Phi}_y$, $\dot{\Theta}_x$ and $\dot{\Theta}_y$.
The next objective is to find the flatness-based representation of the input vector $\mathsf{u} = [T\ \tau]^\top$. From the translation dynamics (2.1.9), we have that $\|\ddot{\xi}+g\mathsf{e}_3\| = \|\mathsf{R}\mathsf{e}_3\|T/m$. By using $\|\mathsf{R}\mathsf{e}_3\| = 1$, the thrust T is obtained as a function of the translation acceleration $\ddot{\xi}$ as follows:

$$T = \Upsilon_T(\ddot{\xi}) = m\sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z}+g)^2}. \tag{2.2.11}$$

Finally, the input torques τ is calculated by using the rotation dynamics (2.1.1):

$$\tau = \Upsilon_\tau(\ddot{\xi}, \dddot{\xi}, \xi^{(4)}, \psi, \dot{\psi}, \ddot{\psi}) = J\dot{\omega} + \omega \times (J\omega), \tag{2.2.12}$$

with ω from (2.2.8)-(2.2.10), hence, allowing to further interpret the torque τ in the flat output space. Note that, the full expression of the torque τ requires up to the fourth-order derivative of the flat output $\mathbf{z}$. This also completes the proof. □

Remark 2.2.3. In [Bipin et al., 2014, Rivera and Sawodny, 2010, Formentin and Lovera, 2011], a simplified version of flat representation is usually employed with the assumption that the yaw angle ψ equals zero. We do not follow these assumptions since we want to exploit all the degrees of freedom, thus fully taking into account the nonlinear system dynamics including the yaw motion. Furthermore, in some of our published works [Nguyen et al., 2017b, Nguyen et al., 2018a, Nguyen et al., 2018b], a different flat output $\boldsymbol{z}$ as in (2.2.4) is employed, i.e., $\boldsymbol{z} = \begin{bmatrix} x & y & z & \arctan(\psi/2) \end{bmatrix}^\top$. This flat output eliminates the trigonometric term of ψ as appearing in (2.2.5)–(2.2.6), hence, providing less convoluted flatness representations for the multicopter system (especially for the torques τ as implicitly given in (2.2.12)). E.g.:

$$\phi = \arcsin\left(\frac{2z_4\ddot{z}_1 - (1-z_4^2)\ddot{z}_2}{(1+z_4^2)\sqrt{z_1^2+z_2^2+(z_3+g)^2}}\right), \tag{2.2.13}$$

with $\begin{bmatrix} z_1 & z_2 & z_3 & z_4 \end{bmatrix} = \begin{bmatrix} x & y & z & \arctan(\psi/2) \end{bmatrix}$. However, no trigonometric terms appear in (2.2.13), hence, hiding some trigonometric properties such as $\sin^2\psi + \cos^2\psi = 1$ (can still be obtained from (2.2.13) but not straightforwardly), which will be of use for us to construct several reliable constraints formulations in the next section. □

2.2.2 Reliable constraints formulation

When solving the trajectory generation problem of a multicopter system, many approximations (e.g., small or even null roll, pitch angles [Chamseddine et al., 2012, Gandolfo et al., 2016]) are employed in the literature for simplifying the system dynamics (2.1.1)-(2.1.9). Furthermore, various states and inputs constraints of the aerial systems are usually imposed based on a predefined yaw angle trajectory, e.g., zero angle as illustrated in Figure 2.2.1 [Cowling et al., 2007, Mueller and D'Andrea, 2013, Lu et al., 2017] or a spline with specific degree [Engelhardt et al., 2016]. Therefore, any change in the yaw angle trajectory (for example, intentionally modified vehicle direction pointing towards the target for camera applications [Engelhardt et al., 2016] or changes in the direction due to faulty events [Nguyen et al., 2017b]), will obviously affect the validation of the above mentioned constraints. These problems raise our interest in designing several reliable angular constraints formulations which bound the roll, pitch angles (ϕ, θ as in (2.2.5)-(2.2.6)) and the angular velocities (ω_x, ω_y) as in (2.2.8)-(2.2.9)) without requiring any information of the yaw angle. Thus, we can decouple the trajectory generation problem of the multicopter system into two separate parts:

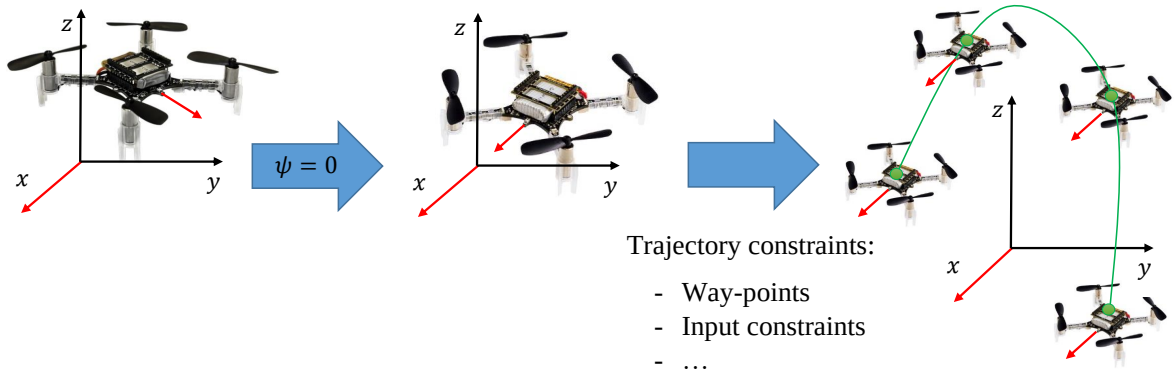


Figure 2.2.1: Illustration of reference trajectory generation under assumption of $\psi = 0$.

i) position reference trajectory providing the reference ξ_r of the position ξ in (2.1.9).

ii) yaw reference trajectory ψ_r.

Especially, the position reference trajectory ξ_r will still satisfy the predefined constraints on the angles and the angular velocities even under any modification on the yaw trajectory due to faulty events or intentional modification of the vehicle direction such as pointing towards the target for camera application.

2.2.2.1 Constraints on roll, pitch angles ϕ, θ

Proposition 2.2.4 ([Nguyen et al., 2018a]). *The roll and pitch angles (i.e., $\phi = \Upsilon_\phi(\ddot{\xi}, \psi)$ and $\theta = \Upsilon_\theta(\ddot{\xi}, \psi)$ from (2.2.5)-(2.2.6)) are bounded by the angle boundary $\epsilon \in [0, \frac{\pi}{2})$ as follows:*

$$\langle |\phi|, |\theta| \rangle \leq \epsilon, \ \forall \psi \in \mathbb{R}, \tag{2.2.14}$$

where the angle boundary $\epsilon \in [0, \frac{\pi}{2})$ is defined as:

$$\epsilon \triangleq \Upsilon_\epsilon(\ddot{\xi}) = \arcsin\left(\sqrt{\frac{\ddot{x}^2 + \ddot{y}^2}{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} + g)^2}}\right). \tag{2.2.15}$$

Proof. At first, let us consider the angle boundary $\epsilon \in [0, \frac{\pi}{2})$ defined in (2.2.15), after some trigonometric manipulations, we have that:

$$\sin\epsilon = \sqrt{\frac{\ddot{x}^2 + \ddot{y}^2}{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} + g)^2}}, \tag{2.2.16}$$

$$\tan\epsilon = \sqrt{\frac{\ddot{x}^2 + \ddot{y}^2}{(\ddot{z} + g)^2}}. \tag{2.2.17}$$

Furthermore, by applying the Cauchy-Schwarz inequality [Bhatia and Davis, 1995] to the flatness-based formulations of the roll, pitch angles from (2.2.5)-(2.2.6), we can bound the two angles as

follows:

$$|\sin\phi| \leq \sqrt{(\sin^2\psi + \cos^2\psi)(\Phi_x^2 + \Phi_y^2)} = \sqrt{\frac{\ddot{x}^2 + \ddot{y}^2}{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z}+g)^2}}, \tag{2.2.18}$$

$$|\tan\theta| \leq \sqrt{(\sin^2\psi + \cos^2\psi)(\Theta_x^2 + \Theta_y^2)} = \sqrt{\frac{\ddot{x}^2 + \ddot{y}^2}{(\ddot{z}+g)^2}}. \tag{2.2.19}$$

Then, introducing (2.2.16)-(2.2.17) to (2.2.18)-(2.2.19), we have that: $\sin(|\phi|) \leq \sin(\epsilon)$ and $\tan(|\theta|) \leq \tan(\epsilon)$. These further provide $|\phi| \leq \epsilon$ and $|\theta| \leq \epsilon$ due to the monotonic property of the $\sin(\cdot)$ and $\tan(\cdot)$ functions in the range of $(-\frac{\pi}{2}, \frac{\pi}{2})$ in which the angles ϕ and θ are constrained to stay as detailed in (2.1.12). □

2.2.2.2 Constraints on angular velocities ω_x, ω_y

Proposition 2.2.5 ([Hehn and D'Andrea, 2015]). *The angular velocities* $\omega_x = \Upsilon_{\omega_x}(\ddot{\xi}, \dddot{\xi}, \psi)$ *from* (2.2.8) *and* $\omega_y = \Upsilon_{\omega_y}(\ddot{\xi}, \dddot{\xi}, \psi)$ *from* (2.2.9) *are bounded by the angle rate boundary* $\omega_b \in \mathbb{R}_+$ *as follows:*

$$\langle |\omega_x|, |\omega_y| \rangle \leq \omega_b, \ \forall \psi \in \mathbb{R}, \tag{2.2.20}$$

where the angle rate boundary $\omega_b \in \mathbb{R}_+$ *is defined as:*

$$\omega_b \triangleq \Upsilon_{\omega_b}(\ddot{\xi}, \dddot{\xi}) = \sqrt{\frac{\dddot{x}^2 + \dddot{y}^2 + \dddot{z}^2}{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z}+g)^2}}. \tag{2.2.21}$$

Proof. Hereinafter, the proof is briefly presented by using our defined notations while more details can be found in [Hehn and D'Andrea, 2015].
Recall the translation dynamics given in (2.1.9). By introducing the flatness-based representation of the thrust T from (2.2.11) to (2.1.9), we have that:

$$\Phi = \mathsf{R}\mathsf{e}_3, \tag{2.2.22}$$

in which, $\Phi \triangleq [\Phi_x \ \Phi_y \ \Phi_z]^\top$ with Φ_x, Φ_y given in (2.2.7) and Φ_z defined as follows:

$$\Phi_z = \frac{\ddot{z}+g}{\sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z}+g)^2}}. \tag{2.2.23}$$

Next, by introducing $\dot{\mathsf{R}} = \mathsf{R}\,\mathrm{sk}(\omega)$ from (2.1.2) to the derivative of (2.2.22), we obtain that:

$$\dot{\Phi} = \mathsf{R}\,\mathrm{sk}(\omega)\mathsf{e}_3. \tag{2.2.24}$$

This allows to further express the angular velocities ω_x and ω_y as:

$$\begin{bmatrix} \omega_y \\ -\omega_x \\ 0 \end{bmatrix} = \mathsf{R}^\top \dot{\Phi}, \tag{2.2.25}$$

with $\mathsf{R}^\top = \mathsf{R}^{-1}$ as $\mathsf{R} \in \mathcal{SO}(3)$. Then, the following holds:

$$\langle \omega_x, \omega_y \rangle \leq \sqrt{\omega_x^2 + \omega_y^2} \leq \|\dot{\Phi}\| \leq \omega_b, \tag{2.2.26}$$

with ω_b from (2.2.21). The final inequality, i.e., $\|\dot{\Phi}\| \leq \omega_b$ is due to the unit norm property of Φ, i.e., $\|\Phi\| = \sqrt{\Phi_x^2 + \Phi_y^2 + \Phi_z^2} = 1$ with Φ_x, Φ_y from (2.2.7), Φ_z from (2.2.23) and is presented explicitly in [Hehn and D'Andrea, 2015]. This also completes the proof. □

Remark 2.2.6. The formulation of (ω_x, ω_y) as in (2.2.25) can be transformed into the flatness representations of $\omega_x = \Upsilon_{\omega_x}(\ddot{\xi}, \dddot{\xi}, \psi)$ and $\omega_y = \Upsilon_{\omega_y}(\ddot{\xi}, \dddot{\xi}, \psi)$ given in (2.2.8)-(2.2.9) by introducing the angles $\phi = \Upsilon_\phi(\ddot{\xi}, \psi)$ and $\theta = \Upsilon_\theta(\ddot{\xi}, \psi)$ from (2.2.5)-(2.2.6) to the rotation matrix R given in (2.1.5). The flatness representations $\Upsilon_{\omega_x}(\ddot{\xi}, \dddot{\xi}, \psi), \Upsilon_{\omega_y}(\ddot{\xi}, \dddot{\xi}, \psi)$ from (2.2.8)-(2.2.9) clearly show the dependence of ω_x, ω_y on $\ddot{\xi}, \dddot{\xi}, \psi$, hence, being compatible with formulating the trajectory generation problem. Also, note that the formulation (2.2.25) benefits from the rotation matrix $\mathsf{R} \in \mathcal{SO}(3)$ and hence, we can easily provide the bound ω_b as in (2.2.21). □

2.2.3 B-spline parametrization for the reference trajectory

This section introduces the parametrization of the reference trajectory $\xi_r(t) \triangleq [x_r(t)\ y_r(t)\ z_r(t)]^\top$ using B-spline basis functions [Piegl and Tiller, 1995]. B-splines curves are well-suited to flatness parametrization due to their ease of enforcing continuity and because their degree depends only up to which derivative is needed to ensure continuity (i.e., 4 for the considered multicopter system given in (2.1.10)) [Nguyen et al., 2018b, Stoican et al., 2017, Prodan et al., 2013, Stoican et al., 2015].
The B-spline parametrization of $\xi_r(t)$ is defined using two basic elements:

i) the degree $d \in \mathbb{N}$ ($d \geq 4$ for guaranteeing the continuity of the input torque τ from (2.2.12)).

ii) the control points in 3D space $\mathbb{R}^3$.

Let us construct the matrix $\mathbf{P} \in \mathbb{R}^{3\times(n+1)}$ consisting of $n+1$ control points $p_0, \ldots, p_n$ as follows:

$$\mathbf{P} = \begin{bmatrix} p_0, p_1, \ldots, p_n \end{bmatrix}. \tag{2.2.27}$$

Then, a B-spline reference trajectory $\xi_r(t)$ is defined as a linear combination of the control points and the same number of B-spline basis functions:

$$\xi_r(t) = \sum_{i=0}^{n} p_i B_{i,d}(t) = \mathbf{P}\mathbf{B}_d(t), \tag{2.2.28}$$

where $\mathbf{B}_d(t) = \begin{bmatrix} B_{0,d}(t) \ldots B_{n,d}(t) \end{bmatrix}^\top$ gathers the B-spline basis functions $B_{i,d}(t)$ defined by a recursive formula as follows:

$$B_{i,1}(t) = \begin{cases} 1, & \text{for } \tilde{t}_i \leq t < \tilde{t}_{i+1} \\ 0, & \text{otherwise} \end{cases}, \tag{2.2.29a}$$

$$B_{i,d}(t) = \frac{t - \tilde{t}_i}{\tilde{t}_{i+d-1} - \tilde{t}_i} B_{i,d-1}(t) + \frac{\tilde{t}_{i+d} - t}{\tilde{t}_{i+d} - \tilde{t}_{i+1}} B_{i+1,d-1}(t), \tag{2.2.29b}$$

for $d > 1$ and $i \in \{0, 1, \ldots, n\}$. The time instants $\tilde{t}_m$ with $m \in \{0, \ldots, n+d\}$ are chosen as follows:

$$\tilde{t}_m = \begin{cases} t_0, & \text{for } m \in \{0, \ldots, d-1\}, \\ t_0 + \left(\dfrac{t_f - t_0}{n-d}\right)(m-d), & \text{for } m \in \{d, \ldots, n\}, \\ t_f, & \text{for } m \in \{n+1, \ldots, n+d\}, \end{cases} \tag{2.2.30}$$

in which, the first and the last $d+1$ points equal to the initial and final time instants t_0 and t_f, respectively while the intermediary points $\tilde{t}_d, \ldots, \tilde{t}_n$ are equally distributed along these

extremes. In literature, this approach is usually referred as "fixed knot-vector" while there also exist different partition types of $\{\tilde{t}_0, \ldots, \tilde{t}_{n+d}\}$ as in (2.2.30) (e.g., uniform, open uniform, non-uniform) with the only requirement is on having $\tilde{t}_i \leq \tilde{t}_{i+1}$ [Piegl and Tiller, 1995]. The B-spline curve as in (2.2.28) yields various interesting properties which are enumerated in [Suryawan, 2011, Stoican et al., 2015, Nguyen et al., 2018b]. Some of them are related to the forthcoming results, and hence, are given in detail:

1. Endpoint interpolation: the first control point coincides with the initial point and the last control point coincides with the last point. E.g.:

$$p_0 = \xi(t_0), \quad p_n = \xi(t_f). \tag{2.2.31}$$

2. The B-spline curve lies in the convex hull generated by the control points gathered in $\mathbf{P}$;

3. The 'q' order derivatives of B-spline basis function can be expressed as linear combination of B-spline basis functions as:

$$\mathbf{B}_d^{(q)}(t) = \mathsf{K}_{d,d-q}\mathbf{B}_d(t), \tag{2.2.32}$$

with matrix $\mathsf{K}_{d,d-q}$ of appropriate dimensions and content given in [Suryawan, 2011, page 55].

2.2.4 State and input constraints parametrization

The reference trajectory has to satisfy the boundary conditions consisting of the initial and final values of the position ξ and up to its fourth-order derivative $\xi^{(4)}$, i.e., $\{\xi_0, \dot{\xi}_0, \ldots, \xi_0^{(4)}\}$ for the initial state and $\{\xi_f, \dot{\xi}_f, \ldots, \xi_f^{(4)}\}$ for the final state. By employing the two properties of the B-splines curve detailed in (2.2.31) and (2.2.32), the boundary constraints are constructed as:

$$p_0 = \xi_0, \quad \mathbf{P}\mathsf{K}_{d,d-q}\mathbf{B}_d(t = t_0) = \xi_0^{(q)}, \ \forall q \in \{1, \ldots, 4\}, \tag{2.2.33}$$

$$p_n = \xi_f, \quad \mathbf{P}\mathsf{K}_{d,d-q}\mathbf{B}_d(t = t_f) = \xi_f^{(q)}, \ \forall q \in \{1, \ldots, 4\}. \tag{2.2.34}$$

Note that, $\mathbf{B}_d(t = t_0) = \begin{bmatrix}1, 0, \ldots, 0\end{bmatrix}^\top$ and $\mathbf{B}_d(t = t_f) = \begin{bmatrix}0, \ldots, 0, 1\end{bmatrix}^\top$.
Moreover, during the flight, the vehicle has to pass through N way-points[2] w_k at the time instances t_k associated to them (with $k \in \{1, \ldots, N\}$), i.e.:

$$\mathbf{P}\mathbf{B}_d(t_k) = w_k, \forall k \in \{1, \ldots, N\}. \tag{2.2.35}$$

Note that, the N way-points w_k are not included the initial and final positions ξ_0, ξ_f from (2.2.33)–(2.2.34) in order not to conflict with these boundary conditions. The equation (2.2.35) can be relaxed to an inequality describing the region where the trajectory has to pass through (e.g. pass near the way-point within the region). We provide the equality form in (2.2.35) since it is conceptually simpler.
Next, since the multicopter system is subject to the state constraint set $\mathcal{X}$ from (2.1.12), the reference trajectory (including both the position and yaw trajectories) also has to limit the reference roll and pitch angles accordingly:

$$\langle |\phi_r|, |\theta_r| \rangle \leq \epsilon_{r_{\max}}, \tag{2.2.36}$$

[2]Note that, considering way-points at the trajectory generation level is coherent with typical software-hardware UAV configurations which use way-points in the communication protocol.

with the desired maximum value $\epsilon_{r_{\max}} \in (0, \epsilon_{\max}]$ and with $\epsilon_{\max}$ the constrained maximum angle value employed in the set $\mathcal{X}$ from (2.1.12). Note that, the reference angles are calculated by using the flatness-based representations of the angles, i.e., $\phi_r = \Upsilon_\phi(\ddot{\xi}_r, \psi_r)$ and $\theta_r = \Upsilon_\theta(\ddot{\xi}_r, \psi_r)$ from (2.2.5)-(2.2.6).
Moreover, since we do not want the multicopter system to have an aggressive altitude variation and the thrust has to respect its constraint $0 \leq T \leq T_{\max}$ from (2.1.13), the thrust reference, T_r, is also limited by its lower bound, $0 < T_{r_{\min}} < mg$ and upper bound, $mg < T_{r_{\max}} < T_{\max}$, given as follows:

$$T_{r_{\min}} \leq T_r \leq T_{r_{\max}}, \tag{2.2.37}$$

Note that, the value mg with m the system mass and g the gravity from (2.1.9) has to lie within the range $[T_{r_{\min}}, T_{r_{\max}}]$ for guaranteeing the hovering capability of the multicopter system.
Then, the reference trajectory also takes into account the constraints on the reference angular velocities from (2.1.12):

$$\langle |\omega_{x_r}|, |\omega_{y_r}| \rangle \leq \omega_{r_{\max}}, \tag{2.2.38}$$

with $\omega_{r_{\max}} \in \mathbb{R}_+$ the desired maximum angle rate satisfying $\omega_{r_{\max}} \leq \omega_{\max}$ with $\omega_{\max}$ from (2.1.12). The reference angular velocities are calculated as $\omega_{x_r} = \Upsilon_{\omega_x}(\dddot{\xi}_r, \ddot{\xi}_r, \psi_r)$ and $\omega_{y_r} = \Upsilon_{\omega_y}(\dddot{\xi}_r, \ddot{\xi}_r, \psi_r)$ with $\Upsilon_{\omega_x}(\cdot), \Upsilon_{\omega_y}(\cdot)$ from (2.2.8)-(2.2.9).
Then, we introduce the constraints formulations which rely only on the position reference ξ_r from (2.2.28). The formulations do not require any information on the undefined reference yaw ψ_r but still can guarantee the angular constraints (2.2.36)-(2.2.38).

Proposition 2.2.7. *If the position reference trajectory ξ_r has its acceleration $\ddot{\xi}_r$ and jerk $\dddot{\xi}_r$ satisfying the followings:*

(i) $\begin{bmatrix} \ddot{x}_r^2 + \ddot{y}_r^2 & (\ddot{z}_r + g)^2 \end{bmatrix}^\top$ *lies inside a polytope $\mathcal{P}$ defined as a convex sum of its vertices:*

$$\begin{bmatrix} \ddot{x}_r^2 + \ddot{y}_r^2 \\ (\ddot{z}_r + g)^2 \end{bmatrix} \in \mathcal{P} \triangleq \mathrm{Conv}(\mathbb{V}), \tag{2.2.39}$$

$$\mathbb{V} = \left\{ \left(0, \frac{T_{r_{min}}^2}{m^2}\right)^\top, \left(\frac{T_{r_{min}}^2}{m^2} \sin^2 \epsilon_{r_{max}}, \frac{T_{r_{min}}^2}{m^2} \cos^2 \epsilon_{r_{max}}\right)^\top, \right.$$
$$\left. \left(\frac{T_{r_{max}}^2}{m^2} \sin^2 \epsilon_{r_{max}}, \frac{T_{r_{max}}^2}{m^2} \cos^2 \epsilon_{r_{max}}\right)^\top, \left(0, \frac{T_{r_{max}}^2}{m^2}\right)^\top \right\}, \tag{2.2.40}$$

then, the constraints on roll and pitch angles $\langle |\phi_r|, |\theta_r| \rangle \leq \epsilon_{r_{max}}$ in (2.2.36), the constraint on thrust $T_{r_{min}} \leq T_r \leq T_{r_{max}}$ in (2.2.37) hold regardless of the reference yaw angle ψ_r.

(ii) the acceleration $\ddot{\xi}$ and the jerk $\dddot{\xi}_r$ satisfies $\Upsilon_{\omega_b}(\ddot{\xi}, \dddot{\xi}) \leq \omega_{r_{max}}$ with $\Upsilon_{\omega_b}(\cdot)$ from (2.2.21), i.e.:

$$\frac{\|\dddot{\xi}_r\|}{\sqrt{\ddot{x}_r^2 + \ddot{y}_r^2 + (\ddot{z}_r + g)^2}} \leq \omega_{r_{max}}, \tag{2.2.41}$$

then, the constraint on angular velocities $\langle |\omega_{x_r}|, |\omega_{y_r}| \rangle \leq \omega_{r_{max}}$ from (2.2.38) are satisfied regardless of the reference yaw angle ψ_r. □

Proof. For point *(i)*, from Proposition 2.2.4, we have that:

$$\langle |\phi_r|, |\theta_r| \rangle \leq \Upsilon_\epsilon(\ddot{\xi}_r), \forall \psi_r \in \mathbb{R}. \tag{2.2.42}$$

with $\Upsilon_\epsilon(\cdot)$ from (2.2.15). Thus, instead of directly constraining $\phi_r = \Upsilon_\phi(\ddot{\xi}_r, \psi_r)$ and $\theta_r = \Upsilon_\theta(\ddot{\xi}_r, \psi_r)$ from (2.2.5)-(2.2.6), we impose a constraint on the angle boundary $\Upsilon_\epsilon(\ddot{\xi}_r) \leq \epsilon_{r_{\max}}$. Next, the constraint on thrust in (2.2.37) is re-formulated as $T_{r_{\min}} \leq \Upsilon_T(\ddot{\xi}_r) \leq T_{r_{\max}}$ with $\Upsilon_T(\cdot)$ from (2.2.11). These provides the following system of inequalities:

$$\begin{bmatrix} 1 & -\tan^2(\epsilon_{r_{\max}}) \\ -1 & -1 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \ddot{x}_r^2 + \ddot{y}_r^2 \\ (\ddot{z}_r + g)^2 \end{bmatrix} \leq \frac{1}{m^2} \begin{bmatrix} 0 \\ -T_{r_{\min}}^2 \\ T_{r_{\max}}^2 \\ 0 \\ 0 \end{bmatrix}, \tag{2.2.43}$$

which is equivalently transformed into (2.2.40) using Fourier-Motzkin algorithm [Ziegler, 2012]. Next, from Proposition 2.2.5, the following holds:

$$\langle |\omega_{x_r}|, |\omega_{y_r}| \rangle \leq \Upsilon_{\omega_b}(\ddot{\xi}_r, \dddot{\xi}_r), \forall \psi_r \in \mathbb{R}. \tag{2.2.44}$$

Therefore, by constraining $\Upsilon_{\omega_b}(\ddot{\xi}_r, \dddot{\xi}_r) \leq \omega_{r_{\max}}$ as in point *(ii)* (2.2.41), $\langle |\omega_{x_r}|, |\omega_{y_r}| \rangle \leq \omega_{r_{\max}}$ are satisfied for any reference yaw angle ψ_r. Thus, completing the proof. □

Remark 2.2.8. At point (ii) of Proposition 2.2.7, the constraint $\Upsilon_{\omega_b}(\ddot{\xi}, \dddot{\xi}) \leq \omega_{r_{\max}}$ as in (2.2.41) can be replaced by a simpler formulation:

$$\|\dddot{\xi}_r\| \leq \frac{T_{r_{\min}}}{m} \omega_{r_{\max}}, \tag{2.2.45}$$

with $T_{r_{\min}}$ satisfying $T_{r_{\min}} \leq T_r = m\sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} + g)^2}$ from (2.2.37). The proof is straightforward to obtain by introducing $T_{r_{\min}} \leq m\sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} + g)^2}$ to (2.2.45). This replacement is also employed in the literature [Hehn and D'Andrea, 2011, Hehn and D'Andrea, 2015] but it is a trade-off between the simplicity and the conservativeness of the resulted reference angular velocities $(\omega_{x_r}, \omega_{y_r})$. □

Example 2.2.9. *Let us illustrate point (i) of Proposition 2.2.7 by using a multicopter system described in* (2.1.9) *with the parameters* $m = 0.5kg$, $T_{r_{min}} = 3N$, $T_{r_{max}} = 5N$ *and* $\epsilon_{r_{max}} = \pi/6$. *Then, the polytope* $\mathcal{P}$ *given in* (2.2.40) *and its vertices* $\mathbb{V} \triangleq \{v_1, \ldots, v_4\}$ *are depicted in Figure 2.2.2.*

2.2.5 Constrained trajectory generation with minimal length

In our work, we choose to minimize the length of the geometric trajectory since it appears to be realistic for various aerial applications [Prodan et al., 2013, Stoican et al., 2017, Nguyen et al., 2018b] . Note that any trajectory computed offline is only "preliminary" as it cannot account for run-time disturbance factors (wind gusts for example) and in this regard, the shortest trajectory cost is usually considered. Therefore, we will generate first the optimal-length position reference trajectory $\xi_r(t)$ along the time interval $[t_0, t_f]$ which is also subject to various states and input constraints (2.2.33)-(2.2.38). It results in an optimization problem with a quadratic cost function in terms of the control points $\mathbf{P}$ from (2.2.27) defined as:

$$\begin{aligned} \mathbf{P} = \arg\min_{\mathbf{P}} \int_{t_0}^{t_f} \dot{\xi}_r^\top(t)\dot{\xi}_r(t)dt = \arg\min_{\mathbf{P}} \int_{t_0}^{t_f} \left(\mathbf{P}\mathsf{K}_{d,d-1}\mathbf{B}_d(t)\right)^\top \left(\mathbf{P}\mathsf{K}_{d,d-1}\mathbf{B}_d(t)\right) dt, \\ \text{s.t. constraints (2.2.33), (2.2.34), (2.2.35), (2.2.40) and (2.2.41) are verified,} \end{aligned} \tag{2.2.46}$$

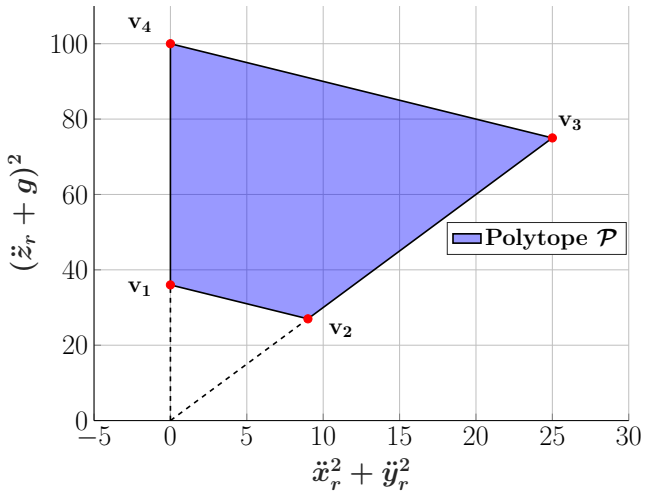


Figure 2.2.2: Illustration of the polyhedral set $\mathcal{P}$ and its vertices as in (2.2.40).

For solving the optimization problem (2.2.46), the constraints (2.2.33), (2.2.34), (2.2.35), (2.2.40) and (2.2.41) can be enforced at discrete moments [Stoican et al., 2015, Eliker et al., 2018] or even be guaranteed continuously [Stoican et al., 2017] along the time interval $[t_0, t_f]$. The first method is straightforward to implement but it does not guarantee the constraints fulfillment "in-between" the discrete moments. In case of constraints violation, increasing the sampling points can alleviate the problem. The inevitable drawback of high computation time, can be accepted since the trajectory generation is done offline before flight. The second method employs particular geometrical properties of the B-spline functions given in Section 2.2.3 in order to obtain a continuous constraints validation with fixed complexity (the number of constraints depends on the B-spline degree and on the number of control points but not on the number of waypoints) [Stoican et al., 2017].

Remark 2.2.10. Solving the nonlinear optimization problem (2.2.46) may provide a local minimum result instead of the expected globally optimal solution (i.e. the shortest curve). The solution is required to be manually verified after and the parameters (e.g., number of control points) may be changed until obtaining a good solution. Besides the minimal trajectory length, (2.2.46) can take into account various optimization objectives like input variation, magnitude, energy minimization and the like [Eliker et al., 2018]. □

After solving (2.2.46), we obtain the geometric reference trajectory, $\xi_r(t)$ within the interval $[t_0, t_f]$, which satisfies all the imposed constraints, i.e. boundary condition (2.2.33)–(2.2.34), waypoints (2.2.35), bounded angles and thrust (2.2.40) and bounded angular velocities (2.2.41). In order to complete the trajectory part, we simply consider the zero-constant reference yaw angle trajectory $\psi_r = 0$ (even though the imposed constraints are still validated for an arbitrary yaw angle trajectory as discussed in Proposition 2.2.7) in order to keep the notations simple. Similar considerations are also used in various works related to trajectory generation and control of the multicopter system [Formentin and Lovera, 2011, Chamseddine et al., 2012, Eliker et al., 2018].

2.2.6 Simulation results for constrained trajectory generation

This section presents two simulation scenarios for the proposed constrained trajectory generation approach.

The first scenario considers an indoor trajectory generation over a nano-drone platform which

will be further employed in an experimental setup in Section 2.5.
The second scenario considers an outdoor trajectroy generation over a larger multicopter system, employed in the inspection of a 3D building. The particularity is that a specific yaw profile is imposed (such as the mounted camera is oriented towards the building at specific time instances).

2.2.6.1 Generic indoor trajectory generation

In this section, we generate a reference trajectory for the Crazyflie 2.0 quadcopter system (c.f. Figure 2.1.1), we consider its simulation model given in [Luis and Ny, 2016] characterized by the following parameters (the parameters employed in the dynamical model from (2.1.10)):

- the physical parameters as in (2.1.1), (2.1.9):

$$m = 0.028\ kg,\ J_x = J_y = 1.4 \times 10^{-5}\ kgm^2,\ J_z = 2.2 \times 10^{-5}\ kgm^2; \tag{2.2.47}$$

- the state constraints as in (2.1.12):

$$\epsilon_{\max} = 10^\circ,\ \omega_{\max} = 2\ \text{rad/s}; \tag{2.2.48}$$

- the input constraints as in (2.1.13):

$$T_{\max} = 0.55\ \text{N},\ \tau_{\max} = 10^{-4} \times [43\ 43\ 17]^\top\ \text{N/m}, \tag{2.2.49}$$

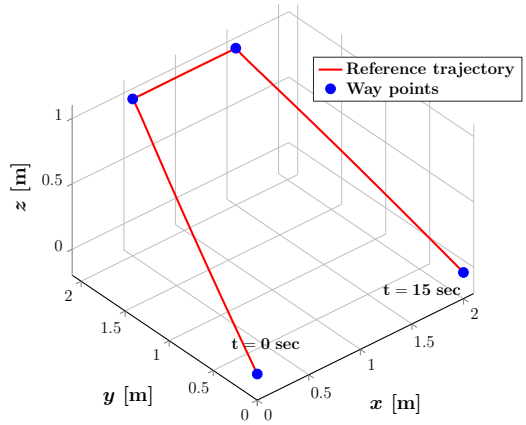
in which, the angle $\epsilon_{\max}$ is chosen to be small due to the limits of the built-in PID controllers (c.f. Figure 2.5.2) which are designed based on the approximation of the rotation dynamics (2.1.1)-(2.1.7) around the hovering condition (i.e., zero roll, pitch angles).
The first step is to choose the upper and lower bounds of the reference thrust $T_{r_{\min}}, T_{r_{\max}}$ and the boundary angle $\epsilon_{r_{\max}}$ satisfying the two consistent conditions (2.4.19)–(2.4.20). Introducing $\epsilon_{\max} = 10^\circ$ and $T_{\max} = 0.55$ from (2.2.48)–(2.2.49) to (2.4.19)–(2.4.20) leads to the feasible values of $T_{r_{\min}}, T_{r_{\max}}$ and $\epsilon_{r_{\max}}$ chosen as follows:

$$T_{r_{\min}} = 0.2609\ \text{N},\ T_{r_{\max}} = 0.2884\ \text{N},\ \epsilon_{r_{\max}} = 5^\circ, \tag{2.2.50}$$

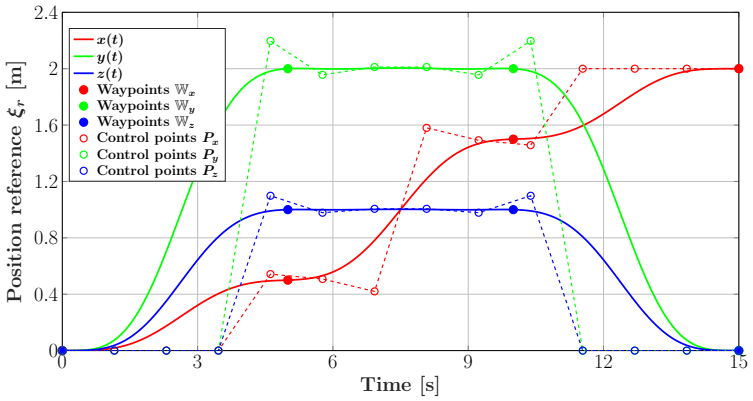
in which, $T_{r_{\min}} < mg < T_{r_{\max}}$ is validated with $mg = 0.2747\ N$.

	Constraints
Position, $\xi_r(t)$	$\xi_r(0) = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^\top$, $\xi_r(5) = \begin{bmatrix}0.5 & 2 & 1\end{bmatrix}^\top$, $\xi_r(10) = \begin{bmatrix}1.5 & 2 & 1\end{bmatrix}^\top$, $\xi_r(15) = \begin{bmatrix}2 & 0 & 0\end{bmatrix}^\top$
Boundary conditions, $\xi_r^{(q)}(t), \forall q \in \{1, \ldots, 4\}$	$\xi_r^{(q)}(0) = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^\top,\ \forall q \in \{1, \ldots, 4\}$, $\xi_r^{(q)}(15) = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^\top,\ \forall q \in \{1, \ldots, 4\}$
Angles, ϕ_r, θ_r	$\langle \lvert\phi_r\rvert, \lvert\theta_r\rvert \rangle \leq \epsilon_{r_{\max}}$ with $\epsilon_{r_{\max}} = 5^\circ$
Angular rates, $\omega_{x_r}, \omega_{y_r}$	$\langle \lvert\omega_{x_r}\rvert, \lvert\omega_{y_r}\rvert \rangle \leq \omega_{r_{\max}}$ with $\omega_{r_{\max}} = 1$ rad/s
Thrust, T_r	$T_{r_{\min}} \leq T_r \leq T_{r_{\max}}$ with $T_{r_{\min}} = 0.2609$ N, $T_{r_{\max}} = 0.2884$ N

Table 2.2.1: Constraints imposed on the optimal trajectory generation



(a) 3D reference trajectory passing through their waypoints.



(b) B-splines trajectories $x(t)$, $y(t)$, $z(t)$ and their control points.

Figure 2.2.3: Reference trajectory $\xi_r(t)$.

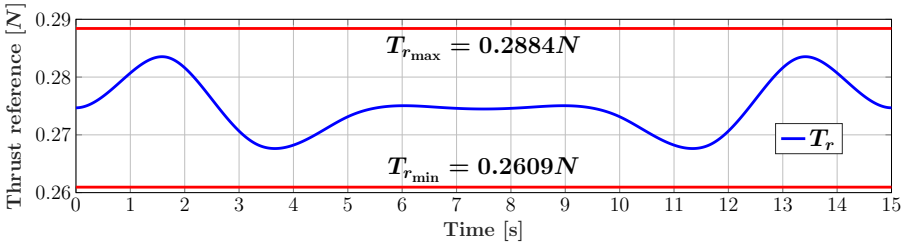
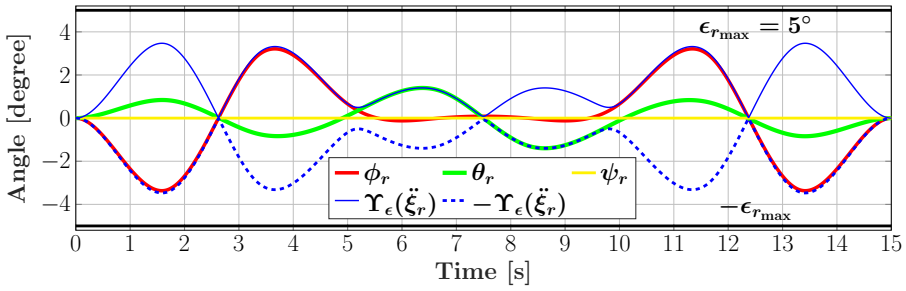


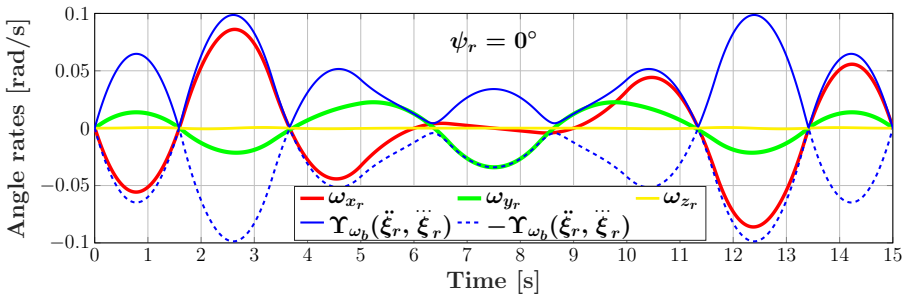
Figure 2.2.4: Thrust reference T_r in comparison with its lower and upper bounds $T_{r_{\min}}, T_{r_{\max}}$.

Next, the trajectory connects two hovering periods, thus, possess all the derivatives of $\{x, y, z\}$ up to 4^{th} order equal 0 at the initial and final time instants, i.e., $\xi_0^{(q)} = \xi_f^{(q)} = \mathbf{0}, \forall q \in \{1, \ldots, 4\}$ as employed in (2.2.33)–(2.2.34). It passes through four waypoints given by $\mathbb{W} = \left\{ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top, \begin{bmatrix} 0.5 & 2 & 1 \end{bmatrix}^\top, \begin{bmatrix} 1.5 & 2 & 1 \end{bmatrix}^\top, \begin{bmatrix} 2 & 0 & 0 \end{bmatrix}^\top \right\}$ with the associated time instants $\{0, 5, 10, 15\}$ seconds. All of the imposed constraints are gathered in Table 2.2.1.

The B-spline parametrization as in (2.2.28) is characterized by the chosen degree $d = 5$ and 14 control points ($n = 13$ from (2.2.27)). The number of the control points is required to be larger than the number of the hard constraints imposed on the trajectory generation problem: 4 waypoints, 6 boundary conditions. The trajectory generation optimization problem as in (2.2.46) are implemented using Yalmip [Löfberg, 2004] in Matlab 2015a with a total processing time of 9.4 seconds which provides the reference trajectory $\xi_r(t)$ and the results of the other variables corresponding to the choice of $\psi_r(t) = 0$ given in Figures 2.2.3-2.2.5. Furthermore, we also provide the references of the roll, pitch angles and angle rates when choosing $\psi_r(t) = 45°$ in Figure 2.2.6 to illustrate the effectiveness of Proposition 2.2.7. Next, as can be seen from Figure 2.2.3a, the reference trajectory $\xi_r(t)$ is extremely abrupt in the sense that the curve looks like three straight paths passing through four way-points but the values over time given in Figure 2.2.3 are still smooth. The references satisfy all the boundary conditions as clearly seen from the first and the last four control points having the same values for each curve (plotted in red for $x(t)$, green for $y(t)$ and blue for $z(t)$). Furthermore, the thrust as plotted in Figure 2.2.4 and the angles given in Figure 2.2.5a, all validate their constraints (2.2.36)–(2.2.37) thanks to the constraint formulation (2.2.40). The flatness properties of $\langle \phi_r, \theta_r \rangle \leq \Upsilon_\epsilon(\ddot{\xi}_r)$ from Proposition 2.2.4 and $\langle \omega_{x_r}, \omega_{y_r} \rangle \leq \Upsilon_{\omega_b}(\dddot{\xi}, \ddot{\xi})$ from Proposition 2.2.5 are illustrated in Figures 2.2.5 for the choice of $\psi_r(t) = 0°$ and in Figure 2.2.6 when choosing $\psi_r(t) = 45°$ in which the boundary functions $\Upsilon_\epsilon(\cdot)$ and $\Upsilon_{\omega_b}(\cdot)$ (solid blue lines) provide the good bounds for the corresponding



(a) References of the roll, pitch angles in comparison with the angle boundary $\Upsilon_\epsilon(\ddot{\xi}_r)$ from (2.2.15) when fixing $\psi_r(t) = 0°$.



(b) Reference angle rates ω_r in comparison with the angle rate boundary $\Upsilon_{\omega_b}(\ddot{\xi}_r, \dddot{\xi}_r)$ from (2.2.21) when fixing $\psi_r(t) = 0°$.

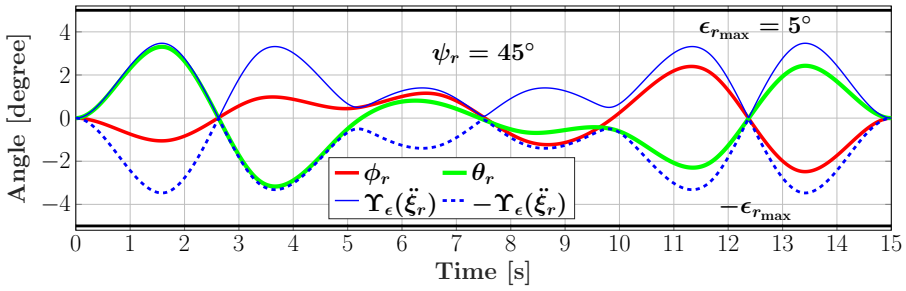
Figure 2.2.5: References of angles and angle rates when fixing $\psi_r(t) = 0°$.

reference variables (the angles ϕ_r, θ_r and the angle rates $\omega_{r_x}, \omega_{r_y}$ plotted in red, green solid lines in the two figures). Note that, the maximum value of the angle rates $\omega_{\max} = 2$ rad/s as in (2.2.48) is clearly respected, hence, not given in Figure 2.2.5b since it is too large to appear in the plot. From Figures 2.2.5–2.2.6, it has been shown clearly that the references of the angular variables are varied according to the choice of the yaw angle reference, however, they still respect their boundary functions, i.e., $\langle \phi_r, \theta_r \rangle \leq \Upsilon_\epsilon(\ddot{\xi}_r)$ and $\langle \omega_{x_r}, \omega_{y_r} \rangle \leq \Upsilon_{\omega_b}(\ddot{\xi}, \dddot{\xi})$ as given in Propositions 2.2.4–2.2.5. The reference trajectory given in Figure 2.2.3 and the zero yaw reference $\psi_r(t) = 0$ are used for conducting the tracking experiment over a real Crazyflie 2.0 nano drone as will being introduced in Section 2.5. In the following, we present another application of the proposed approach on the building inspection problem [Stoican et al., 2019].

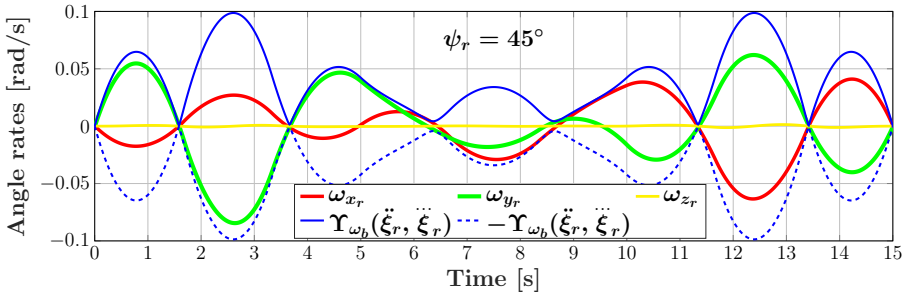
2.2.6.2 Outdoor trajectory generation for building inspection

In [Stoican et al., 2019], we employ the proposed approach to generate a reference trajectory which fully covers an a priori known 3D structure. The authors make use of hyperplane arrangement, cell merging procedures and mixed-integer formulations [Prodan et al., 2015] to provide feasible cells through which the trajectory has to pass (see also the details presented at (2.2.35)). Then, the minimum-length trajectory generation algorithm (2.2.46) provides the viewpoints (considered as way-points in the context of this chapter) which allows to generate the shortest trajectory fully inspecting the 3D structure. The readers are referred to [Stoican et al., 2019] for more details on the theoretical contributions, hereinafter we present only the simulation results related to the constrained trajectory generation. For inspecting the building of $100m \times 60m \times 60m$ (length $\times$ width $\times$ height), the multicopter platform and the reference trajectory are constrained by the following parameters:

- $m = 0.5\ kg$ as in (2.1.9);



(a) References of the roll, pitch angles in comparison with the angle boundary $\Upsilon_\epsilon(\ddot{\xi}_r)$ from (2.2.15) when fixing $\psi_r(t) = 45°$.



(b) Reference angle rates ω_r in comparison with the angle rate boundary $\Upsilon_{\omega_b}(\ddot{\xi}_r, \dddot{\xi}_r)$ from (2.2.21) when fixing $\psi_r(t) = 45°$.

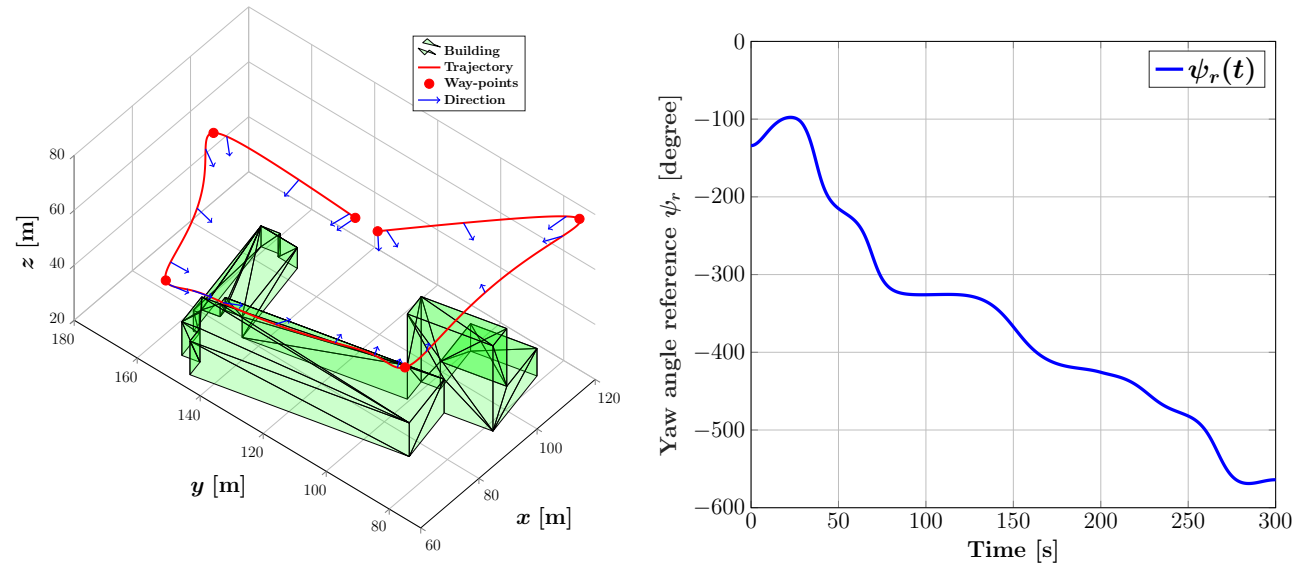
Figure 2.2.6: References of angles and angle rates when fixing $\psi_r(t) = 45°$.

- $\langle |\phi_r|, |\theta_r| \rangle \leq 15°$ as in (2.2.36), $\langle |\omega_{x_r}|, |\omega_{y_r}| \rangle \leq 0.27\ rad/s$ as in (2.2.38);
- $4.8\ N \leq T_r \leq 5\ N$ as in (2.2.37).
- 6 way-points:

$$\begin{aligned} \xi_r(0) &= \begin{bmatrix} 102.5 & 123 & 53 \end{bmatrix}^\top, & \xi_r(50) &= \begin{bmatrix} 120 & 75 & 75 \end{bmatrix}^\top, \\ \xi_r(100) &= \begin{bmatrix} 60 & 75 & 75 \end{bmatrix}^\top, & \xi_r(200) &= \begin{bmatrix} 60.74 & 151.6 & 53.33 \end{bmatrix}^\top, \\ \xi_r(250) &= \begin{bmatrix} 102.5 & 175 & 53.33 \end{bmatrix}^\top, & \xi_r(300) &= \begin{bmatrix} 102.5 & 130 & 53.33 \end{bmatrix}^\top. \end{aligned} \tag{2.2.51}$$

- $\psi_r(t)$ varying accordingly to the movement of the multicopter in order to point the camera (fixed to the vehicle) to the predefined targets.

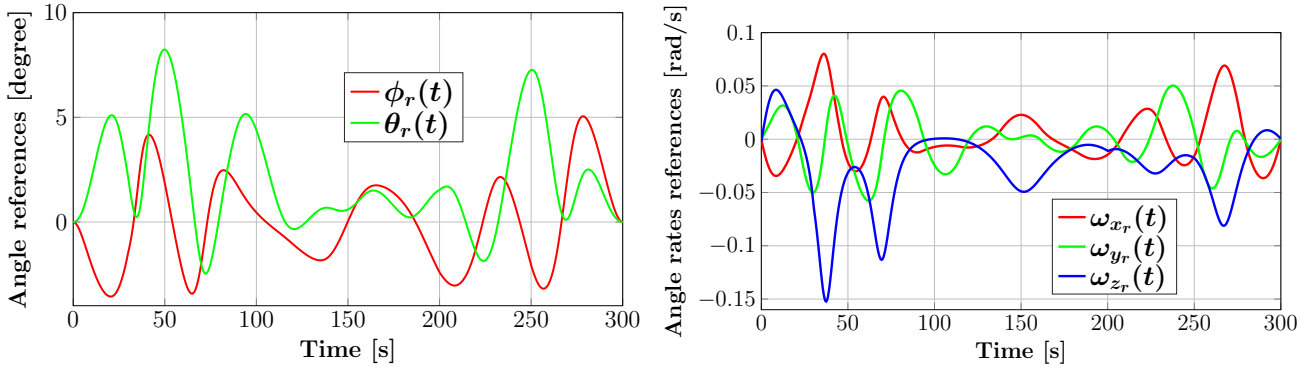
Note that, the platform does not require any constraints on the torques as in (2.1.13). Therefore, we neglect the rotation dynamics (2.1.1) but still consider the rotation kinematics given in (2.1.7). The results are given in Figures 2.2.7–2.2.9 with the calculation time of 20.4 seconds. In Figure 2.2.7a, the reference trajectory is plotted in solid red line with respect to the building given as green polytopic blocks. It passes through the predefined way-points (red circle marks) while trying to keep the connecting paths as short as possible. The yaw angle ψ_r varies over time as plotted in Figure 2.2.7b, therefore, any trajectory generation approaches employing the simplification of $\psi_r(t) = 0$ cannot be used under this scenario. Furthermore, as the benefit from the reliable constraints formulations (c.f. Proposition 2.2.7), the references of the roll, pitch angles $(\phi_r(t), \theta_r(t))$ (given in Figure 2.2.8a), the angle rates $(\omega_{x_r}(t), \omega_{y_r}(t))$ (given in Figure 2.2.8b) and the thrust T_r (given in Figure 2.2.9) satisfy the imposed constraints regardless the varying of the yaw angle.



(a) Reference trajectory passing through way-points.

(b) Time-varying yaw angle reference $\psi_r(t)$.

Figure 2.2.7: Reference trajectories $(\xi_r(t), \psi_r(t))$ which ensure full coverage of the building.



(a) Angle references.

(b) Angle rate references.

Figure 2.2.8: References of the angles η_r and the angle rates ω_r under building inspection scenario.

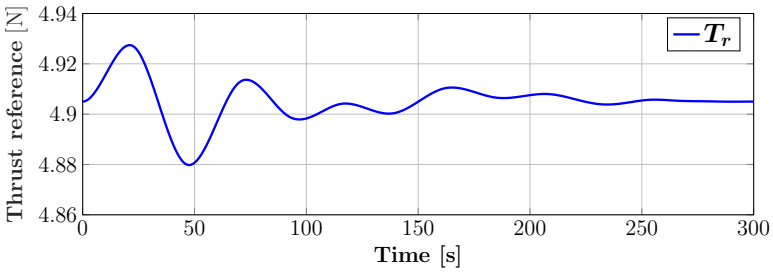


Figure 2.2.9: Thrust reference under building inspection scenario.

By this application on the trajectory generation for building inspection, we complete our contributions on the off-line trajectory generation problem for the multicopter system subject to state and input constraints. In the next section, the online tracking mechanism will be presented which makes use of a hierarchical control scheme (and also referred as cascade control design).

2.3 Hierarchical control design

The typical hierarchical control scheme for controlling the multicopter system is presented in Figure 2.3.1 (similar design can be found in [Formentin and Lovera, 2011, Freddi et al., 2011, Zhao and Go, 2014, Xu, 2017, Nguyen et al., 2017b, Nguyen et al., 2018b]). We consider the control problem of tracking the references of the 3D positions and the yaw (direction) angle, denoted by $\xi_r \in \mathbb{R}^3$ and $\psi_r \in \mathbb{R}$, respectively. For doing this, the scheme contains two control layers in which the designs exploit the natural decoupling between the rotation and translation dynamics of the multicopter system as shown in Figure 2.1.2.
At the high control level, the *position controller* compares the reference position ξ_r with the actual position ξ in order to provide the desired thrust T_d. It further requires the actual yaw value ψ for calculating the desired roll, ϕ_d, pitch, θ_d, angles. Then, they are gathered with the a priori given reference yaw angle ψ_r to create the reference angles for the *attitude controller* to track at the low control level. Based on the angle errors and the actual angle rate ω, the controller can provide the desired torques τ_d. Next, the real inputs u acting on the multicopter system as in (2.1.10) track their desired values $\mathsf{u}_d \triangleq [T_d \ \tau_d]^\top$ through a black box which stands for various uncertainties such as input saturation as in (2.1.13), rotor configuration and faults. Within the thesis, we consider full-state feedback control for the multicopter system (2.1.10) and neither delays nor mismatches on the state feedback are considered.
In the following, we detail the control design problems of the two layers and introduce two feedback linearization control candidates which are usually applied in the literature.

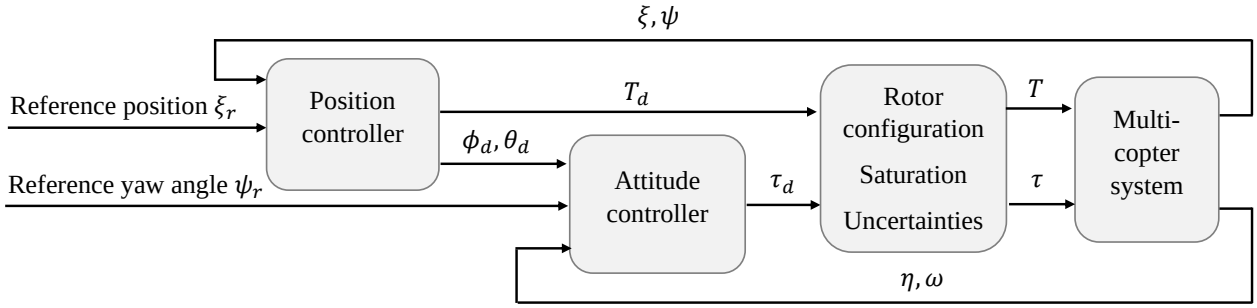


Figure 2.3.1: Hierarchical control scheme for a multicopter system.

2.3.1 High level control problem

The position controller as shown in Figure 2.3.1 drives the translation dynamics (2.1.9) to track the reference position ξ_r. This leads to an over-actuated problem since there are maximum four inputs, i.e., thrust T and three angles ϕ, θ, ψ, which can be employed for controlling the dynamics (2.1.9) while the output of the dynamics are only three positions gathered in $\xi \in \mathbb{R}^3$. In the literature, there are works which consider all of these four input variables, i.e., thrust and three angles, at the same time, in which the yaw angle input is assumed to be equal to the yaw reference ψ_r [Cowling et al., 2007, Mueller and D'Andrea, 2013, Lu et al., 2017]. Therefore, the

method relies on the yaw angle tracking capability of the attitude controller at the low control level (c.f. Figure 2.3.1) which actually cannot be well maintained due to difficulties in yaw angle measurement (in comparison with measuring the roll, pitch angles [Bazin et al., 2008]), changes in yaw reference during flight (e.g., for camera application as shown in Figure 2.2.7 or [Engelhardt et al., 2016]) and faults [Mueller and D'Andrea, 2014, Jiang et al., 2016, Nguyen et al., 2017b].
In order to avoid this unnecessary dependence on yaw tracking capability and the complicated over-actuated control design, we employ only the thrust T and roll, pitch angels ϕ, θ as the inputs of the dynamics (2.1.9) while the yaw angle ψ acts an external variable affecting the system. This decoupling design also receives a lot of attention in the research community [Formentin and Lovera, 2011, Freddi et al., 2011, Zhao and Go, 2014, Xu, 2017, Nguyen et al., 2017b]) due to its valuable advantage which is to allow us to compensate the mismatch on yaw angle tracking. For more details, from the aforementioned control perspective, the translation dynamics (2.1.9) is transformed into its state-space representation as follows:

$$\dot{\mathsf{p}} = \mathsf{f}_{\mathsf{p}}\left(\mathsf{p}, u, \psi\right), \tag{2.3.1}$$

where the state vector is $\mathsf{p} \triangleq [\xi \ v]^\top$ with ξ the position vector, $v = \dot{\xi}$ is the velocity vector from (2.1.10). The input vector $u \triangleq [T \ \phi \ \theta]$ consists of the thrust and the roll, pitch angles. The dynamical function $\mathsf{f}_{\mathsf{p}}\left(\mathsf{p}, u, \psi\right) \triangleq [v \ h(T, \eta)]^\top$ is appropriately taken from (2.1.10) where the function $h(\cdot)$ is explicitly given by:

$$h(T, \eta) = -g\mathsf{e}_3 + \frac{T}{m}\mathsf{R}\mathsf{e}_3 = -\begin{bmatrix}0\\0\\g\end{bmatrix} + \frac{T}{m}\begin{bmatrix}\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi\\ \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi\\ \cos\phi\cos\theta\end{bmatrix}, \tag{2.3.2}$$

with R the rotation matrix as in (2.1.5) and $\mathsf{e}_3 = [0\ 0\ 1]^\top$ from (2.1.9).
Furthermore, since the multicopter system is subject to its state and input constraints from (2.1.12)-(2.1.13), the input $u = [T \ \phi \ \theta]^\top$ also has to take into account the input saturation effects:

$$u = \mathrm{sat}(u_d, u_{\max}), \tag{2.3.3}$$

with $u_d \triangleq [T_d, \phi_d, \theta_d]$ the control law to be designed and $u_{\max} \triangleq [T_{\max} \ \epsilon_{\max} \ \epsilon_{\max}]$ from (2.1.12)–(2.1.13). The saturation function $\mathrm{sat}(\cdot)$ is firstly defined for two scalars, $x \in \mathbb{R}$ and its limit $x_{\max} \in \mathbb{R}_+$, as follows:

$$\mathrm{sat}(x, x_{\max}) = \mathrm{sign}(x)\max(|x|, x_{\max}), \tag{2.3.4}$$

and then, it is extended for two vectors as in (2.3.3), i.e., gathering the saturation functions for each pair of the elements: $\mathrm{sat}(u_d, u_{\max}) \triangleq [\mathrm{sat}(T_d, T_{\max}) \ \mathrm{sat}(\phi_d, \epsilon_{\max}) \ \mathrm{sat}(\theta_d, \epsilon_{\max})]^\top$.

2.3.2 Low level control problem

Recall the hierarchical control scheme given in Figure 2.3.1. The attitude controller tracks the desired angle vector denoted by η_d which gathers the desired roll, pitch angles ϕ_d, θ_d provided from the high control level and the reference yaw ψ_r sent from the user:

$$\eta_d \triangleq [\phi_d \ \theta_d \ \psi_r]^\top. \tag{2.3.5}$$

Furthermore, it is well-known in the literature that the low level is required to execute at faster frequency than of the controller at the high level in order to establish the stability of the whole

scheme [Scattolini, 2009, Bemporad et al., 2009, Freddi et al., 2011]. Thus, from the perspective of the low control level, the desired roll, pitch angles (ϕ_d, θ_d) obtained from the high control level are usually considered as step references. Many related works in the literature also avoid taking into account the variation of these references due to large noises potentially introduced to the controlled system under practical considerations [Cao and Lynch, 2016]. Thus, within this thesis, the role of the attitude controller is to stabilize the rotation dynamics (2.1.1)–(2.1.7) around the desired equilibrium point η_d from (2.3.5) at each step.
Moreover, considering the system constraints (2.1.12)–(2.1.13), the control design problem within the attitude controller is to provide the torque law τ_d such that when applying the real torque τ calculated by:

$$\tau = \text{sat}(\tau_d, \tau_{\max}), \tag{2.3.6}$$

with $\tau_{\max}$ the maximum torque value as in (2.1.13), to the rotation dynamics (2.1.1)–(2.1.7), the torque τ stabilizes the dynamics around the desired angle η_d from (2.3.5). Furthermore, the resulted angles and angular velocities are required to satisfy $\langle|\phi|, |\theta|\rangle \leq \epsilon_{\max}$, $\langle|\omega_x|, |\omega_y|\rangle \leq \omega_{\max}$ as delineated in (2.1.12).
Indeed, this control design problem subject to state and input constraints is convoluted. Thus, in the literature, when considering the rotation controller, both the input saturation as in (2.3.6) and the state constraints from (2.1.12) are usually neglected [Formentin and Lovera, 2011, Mellinger and Kumar, 2011, Roza and Maggiore, 2012, Carino et al., 2015, Nguyen et al., 2017b]. Even though these probably lead to undesired and even unstable motions, the obtained control designs are worth considering in order to obtain an insight on the aforementioned sophisticated control problem. Therefore, we present in the following a standard computed-torque control design for the attitude controller. The design is given under the assumption of no state and input constraints.

Computed-torque control for attitude control:
Computed-torque control (CTC) is a special application of feedback linearization control particularized for a broad range of robotics systems [Craig, 2005]. These systems are also referred as "computed-torque like" systems and admit the Lagrangian dynamical models [Lewis et al., 2003]. Some typical examples are industrial robot arms and also the rotation dynamics (2.1.1)–(2.1.7) which is transformed into its "computed-torque like" formulation by using $\omega = W\dot{\eta}$ from (2.1.7):

$$JW\ddot{\eta} + J\dot{W}\dot{\eta} + (W\dot{\eta}) \times (JW\dot{\eta}) = \text{sat}(\tau, \tau_d), \tag{2.3.7}$$

with $J = \text{diag}\{J_x, J_y, J_z\}$ the inertial tensor and the matrix W from (2.1.8). The CTC design presented in [Craig, 2005] provides the desired torque τ_d given by:

$$\tau_d \triangleq \tau_{\text{CTC}} = JW\mu_\eta + J\dot{W}\dot{\eta} + (W\dot{\eta}) \times (JW\dot{\eta}), \tag{2.3.8}$$

in which, J is the inertial tensor and $W, \eta, \dot{\eta}$ are obtained as feedback from the system (2.3.7). The virtual input $\mu_\eta \in \mathbb{R}^3$ is designed using the well-known PD control method:

$$\mu_\eta = \ddot{\eta}_d + K_\eta e_\eta + K_{\dot{\eta}} e_{\dot{\eta}}, \tag{2.3.9}$$

with $e_\eta = \eta - \eta_d$, $e_{\dot{\eta}} \triangleq \dot{e}_\eta = \dot{\eta} - \dot{\eta}_d$ the errors on the angles and the angle derivatives. Note that, the reference terms $(\eta_d, \dot{\eta}_d, \ddot{\eta}_d)$ are taken and derived from the desired angle signal η_d. The control gain matrices are chosen as $K_\eta = \text{diag}\{K_\phi, K_\theta, K_\psi\}$ and $K_{\dot{\eta}} = \text{diag}\{K_{\dot{\phi}}, K_{\dot{\theta}}, K_{\dot{\psi}}\}$ in which all the control gains are strictly negative to ensure the closed-loop stability. Thus, by introducing $\tau = \tau_{\text{CTC}}$ as in (2.3.7)–(2.3.8) (i.e., no consideration on the input saturation) to the dynamics (2.3.7), we obtain the following error dynamics:

$$\ddot{e}_\eta + K_{\dot{\eta}}\dot{e}_\eta + K_\eta e_\eta = \mathbf{0}, \tag{2.3.10}$$

which can be made asymptotically stable by appropriate choices of the tuning parameters according to Routh-Hurwitz criterion [Craig, 2005].

Remark 2.3.1. The torque design using CTC method as in (2.3.8) implies several robustness problems especially when considering model mismatches, measurement delays and system constraints [Egeland, 1986]. Therefore, a more advanced control technique must be applied instead. We will show in the next chapter that an optimization-based controller with the capability of guaranteeing the imposed state and input constraints (2.1.12)-(2.1.13) can be derived from the knowledge on the CTC controller (2.3.8). More precisely, the CTC controller (2.3.8) will act only as a feasibility guarantee for the optimization problem and its applying region will be limited around the desired equilibrium point, hence, ensuring the satisfaction of the system constraints. Note also that, the PD control design as employed in (2.3.9) is just a simple choice for illustration purpose. Various control techniques such as nested control design [Teel, 1992], LQR (Linear Quadratic Regulator) [Kuantama et al., 2018], etc, can be applied to control the resulted linear error dynamics (2.3.10). □

In the following, we present a control candidate for the position controller at the high level. Not like the CTC attitude controller given in (2.3.8), the design fully satisfies the control problem presented in Section 2.3.1, i.e., ensuring the trajectory tracking capability under input constraints. The design is actually a feedback linearization controller which makes use of the flatness properties of the translation dynamics as presented in Section 2.2.1, thus, being reliable even under tracking mismatch or intentional change of the predefined yaw angle.

2.4 Position control through feedback linearization via flatness

For controlling the dynamics (2.3.1), there exists a feedback linearization control candidate which is facilitated by the differential flatness property of the multicopter system introduced in Section 2.2.1. The controller is denoted by $u_{\text{FL}}(\mu_\xi, \psi) \triangleq [T_{\text{FL}}(\mu_\xi) \quad \phi_{\text{FL}}(\mu_\xi, \psi) \quad \theta_{\text{FL}}(\mu_\xi, \psi)]^\top$ in which $T_{\text{FL}}(\mu_\xi) \triangleq \Upsilon_T(\mu_\xi)$, $\phi_{\text{FL}}(\mu_\xi, \psi) \triangleq \Upsilon_\phi(\mu_\xi, \psi)$ and $\theta_{\text{FL}}(\mu_\xi, \psi) \triangleq \Upsilon_\theta(\mu_\xi, \psi)$ with $\Upsilon_T(\cdot)$ from (2.2.11), $\Upsilon_\phi(\cdot)$ from (2.2.5) and , $\Upsilon_\theta(\cdot)$ from (2.2.6) are the flatness representations of thrust and roll, pitch angles, respectively. Explicitly, the elements are given as follows:

$$T_{\text{FL}}(\mu_\xi) = m\sqrt{\mu_x^2 + \mu_y^2 + (\mu_z + g)^2}, \tag{2.4.1a}$$

$$\phi_{\text{FL}}(\mu_\xi, \psi) = \arcsin\left(\frac{\mu_x \sin\psi - \mu_y \cos\psi}{\sqrt{\mu_x^2 + \mu_y^2 + (\mu_z + g)^2}}\right), \tag{2.4.1b}$$

$$\theta_{\text{FL}}(\mu_\xi, \psi) = \arctan\left(\frac{\mu_x \cos\psi + \mu_y \sin\psi}{\mu_z + g}\right), \tag{2.4.1c}$$

with $\mu_\xi \triangleq [\mu_x \ \mu_y \ \mu_z]^\top$ the virtual input vector of the dynamics (2.3.1). Even though the feedback linearization law (2.4.1) has been employed widely in the literature [Formentin and Lovera, 2011, Freddi et al., 2011, Zhao and Go, 2014, Xu, 2017, Nguyen et al., 2017b]), most of the works do not notice its restriction. To the best of our knowledge, the limitation was first discussed in [Nguyen et al., 2018b] and will be recapitulated hereinafter.

Proposition 2.4.1 ([Nguyen et al., 2018b]). *The feedback law* $u_{FL}(\mu_\xi, \psi)$ *given by* (2.4.1) *drives the dynamics* $\dot{\mathsf{p}} = \mathsf{f}_\mathsf{p}(\mathsf{p}, u_{FL}(\cdot), \psi)$ *from* (2.3.1) *to one of the two following systems depending on*

the value of the virtual input μ_z*:*

$$\begin{cases} \dot{x} = v_x, \ \dot{v}_x = \mu_x \\ \dot{y} = v_y, \ \dot{v}_y = \mu_y \\ \dot{z} = v_z, \ \dot{v}_z = \mu_z \end{cases} \quad \textit{(equivalent to } \ddot{\xi} = \mu_\xi\textit{)}, \ \textit{if } \mu_z \geq -g, \tag{2.4.2}$$

$$\begin{cases} \dot{x} = v_x, \ \dot{v}_x = -\cos(2\psi)\mu_x - \sin(2\psi)\mu_y \\ \dot{y} = v_y, \ \dot{v}_y = -\sin(2\psi)\mu_x + \cos(2\psi)\mu_y \\ \dot{z} = v_z, \ \dot{v}_z = -\mu_z - 2g \end{cases} , \ \textit{if } \mu_z < -g. \tag{2.4.3}$$

Proof. We provide here the sketch of the proof while further details can be found in [Nguyen et al., 2018b]. Let us consider the altitude dynamics exploited from (2.3.1) given as:

$$m\dot{v}_z = -mg + T\cos\phi\cos\theta. \tag{2.4.4}$$

Since $T\cos\phi\cos\theta \geq 0$, $\dot{v}_z + g$ always has a non-negative value. Thus, introducing (2.4.1) into the dynamics (2.3.1) leads to $\dot{v}_z + g = |\mu_z + g|$. Consequently, $\dot{v}_z = \mu_z$ if $\mu_z \geq -g$ and $\dot{v}_z = -\mu_z - 2g$ if $\mu_z < -g$. Introducing them back to the system (2.3.1) leads to the corresponding results of $\dot{v}_x$ and $\dot{v}_y$. The relation $v = \dot{\xi}$ is not affected. □

The dynamical system (2.4.2) is usually the desired goal of using the feedback law u_{FL} from (2.4.1) in various works [Nguyen et al., 2017b, Zhao and Go, 2014, Formentin and Lovera, 2011]. However, if the condition of $\mu_z \geq -g$ is not validated, the undesired dynamics (2.4.3) may occur which probably leads to unstable behavior. Furthermore, it requires the desired input $u_d = u_{\text{FL}}(\mu_\xi, \psi)$ from (2.4.1) not to be affected by the saturation constraint $u = \text{sat}(u_d, u_{\max})$ (2.3.3), hence, providing exactly $u = u_{\text{FL}}(\cdot)$ for the closed-loop system $\dot{\mathsf{p}} = \mathsf{f}_{\mathsf{p}}(\mathsf{p}, u, \psi)$ from (2.3.1) to be perfectly feedback linearized. Therefore, we introduce another result published in [Nguyen et al., 2018b] related to the construction of an input constraint admissible set for the feedback linearization law $u_{\text{FL}}(\cdot)$ from (2.4.1).

Proposition 2.4.2 ([Nguyen et al., 2018b])**.** *By choosing the values of the three positive saturation limits* U_x*,* U_y *and* U_z *such that:*

$$U_z < g, \tag{2.4.5}$$

$$U_x^2 + U_y^2 \leq (-U_z + g)^2 \tan^2 \epsilon_{max}, \tag{2.4.6}$$

$$m\sqrt{U_x^2 + U_y^2 + (U_z + g)^2} \leq T_{max}, \tag{2.4.7}$$

with ϵ_{max}, T_{max} *from* (2.1.12)-(2.1.13)*, the following holds. If the virtual inputs* μ_ξ *from* (2.4.1) *satisfy*[3]*:*

$$|\mu_\xi| \leq U_\xi, \tag{2.4.8}$$

with $U_\xi \triangleq [U_x \ U_y \ U_z]^\top$*, then, the feedback linearization input* $u_{FL}(\mu_\xi, \psi)$ *from* (2.4.1) *satisfies the system constraints, i.e.* $|u_{FL}(\cdot)| \leq u_{max}$ *from* (2.3.3)*, for all values of the yaw angle* $\psi \in \mathbb{R}$*.* □

Proof. At first, (2.4.5) is to ensure the achievement of the desired dynamics (2.4.2). Then, let us apply the results of Proposition 2.2.4 to $\phi_{\text{FL}} = \Upsilon_\phi(\mu_\xi, \psi)$ and $\theta_{\text{FL}} = \Upsilon_\theta(\mu_\xi, \psi)$ from (2.4.1b)-(2.4.1c):

$$\langle |\phi_{\text{FL}}(\mu_\xi, \psi)|, |\theta_{\text{FL}}(\mu_\xi, \psi)| \rangle \leq \Upsilon_\epsilon(\mu_\xi) = \arctan\left(\sqrt{\frac{\mu_x^2 + \mu_y^2}{(\mu_z + g)^2}}\right), \tag{2.4.9}$$

[3] In (2.4.8), $|\mu_\xi| \leq U_\xi$ is used to express the triplet of $|\mu_x| \leq U_x$, $|\mu_y| \leq U_y$ and $|\mu_z| \leq U_z$ with $U_\xi \triangleq [U_x \ U_y \ U_z]^\top$ from (2.4.5)–(2.4.7).

with $\Upsilon_\epsilon(\cdot)$ the angle boundary from (2.2.15). Next, by introducing $|\mu_x| \leq U_x$, $|\mu_y| \leq U_y$ and $|u_z| \leq U_z$ with $U_z < g$ from (2.4.5) to the final term of (2.4.9) and to $T_{\text{FL}}(\mu_\xi)$ from (2.4.1a), we have that:

$$\langle |\phi_{\text{FL}}(\mu_\xi, \psi)|, |\theta_{\text{FL}}(\mu_\xi, \psi)| \rangle \leq \arctan\left(\sqrt{\frac{U_x^2 + U_y^2}{(-U_z + g)^2}}\right), \ \forall \psi \in \mathbb{R}, \tag{2.4.10}$$

$$T_{\text{FL}}(\mu_\xi) \leq m\sqrt{U_x^2 + U_y^2 + (U_z + g)^2}. \tag{2.4.11}$$

Therefore, by introducing the two first conditions (2.4.5)-(2.4.6) to (2.4.10) and the last one (2.4.7) to (2.4.11), we obtain $\langle |\phi_{\text{FL}}(\cdot)|, |\theta_{\text{FL}}(\cdot)| \rangle \leq \epsilon_{\max}$ and $T_{\text{FL}}(\mu_\xi) \leq T_{\max}$, hence, completing the proof. □

2.4.1 Virtual input design using nested-control method

By Proposition 2.4.2, we need to ensure the bound of the virtual inputs, i.e., $|\mu|_{\mathsf{p}} \leq U_\xi$ from (2.4.8) in order to guarantee the feedback linearization property of $u_{\text{FL}}(\cdot)$ as shown in (2.4.2). In the literature, a popular candidate for guaranteeing the input saturation constraints is nested-control method [Teel, 1992] which will be employed for designing the virtual inputs $|\mu|_{\mathsf{p}}$ hereinafter.

Proposition 2.4.3 ([Nguyen et al., 2018b]). *Let us consider the design of the virtual input vector* $\mu_\xi \triangleq [\mu_x \ \mu_y \ \mu_z]$ *from* (2.4.1) *given by:*

$$\mu_\xi = \ddot{\xi}_r + \text{sat}\left(K_1\dot{e}_\xi + \text{sat}\left(-K_1K_2e_\xi + K_2\dot{e}_\xi, \ \frac{\lambda}{2}\right), \lambda\right), \tag{2.4.12}$$

where $e_\xi = \xi - \xi_r$ *represents the position tracking error and* $K_1, K_2 \in \mathbb{R}^{3\times3}$ *the control gain matrices required to be diagonal and negative definite. The saturation function* $\text{sat}(\cdot)$ *is defined in* (2.3.4) *while the limit* $\lambda \in \mathbb{R}^3_+$ *is given by:*

$$\lambda = U_\xi - \left|\ddot{\xi}_r\right|, \tag{2.4.13}$$

where $U_\xi \in \mathbb{R}^3_+$ *from Proposition 2.4.1 are chosen (beside the conditions* (2.4.5)-(2.4.7)*) such that:*

$$U_\xi > \max_t \left|\ddot{\xi}_r(t)\right|. \tag{2.4.14}$$

Then, the design in (2.4.12) *satisfies* $|\mu_\xi| \leq U_\xi$ *(required by Proposition 2.4.2). Furthermore, introducing the controller* $u_d := u_{FL}(\mu_\xi, \psi)$ *with* μ_ξ *from* (2.4.12) *to the system* $\dot{\mathsf{p}} = \mathsf{f}_{\mathsf{p}}(\mathsf{p}, u, \psi)$ *with* $u = \text{sat}(u_d, u_{max})$ *given in* (2.3.1), (2.3.3) *leads to globally asymptotically stable error dynamics in terms of the position tracking error* e_ξ. □

Proof. Applying the Triangle inequality [Meyer, 2000] to the virtual input μ_ξ from (2.4.12) leads to:

$$|\mu_\xi| \leq \left|\ddot{\xi}_r\right| + \lambda = U_\xi, \tag{2.4.15}$$

with λ defined as in (2.4.13). Then, from Proposition 2.4.2, we have that $|u_{\text{FL}}(\mu_\xi, \psi)| \leq u_{\max}$ with μ_ξ as in (2.4.12) and $u_{\max}$ from (2.3.3). Thus, introducing the controller $u_d := u_{\text{FL}}(\mu_\xi, \psi)$ to the dynamics $\dot{\mathsf{p}} = \mathsf{f}_{\mathsf{p}}(\mathsf{p}, u, \psi)$ with $u = \text{sat}(u_d, u_{\max})$ given in (2.3.1), (2.3.3) leads to the following system:

$$\ddot{e}_\xi = \text{sat}\left(K_1\dot{e}_\xi + \text{sat}\left(-K_1K_2e_\xi + K_2\dot{e}_\xi, \ \frac{\lambda}{2}\right), \lambda\right), \tag{2.4.16}$$

with $e_\xi = \xi - \xi_r$ from (2.4.12). The error dynamics (2.4.16) is obtained from the dynamics (2.4.2) and is globally asymptotically stable for any negative definite and diagonal matrices K_1 and K_2 following the nested-control design method introduced in [Teel, 1992] and recapitulated in Appendix C. □

Remark 2.4.4. Instead of applying the nested control design as in (2.4.12), the usual PD control can also be employed for designing the virtual input μ_ξ from (2.4.1) as follows:

$$\mu_\xi = \ddot{\xi}_r + K_\xi e_\xi + K_v \dot{e}_\xi, \tag{2.4.17}$$

with $\ddot{\xi}_r$ the reference acceleration, e_ξ the position tracking error as in (2.4.12). The control gain matrices $K_\xi \in \mathbb{R}^{3\times3}$ and $K_v \in \mathbb{R}^{3\times3}$ are chosen to be diagonal and negative definite as similar to (2.4.12). Then, the closed-loop error dynamics are locally asymptotically stable. This is due to the condition $|\mu_\xi| \leq U_\xi$ from Proposition 2.4.2 which can only be satisfied locally where the following condition holds:

$$|\ddot{\xi}_r + K_\xi e_\xi + K_v e_v| \leq U_\xi. \tag{2.4.18}$$

Note that, (2.4.18) can be enforced by designing a positive invariant set in which $|K_\xi e_\xi + K_v e_v| \leq U_\xi - \max_t |\ddot{\xi}_r(t)|$ is guaranteed (note that, the condition $U_\xi > \max_t |\ddot{\xi}_r(t)|$ as in (2.4.14) is required again). These results will be introduced in Chapter 4 where a constraint admissible and positive invariant set is essential for the design of an optimization-based controller. □

2.4.2 Consistency of the constraints on thrust and roll, pitch angles

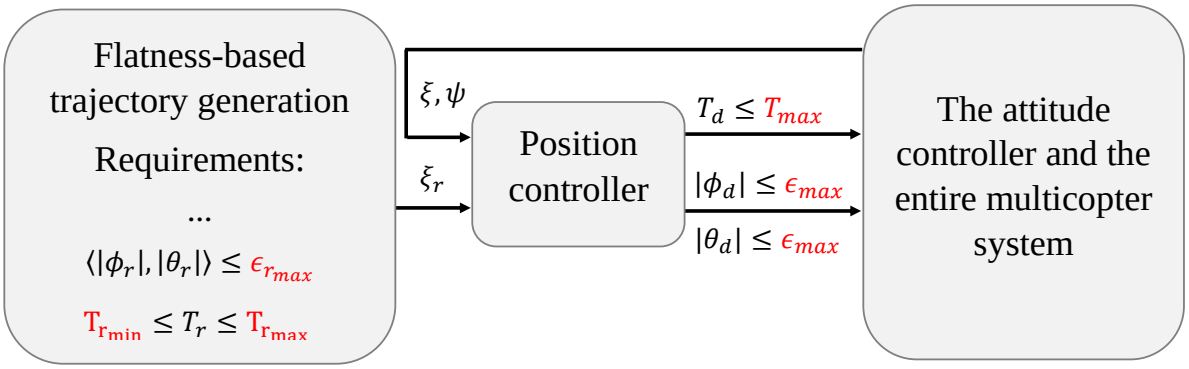


Figure 2.4.1: Flatness-based trajectory generation process and the position controller sharing similar constraints on thrust and roll, pitch angles.

Let us recall the saturation limits $U_\xi = [U_x\ U_y\ U_z]$ from Proposition 2.4.2 which are required to satisfy the conditions (2.4.5)–(2.4.7) and also (2.4.14) (note that, (2.4.14) is employed not only for the nested control design but also the PID design as in (2.4.18)). This raises a question on the existence of these saturation limits or in other words, on the consistency between the related variables: $m, g, \epsilon_{\max}, T_{\max}$ (representing the physical parameters of the multicopter system) employed in (2.4.5)–(2.4.7) and $\ddot{\xi}_r$ (representing the reference trajectory) used in (2.4.14). In the following, we provide the sufficient condition for the existence of U_ξ satisfying (2.4.5)–(2.4.7) and (2.4.14). Furthermore, via the answer, we propose a condition for choosing the parameters of the reference trajectory such that it allows to design the tracking controller $u_{\text{FL}}(\mu_\xi, \psi)$ as in (2.4.1) subject to the limits of the multicopter system.

Proposition 2.4.5 ([Nguyen et al., 2018b]). *A sufficient condition for the existence of* U_x, U_y, U_z *satisfying* (2.4.5)–(2.4.7) *and* (2.4.14) *is that the reference boundary angle* $\epsilon_{r_{max}}$ *from* (2.2.36) *and the reference thrust limits* $T_{r_{min}}, T_{r_{max}}$ *from* (2.2.37)*, all employed within the trajectory generation problem* (2.2.46)*, have to satisfy:*

- *If* $T_{r_{min}} \cos \epsilon_{r_{max}} + T_{r_{max}} < 2mg$ *holds, then:*

$$\sqrt{2} \frac{T_{r_{max}}}{T_{r_{min}}} \tan \epsilon_{r_{max}} < \tan \epsilon_{max}, \tag{2.4.19}$$

$$\sqrt{(2mg - T_{r_{min}} \cos \epsilon_{r_{max}})^2 + 2T_{r_{max}}^2 \sin^2 \epsilon_{r_{max}}} < T_{max}, \tag{2.4.20}$$

- *otherwise, if* $T_{r_{min}} \cos \epsilon_{r_{max}} + T_{r_{max}} \geq 2mg$ *holds, then:*

$$T_{r_{max}}(1 + \sqrt{2} \operatorname{cotan}(\epsilon_{max}) \sin \epsilon_{r_{max}}) \leq 2mg, \tag{2.4.21}$$

$$T_{r_{max}} \sqrt{1 + \sin^2 \epsilon_{r_{max}}} \leq T_{max}, \tag{2.4.22}$$

with m *the system mass,* g *the gravity,* ϵ_{max} *and* T_{max} *the constraint parameters of the multicopter system given in* (2.1.12)–(2.1.13). □

Proof. The proof is explicitly given in Appendix D and also in [Nguyen et al., 2018b]. Note that, in [Nguyen et al., 2018b], the condition $T_{r_{\min}} \cos \epsilon_{r_{\max}} + T_{r_{\max}} < 2mg$ is enforced by constraining $T_{r_{\min}} + T_{r_{\max}} = 2mg$, hence, the second case does not appear there. We also emphasize that the domain where $T_{r_{\min}} \cos \epsilon_{r_{\max}} + T_{r_{\max}} < 2mg$ holds is usually the operating zone of the standard multicopter system (near hovering), the other case is only applied when considering extremely aggressive movements (e.g., requiring very large thrust) as in [Landry et al., 2016]. □

By Proposition 2.4.5, we link the constraints shown in Figure 2.4.1 i.e., $T_{r_{\min}} \leq T_r \leq T_{r_{\max}}$ as in (2.2.37) and $\langle |\phi_r|, |\theta_r| \rangle \leq \epsilon_{r_{\max}}$ imposed on the trajectory generation problem (2.2.46) with their actual constraints $T_d \leq T_{\max}$ and $\langle |\phi_d|, |\theta_d| \rangle \leq \epsilon_{\max}$ considered within the design of the position controller. Therefore, we create a unified design scheme for trajectory generation and tracking with bounded thrust and bounded angles while respecting the physical constraints of the system.

As a summary, this section introduces the design of the position controller using the feedback linearization controller $u_{\text{FL}}(\mu_\xi, \psi)$ from (2.4.1) with μ_ξ the virtual control input designed as in (2.4.17). Based on the properties of the flatness-based representations of the system, the feedback linearization controller can ensure the saturation guarantees as detailed in Proposition 2.4.2. The nested control design of the virtual inputs μ_ξ as in (2.4.12) allows the closed-loop system to overcome the unstable mode as in (2.4.3) and to achieve the global asymptotic stability for tracking the reference trajectory $\xi_r(t)$. The existence of the control design parameters (i.e., the saturation limit U_ξ from Proposition 2.4.2) is enforced by constraining the reference angle limit $\epsilon_{r_{\max}}$, the lower and upper bounds of thrust $T_{r_{\min}}, T_{r_{\max}}$, employed for generating the trajectory, to satisfy the conditions (2.4.19)-(2.4.20).

Next section will present the validation of the proposed feedback linearization position controller through simulation and experimental results over a real quadcopter platform.

2.5 Simulation and experimental results for trajectory tracking

This section firstly introduces the experimental platform using the Crazyflie 2.0 (CF) nano-drone equipped at the Laboratory of Conception and Integration of System (LCIS), in Valence,

France. Then, we show the experimental results for tracking the reference trajectory generated as in Section 2.2.6 by using the hierarchical control scheme detailed in Section 2.3.

2.5.1 Experimental platform

The experimental platform at laboratory LCIS shown in Figure 2.5.1 includes an indoor CF quadcopter equipped with a 10-DOF Inertial Measurement Unit (accelerometer, gyro, magnetometer, and high-precision pressure sensor) and a motion tracking system called the Loco positioning system[4] including a deck attached to the CF quadcopter and six nodes fixed around the experimental room with known positions. The Loco positioning system is functioning in Two Way Ranging (TWR) mode in which, the deck pings the nodes in sequence. This allows the deck to obtain the distance between itself and the six nodes, then, the deck can calculate its position compared to the six nodes. After gathering all the necessary feedback information such as the position and the angles using the embedded software, the CF quadcopter communicates with the ground station computer through a 2.4Ghz low-latency/long-range radio messages by using the Crazyradio PA USB radio dongle. The computer sends an input message including the four inputs $\{T_d, \phi_d, \theta_d, \dot{\psi}_r\}$ to the CF quadcopter with the maximum communicating frequency of 100 Hz. Note that the thrust input T_d after being calculated by the desired control law (e.g., the feedback linearizaion controller T_{FL} as in (2.4.1a)) is required to be converted into the thrust unit of the CF quadcopter system, i.e. 16-bit integer valued from 0 to 65535. Furthermore, since the Loco positioning system provides only the position of the CF, its velocity is estimated by using a Kalman filter in which the required noise information is measured beforehand.

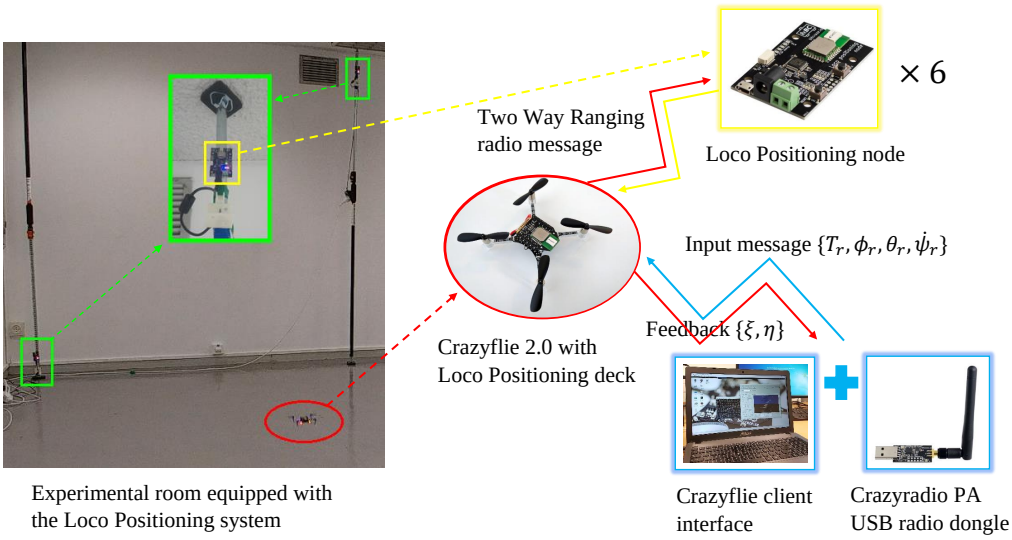


Figure 2.5.1: Experiment setup using the Crazyflie 2.0 nano-quadcopter.

Figure 2.5.2 shows the hierarchical control scheme of the CF in which the built-in controller controls the four rotors (by sending PWM signals) to track the desired set-point of thrust T_d, roll and pitch angles, ϕ_d, θ_d, and reference yaw rate $\dot{\psi}_r$. The built-in controllers of CF contain two loops [Luis and Ny, 2016, Giernacki et al., 2017]:

i) an attitude PID controller which compares the desired angles, and the real angles received as the feedback from CF, then, provides the references of the roll and pitch angle rates;

[4]More information of the Crazyflie quadcopter and the Loco positioning system can be found in https://www.bitcraze.io

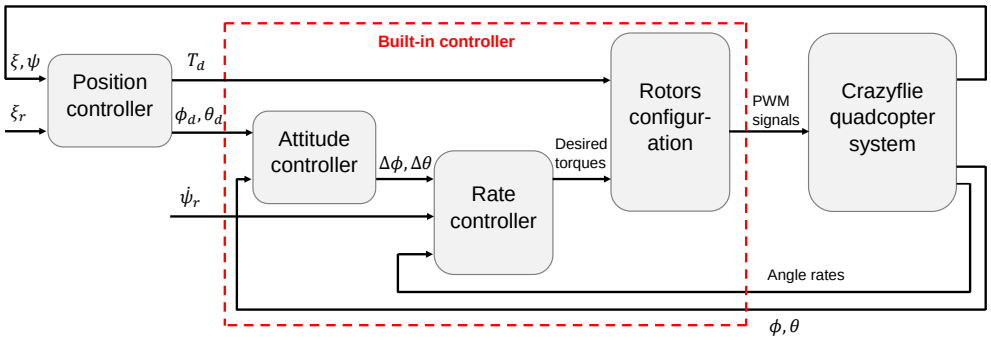


Figure 2.5.2: Hierarchical control scheme with the built-in controller of the Crazyflie quadcopter system.

ii) a PID rate controller which compares the rate references included the foregoing yaw rate input $\dot{\psi}_r$, and the real angle rates obtained from CF quadcopter in order to calculate the torques.

Finally, the torques and the desired thrust input T_d are transformed into the four rotor speeds by using the appropriate configuration (i.e. X configuration for the CF quadcopter [Luis and Ny, 2016]).
Note that, for experimental validation, we only apply the position controller detailed in Section 2.3.1 and directly employ the built-in controller without any modification while for simulation, we employ the CTC controller given in (2.3.8) (with noticing all of its drawbacks) in order to obtain the whole controlled multirotor system.

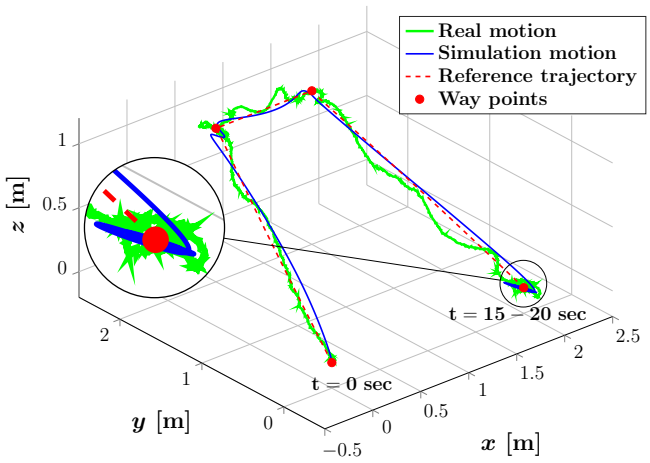
2.5.2 Simulation and experiment results for trajectory tracking

In this section, we consider the simulation scenario of tracking the a priori given reference trajectory $\xi_r(t)$ as shown in Figure 2.2.3b during the first 15 seconds, then, hovering at the final position $\xi_r(15) = [2\ 0\ 0]^\top$ for 5 seconds. Under simulation, we validate both the position and attitude controller detailed in Sections 2.3.1–2.3.2 while only the position controller $u_{\text{FL}}(\mu_\xi, \psi)$ as in (2.4.1), (2.4.12) is tested under real experiment due to the technical limitation of the platform (as shown in Figure 2.5.2). Note that, the drone actually starts and finishes the trajectory with the two hovering periods. After obtaining the experimental results, we move the coordinate such that the first hovering position becomes the zero point.
We show in Table 2.5.1 the tuning parameters of the feedback linearization controller $u_{\text{FL}}(\mu_\xi, \psi)$ as in (2.4.1), (2.4.12) and the computed-torque attitude controller τ_{CTC} from (2.3.8). The saturation limits $U_\xi = [U_x\ U_y\ U_z]^\top$ are chosen to satisfy the conditions (2.4.5)–(2.4.7) and also $U_\xi > \max_t |\ddot{\xi}_r(t)|$ as in (2.4.14) with $\max_t |\ddot{\xi}_r(t)| = [0.2396\ 0.5948\ 0.2974]^\top$ obtained from the reference trajectory $\xi_r(t)$ in Figure 2.2.3b. We choose $U_x = U_y$ and maximize the value of U_z since we focus more on the convergence of the altitude while equally treat the motions along x and y axes.

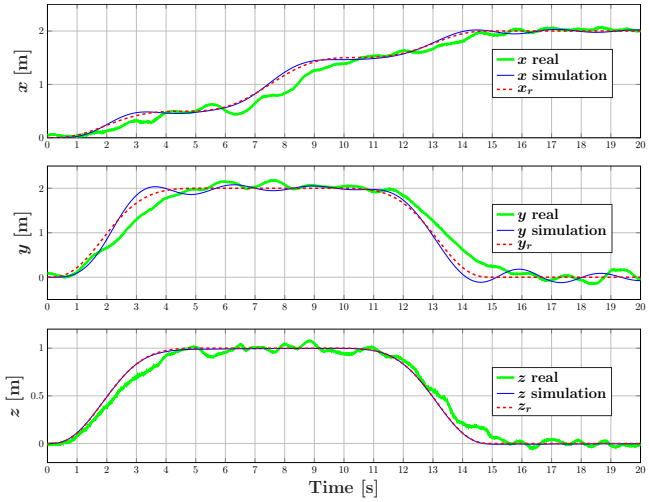
The tracking motions of the CF quadcopter using the position controller $u_{\text{FL}}(\mu_\xi, \psi)$ are shown in Figures 2.5.3a-2.5.3b in which the reference trajectories $\xi_r = [x_r\ y_r\ z_r]^\top$ are plotted in dashed red lines while the simulation and experimental results are illustrated by a thin blue line and thick green line, respectively. Both tracks well the reference during the time interval of $[0, 15]$ seconds and also provides good hovering capability as can be seen during the last 5 seconds (3D motions of hovering scenario can be seen from the magnifying inset in Figure 2.5.3a).
Next, the desired values of thrust T_d obtained from the feedback linearization position controller

	Parameter	Value
Position controller $u_{\text{FL}}(\mu_\xi, \psi)$ from (2.4.1), (2.4.12)	U_ξ as in (2.4.13)	$[0.7168\ 0.7168\ 2.5970]^\top$
	K_1 as in (2.4.12)	$-\operatorname{diag}\{1, 1, 2\}$
	K_2 as in (2.4.12)	$-\operatorname{diag}\{2, 2, 3\}$
Attitude controller τ_{CTC} from (2.3.8)	K_η as in (2.3.9)	$-\operatorname{diag}\{10, 10, 4\}$
	$K_{\dot\eta}$ as in (2.3.9)	$-\operatorname{diag}\{25, 25, 4\}$

Table 2.5.1: Parameters of the position and attitude controllers given in (2.4.1) and (2.3.8).



(a) Motions of the CF quadcopter under simulation and experiment.



(b) Tracking results along three axes.

Figure 2.5.3: Position tracking results along the three axes of the feedback linearization position controller $u_{\text{FL}}(\mu_\xi, \psi)$ given in (2.4.1),(2.4.12) [Nguyen et al., 2018b].

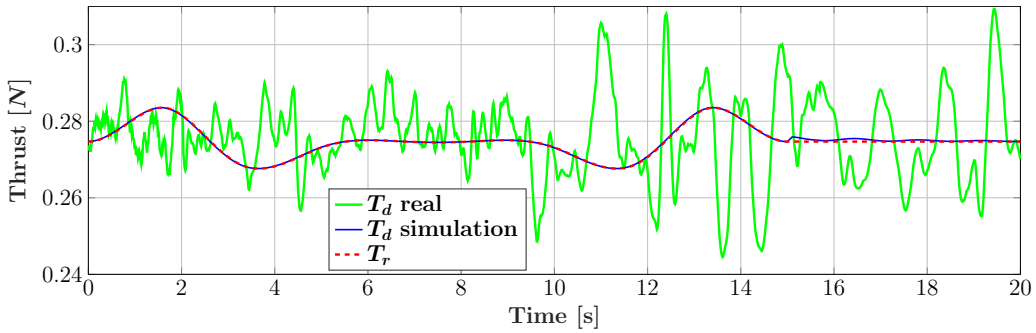


Figure 2.5.4: The desired thrust values T_d provided by the feedback linearization position controller $u_{\text{FL}}(\mu_\xi, \psi)$ given in (2.4.1),(2.4.12) with respect to the thrust reference T_r [Nguyen et al., 2018b].

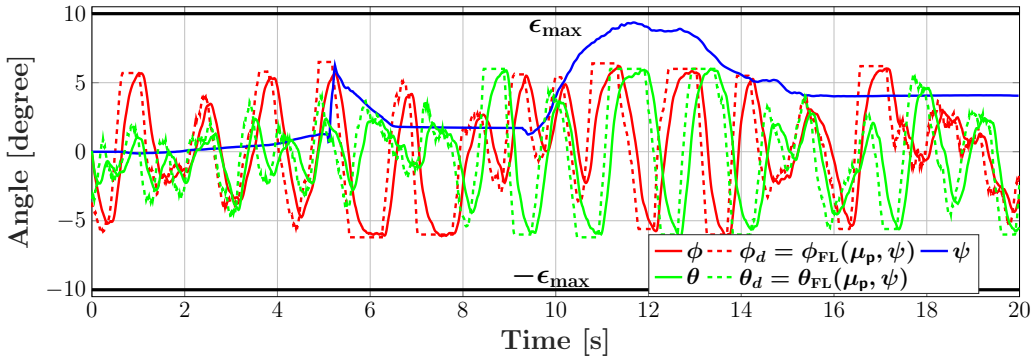


Figure 2.5.5: Three actual Euler angles (ϕ, θ, ψ) with respect to their desired values: ϕ_d, θ_d obtained from the feedback linearization position controller $u_{\text{FL}}(\mu_\xi, \psi)$ given in (2.4.1),(2.4.12) and $\dot{\psi}_r(t) = 0$ under experiment [Nguyen et al., 2018b].

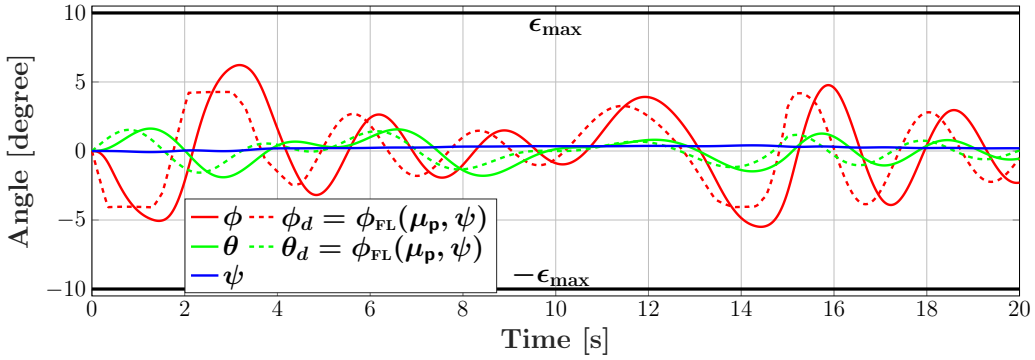


Figure 2.5.6: Three Euler angles (ϕ, θ, ψ) under simulation with respect to their desired values: ϕ_d, θ_d obtained from the feedback linearization position controller $u_{\text{FL}}(\mu_\xi, \psi)$ given in (2.4.1), (2.4.12) and $\psi_r(t) = 0$ [Nguyen et al., 2018b].

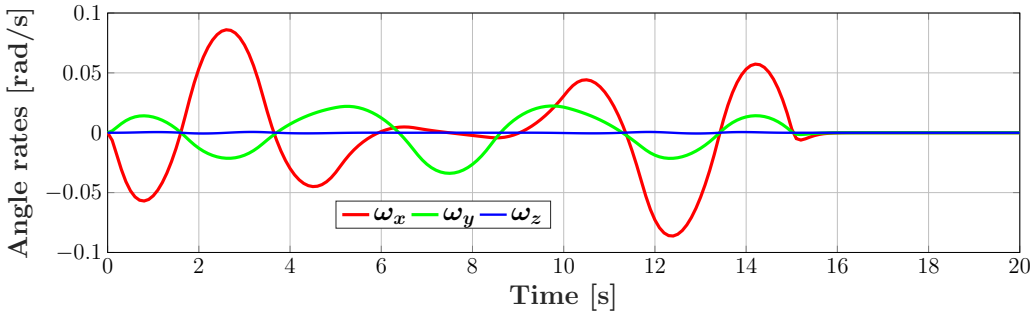


Figure 2.5.7: Angle rates $(\omega_x, \omega_y, \omega_z)$ under simulation using the CTC attitude controller given in (2.3.8).

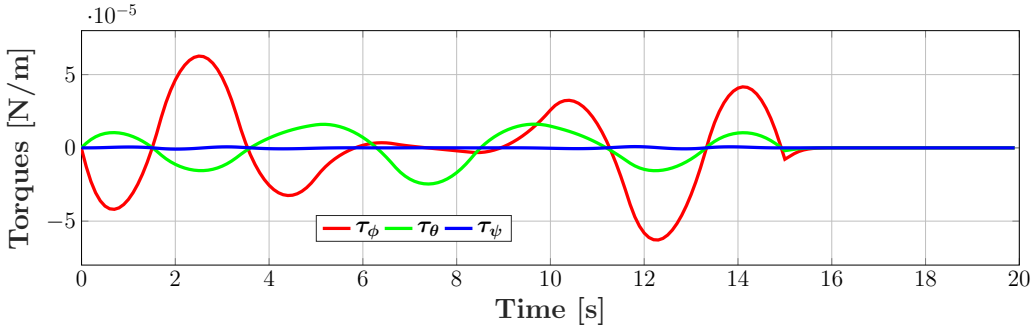


Figure 2.5.8: Torques under simulation using the CTC attitude controller given in (2.3.8).

$u_{\text{FL}}(\mu_\xi, \psi)$ from (2.4.1) are given in Figure 2.5.4 for both experiment (plotted in green line) and simulation (plotted in blue line). The simulation thrust follows well the reference T_r while the experiment thrust oscillates around the reference with large amplitude. This is due to the limited precision of the Loco positioning system (c.f. Figure 2.5.1) of around 10 centimeters, hence, even during the hovering period in the last five seconds, the position feedback is still varying significantly as can be seen from Figure 2.5.3b. Note that, all the signals respect the thrust limit $T_{\max} = 0.55\ N$ which is hidden from the plot to enhance the clarity.
The position controller $u_{\text{FL}}(\mu_\xi, \psi)$ from (2.4.1) also provides the desired roll, pitch angles (ϕ_d, θ_d) which are given in Figure 2.5.5 for the experimental scenario and in Figure 2.5.6 for simulation. From Figure 2.5.5, it can be observed that the actual yaw angle ψ plotted in solid blue line varies significantly and does not follow the predefined reference $\psi_r(t) = 0$. This is due to the fact that we intentionally give $\dot{\psi}_r = 0$ as the input to the CF system, thus, we can examine the validation of the reliable constraints $\langle \phi_{\text{FL}}(\mu_\xi, \psi), \theta_{\text{FL}}(\mu_\xi, \psi) \rangle \leq \epsilon_{\max},\ \forall \psi \in \mathbb{R}$ from (2.4.10). And as can be seen from Figure 2.5.5, even under this unpredicted varying yaw angle ψ, the controller $u_{\text{FL}}(\mu_\xi, \psi)$ from (2.4.1) is still capable of providing ϕ_d (dashed red line) and θ_d (dashed green line) which respect the maximum angle $\epsilon_{\max} = 10°$, and furthermore, ensures the trajectory tracking capability as mentioned before. Note that, since the yaw angle ψ has been changed from its reference $\psi_r = 0$, the desired roll, pitch angles (ϕ_d, θ_d) do not follow their references (ϕ_r, θ_r) given in Figure 2.2.5a.
Next, under simulation, the CTC attitude controller $\tau_{\text{tiny CTC}}$ from (2.3.8) tracks well the desired roll, pitch angles (ϕ_d, θ_d) as observed in Figure 2.5.6. It further provides the angle rates $(\omega_x, \omega_y, \omega_z)$ and the torques $(\tau_\phi, \tau_\theta, \tau\psi)$ shown in Figures 2.5.7-2.5.8 which clearly respect the corresponding constraints, i.e., $\langle |\omega_x|, |\omega_y| \rangle \leq \omega_{\max}$ with $\omega_{\max} = 1\ rad/s$ from (2.2.48) and $|\tau| \leq \tau_{\max}$ with $\tau_{\max} = 10^{-4}[43\ 43\ 17]^\top\ N/m$. Note that, these constraint validations are mostly due to the fact that the reference trajectory is already feasible for the system, i.e., it validates all the aforementioned constraints as can be seen from Figures 2.2.5a-2.2.5b. The CTC attitude controller does not possess any mechanism for guaranteeing these constraints.

2.6 Concluding remarks and open questions

This chapter presented the fundamental tools for solving the motion planning problem for multicopter systems. These will also be used to tackle further extensions. The overall process can be summarized as follows:

i) examine the system dynamics in order to derive its flatness-based representation;

ii) generate off-line a feasible reference trajectory by using flatness and a specific parametrization - the B-splines curves;

iii) design an online tracking mechanism.

Following the aforementioned steps, this chapter firstly addresses the dynamical model of a standard multicopter system including two subsystems: rotation and translation dynamics with their corresponding constraints. Then, the flatness property is introduced which allows us to formulate some reliable angular constraint formulations. These constraints help not only guaranteeing the constraints under change of a predefined yaw angle trajectory but also provide the ease to counteract the control design problem with respect to the uncertainties on yaw angle tracking. Next, we parametrize the flatness representation of the system by using B-spline curves which provide the ease to formulate the minimal-length trajectory generation problem into a standard optimization problem with a quadratic cost.

Then, for online tracking control design, we employ the standard hierarchical control scheme consisting of two layers due to the natural decoupling of the system dynamics. The *position controller* at the high level compensate the position errors by providing the desired values of the thrust and the roll, pitch angles. These desired angles are tracked by the *attitude controller* at the low level. The two control candidates are introduced into the scheme:

i) at the high level, the feedback linearization controller via flatness $u_{\text{FL}}(\mu_\xi, \psi)$ (with μ_ξ the virtual input employing nested control method) is employed;

ii) at the low level, the computed-torque controller τ_{CTC} ensures the angle tracking.

The controller u_{FL} is capable of globally guaranteeing the thrust and angle constraints even under the uncontrolled yaw motion while the controller τ_{CTC} is employed locally where all the constraints are assumed to be satisfied.
The whole process is successfully validated through simulations. Furthermore, by using a real experimental platform consisting of a Crazyflie 2.0 nano-quadcopter and a position estimation system, the position controller u_{FL} shows its capability of counteracting the mismatch on yaw angle tracking and of guaranteeing all the imposed constraints. The attitude controller is not implemented for a real test due to the non-modifiable built-in controller of the platform.
The results are promising, however, they still raise some questions:

1. The feedback linearization controller $u_{\text{FL}}(\mu_\xi, \psi)$ from (2.4.1), (2.4.12) can guarantee the input constraints, but at the price of providing needlessly conservative values which causes significant loss of efficiency. Furthermore, it requires a sophisticated design process (i.e., the feedback law $u_{\text{FL}}(\mu_\xi, \psi)$ from (2.4.1), the nested control design μ_ξ from (2.4.12), the saturation limits U_ξ from Proposition 2.4.2 and the consistency conditions from Proposition 2.4.5). Hence, the following question arises: can we employ another control candidate which can still guarantee the imposed constraints but allow a less complicated design procedure?

2. The CTC controller τ_{CTC} can only be used locally where all the system constraints are not violated. Even though this restriction can be counteracted by generating a feasible reference before applying the controller, it still limits the operation of the system when considering uncertainties.

These open problems will be answered in the following chapters. We firstly answer the second problem in the next chapter since this is more general than the particular case of the translation dynamics. In order to overcome the difficulties, we propose the design of an optimization-based controller with stability induced by the existence of a computed-torque control law of the considered system. The proposed theory will be applied for designing the attitude controller at the low level of the control scheme.

Chapter 3

Attitude control through NMPC with guaranteed stability

The attitude control of aerial vehicles (usually considered as 3D rigid bodies) has raised much interest in both the research and the aerospace industrial communities [Chaturvedi et al., 2011, Intwala and Parikh, 2015] due to the growth of various applications such as path/trajectory tracking or inspection and surveillance using drones including commercial fly-cams. The attitude controller is an important element of the control process since it stabilizes the vehicle at the desired angle set-point received from the high control level (c.f. the hierarchical control scheme of the multicopter system given in Figure 2.3.1). While conceptually simple, there are interesting intricacies due to two reasons:

i) the angle rate vector ω from (2.1.1) is not directly integrated into the attitude but through a nonlinear transformation $\omega = W\dot{\eta}$ as in (2.1.7);

ii) the attitude, as characterized by the three Euler angles η from (2.1.5), implies the existence of singularities occurring at some specific orientations (e.g., upright position).

Even though the second point ii) can be easily overcame by using the quaternion representation [Nguyen et al., 2020a, Carino et al., 2015, Tayebi and McGilvray, 2006], we still employ the Euler angles representation since it has been widely used in various aerospace applications and furthermore, has been applied to numerous existing commercial platforms (e.g., the Crazyflie nano-drone as shown in Figure 2.5.1). A practical reason for this is the equivalent between the yaw angle and the vehicle's direction [Nex and Remondino, 2014, Intwala and Parikh, 2015]. Therefore, in order to avoid the singularities, it is essential to take into account all the state constraints as in (2.1.12) and also the torque constraint as in (2.1.13) when designing the controller. However, various attitude control applications in the literature following the same reasoning only assume that the controlled system will operate in the admissible range without any necessary enforcement [Cowling et al., 2007, Freddi et al., 2011, Mueller and D'Andrea, 2014, Nguyen et al., 2017b]. In view of these shortcomings, we propose in this chapter an NMPC (Nonlinear Model Predictive Control) controller for tackling the attitude control problem under state and input constraints since the MPC approach is well-known for its capability of easily handling various constraints with a standard design while still providing good control performance [Mayne et al., 2000].

MPC is a control strategy in which, at each time step, a finite horizon open-loop optimal control problem, subject to (linear or nonlinear) system dynamics and constraints on states (including the current state as the initial condition) and inputs, is solved to obtain an optimal control sequence. From the sequence, only the control action in the first sampling time interval is applied to the system (considered here in the continuous-time domain). At the next sampling

instant, the state is measured again and introduced into the optimization problem. The process is iteratively executed to establish the closed-loop controlled system. Thanks to its capacity to handle various constraints, MPC has become a popular control candidate for both research and industrial purposes [Mayne et al., 2000, Badgwell and Qin, 2015]. However, there are still various remaining issues on MPC design, such as heavy computation requirements, stability and feasibility. Recently, the consequence of the computational burden has been reduced by various research works on fast-solving methods [Alamir and Murilo, 2008, Badgwell and Qin, 2015], more powerful solvers [Houska et al., 2011, Wächter and Biegler, 2006] and advancing processors/microprocessors technologies [Mayne, 2014]. As a result, it has already become possible to employ MPC method for aerospace applications [Marchand and Alamir, 2003, Rucco et al., 2015, Reinhardt and Johansen, 2019, Nguyen et al., 2019b], well-known for their strict timing demands.

Furthermore, it is important to mention that the stability and feasibility issues need to be taken into account from the beginning of the design process, hence, avoiding the possibility of leading to infeasible solutions during execution or even to instability. For tackling these issues, two additional ingredients in the MPC scheme are reported in the literature: a terminal cost and a terminal constraint set (besides the standard indispensable stage cost). This design is facilitated by the existing designing rules presented in [Mayne et al., 2000] which revolve around an important ingredient , usually hidden from the MPC scheme: a local controller. The term "local" refers to the fact that it is locally input constraint admissible. The terminal constraint set is usually positive invariant and both state and input constraints are admissible within the set under this local controller. Therefore, it is required to define explicitly the local controller and then, to construct the corresponding constraint admissible positive invariant set in order to apply the method. Bear in mind that the local controller is only used to design the ingredients of the MPC controller and furthermore, it has less advantages in comparison with the MPC one. I.e., the MPC controller allows the system to start from an initial state outside the terminal constraint set (only the final predicted state is required to stay within the set) while using only the local controller implies that the operating zone should be restricted within the set (in order to respect all the system's constraints and to achieve the positive invariant property). Furthermore, the MPC strategy allows the controlled system to efficiently make use of its power (as the input is only restricted by its constraints) while the local controller has to follow its defined formulation. Last but not least, using a local controller not only guarantees stability but also functions as an initial guess for the optimal control, to be found by the optimization problem. Thus, while nominally the local control is never used, it can still function as a fail-back if the optimization problem cannot provide an optimal results within the predefined solving time.

In the literature, almost all related NMPC (nonlinear MPC, i.e., using nonlinear prediction model) applications employ a linear local controller [Chen and Allgöwer, 1998, Mayne et al., 2000, Cannon et al., 2003, Kohler et al., 2018] due to the complexity of the nonlinear controlled systems. However, employing a linear controller for a general nonlinear system obviously restricts the corresponding invariant set (also serving as the terminal region), reducing the efficiency of the associated NMPC controller. In view of these shortcomings and also motivated by our pursuit of a more appropriate local controller for the NMPC design of robotics systems, the CTC (Computed-Torque Control) law appears as a promising candidate which hopefully provides a larger invariant set and a better insight into the system's behavior (w.r.t. the set associated to the linear controller). Indeed, one way to classify robotic control schemes is to divide them into "computed-torque-like" or "non-computed-torque-like" [Lewis et al., 2003]. There is a broad range of systems employing a "computed-torque-like" controllers, such as, aerospace crafts, industrial robot arms and mobile robots [Poignet and Gautier, 2000, Ferrara et al., 2013, Uebel et al., 1992] among which are the rotation dynamics given in (2.1.11) and which are the main

control objective in this chapter. Thus, by exploiting the CTC law, we propose several contributions related to improvements for the NMPC design, which are, to the best of our knowledge, new to the state of the art:

1. An application of the CTC law to the existing NMPC design principles with guaranteed stability. The CTC law allows to construct an ellipsoidal invariant set (serving as the terminal region) associated with the closed-loop linear dynamics. It also guarantees the input constraint validation within this set;

2. An upper bound is provided for the weighted norm of the CTC controller in terms of the corresponding state within the terminal region. This is obtained using Taylor's approximation for the CTC controller;

3. An explicit formulation of the terminal region is provided in terms of the design parameters, thus, the NMPC design is capable of easily modifying (e.g., re-orientating or enlarging) the proposed terminal region.

4. An NMPC design for attitude control with stability guaranteed by the existing CTC law of the rotation dynamics which is obtained by applying the proposed theorem.

This chapter is organized as follows. Section 3.1 presents the principles of NMPC design with stability guaranteed by using a terminal invariant set. Next, in Section 3.2, we first introduce a "computed-torque like" system and continue by detailing the constraint satisfaction, the construction of an invariant set and the input boundedness under the computed-torque controller. These ingredients will be combined to formulate an NMPC design in Section 3.3 which satisfies all the general principles, and hence, guarantees the system's closed-loop stability. The performance and advantages of the design are illustrated through various examples on an inverted pendulum system and a comparison with the classical quasi-infinite horizon NMPC design [Chen and Allgöwer, 1998] in Section 3.4. Next, an NMPC design for the attitude control of a multicopter system is addressed in Section 3.5. We apply the proposed method to the existing compute-torque controller of the rotation dynamics with several important improvements on reducing the complexity of the design and finally obtain an NMPC controller for angle set-point tracking. Section 3.5.2 presents the simulation results on tracking the flatness-based reference angle trajectory obtained in the previous chapter.

3.1 Principles of NMPC design with terminal stabilizing constraints

In this section, we recapitulate the design principles of an NMPC controller using both a terminal cost and a terminal constraint set. The principles have their origin in [Chen and Allgöwer, 1998] and are summarized in [Mayne et al., 2000].
Specially, the terminal set is required to be constraint admissible and positive invariant under a predefined local controller. Then, the NMPC design enforces the terminal predicted state of the present prediction horizon to stay within the set which ensures the feasibility of the next NMPC iteration. Furthermore, the closed-loop stability is established by choosing the terminal cost such that the optimal cost value becomes a Lyapunov function [Mayne et al., 2000].
The approach considers a nonlinear system given as follows:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}), \tag{3.1.1}$$

where the function $f(\cdot)$ admits one unique solution for a given initial condition. The control problem is to stabilize the system (3.1.1) around its equilibrium denoted by $(\boldsymbol{x}_e, \boldsymbol{u}_e)$ which satisfies the standard equilibrium definition:

$$f(\boldsymbol{x}_e, \boldsymbol{u}_e) = \mathbf{0}. \tag{3.1.2}$$

The state $\boldsymbol{x} \in \mathbb{R}^n$ and input $\boldsymbol{u} \in \mathbb{R}^m$, both subject to constraints:

$$\boldsymbol{x} \in \mathcal{X},\ \boldsymbol{u} \in \mathcal{U}, \tag{3.1.3}$$

with $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{U} \subseteq \mathbb{R}^m$. For the problem to be well-defined, $\mathcal{X} \times \mathcal{U}$ also contains the equilibrium point $(\boldsymbol{x}_e, \boldsymbol{u}_e)$ from (3.1.2).
The NMPC optimization problem for stabilizing the dynamics (3.1.1) at the equilibrium point $(\boldsymbol{x}_e, \boldsymbol{u}_e)$ at time instant t, using the known state $\boldsymbol{x}(t)$ is defined as follows:

$$\min_{\bar{\boldsymbol{u}}_t(\cdot)} J(\boldsymbol{x}(t), \bar{\boldsymbol{u}}_t(\cdot)), \tag{3.1.4}$$

with

$$J(\boldsymbol{x}(t), \bar{\boldsymbol{u}}_t(\cdot)) = \int_t^{t+T_p} \ell(\bar{\boldsymbol{x}}_t(s), \bar{\boldsymbol{u}}_t(s))ds + F(\bar{\boldsymbol{x}}_t(t+T_p)), \tag{3.1.5}$$

subject to

$$\dot{\bar{\boldsymbol{x}}}_t = f(\bar{\boldsymbol{x}}_t, \bar{\boldsymbol{u}}_t), \tag{3.1.6a}$$
$$\bar{\boldsymbol{x}}_t(s) \in \mathcal{X},\ \bar{\boldsymbol{u}}_t(s) \in \mathcal{U},\ s \in [t, t+T_p], \tag{3.1.6b}$$
$$\bar{\boldsymbol{x}}_t(t) = \boldsymbol{x}(t), \tag{3.1.6c}$$
$$\bar{\boldsymbol{x}}_t(T_p) \in \mathcal{X}_f, \tag{3.1.6d}$$

with $T_p \in \mathbb{R}_+$ the prediction horizon. Furthermore, $\bar{\boldsymbol{x}}_t(s)$ and $\bar{\boldsymbol{u}}_t(s)$ represent the predicted state and input at time s ($t \leq s \leq t+T_p$) while $\bar{\boldsymbol{u}}_t(\cdot)$ as in (3.1.4)-(3.1.5) stands for the whole predicted input trajectory along the prediction horizon length $[t, t+T_p]$, all corresponding to the optimization problem (3.1.4) at time t. Next, the stage cost $\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is a positive definite function satisfying $\ell(\boldsymbol{x}_e, \boldsymbol{u}_e) = 0$ and $\ell(\boldsymbol{x}, \boldsymbol{u}) > 0,\ \forall (\boldsymbol{x}, \boldsymbol{u}) \in \mathcal{X} \times \mathcal{U} \backslash \{\boldsymbol{x}_e, \boldsymbol{u}_e\}$. The terminal cost $F : \mathcal{X} \to \mathbb{R}$ (also required to be positive definite, i.e., satisfying $F(\boldsymbol{x}_e) = 0$ and $F(\boldsymbol{x}) > 0\ \forall \boldsymbol{x} \in \mathcal{X} \backslash \{\boldsymbol{x}_e\}$) and the terminal region $\mathcal{X}_f \subseteq \mathbb{R}^n$ are both chosen to satisfy the design principles presented hereinafter.
Assuming that at time t, the optimization problem (3.1.4)–(3.1.6) provides the optimal input denoted by $\bar{\boldsymbol{u}}_t^*(s)$ with $s \in [t, t+T_p]$, then, the control action applied to the system (3.1.1) during the time interval $[t, t+\delta]$ is defined as follows:

$$\boldsymbol{u}_{\text{MPC}}(s) = \bar{\boldsymbol{u}}_t^*(s),\ \forall s \in [t, t+\delta], \tag{3.1.7}$$

where the NMPC sampling time $0 < \delta < T_p$ is chosen depending on the specification requirements of the considered system. More precisely, for a real implementation, it needs to be larger than the computing time of the optimization problem (3.1.4)-(3.1.6) so that there remains time for other control processes such as state measurement or estimation.
According to the MPC design classification presented in [Mayne et al., 2000], the NMPC problem proposed in (3.1.4)-(3.1.6) belongs to the category in which both terminal cost and constraint set are employed. For this category, [Chen and Allgöwer, 1998, Mayne et al., 2000] also provide four design conditions, that, if satisfied, ensure the recursive feasibility[1] and the closed-loop asymptotic (exponential) stability of the scheme. These are summarized in the followings.

[1]The initial iteration successfully executed implies the feasibility of all the further steps [Chen and Allgöwer, 1998].

Lemma 3.1.1 ([Mayne et al., 2000], page 799)**.** *Let us consider the NMPC controller* (3.1.4)-(3.1.7) *for stabilizing the system* $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$ *from* (3.1.1) *around its equilibrium point* $\boldsymbol{x}_e$*. If there exists a local controller* $\boldsymbol{u}_{loc}(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}^m$ *such that the four following conditions hold:*

C1*:* [State constraints satisfied in $\mathcal{X}_f$]*. The terminal region* $\mathcal{X}_f$ *is required to satisfy:*

$$\mathcal{X}_f \subset \mathcal{X}, \ \mathcal{X}_f \text{ closed}, \ \boldsymbol{x}_e \in \mathcal{X}_f. \tag{3.1.8}$$

C2*:* [Input constraints satisfied in $\mathcal{X}_f$]*. The local controller* $\boldsymbol{u}_{loc}(\boldsymbol{x})$ *respects the input constraints within* $\mathcal{X}_f$*:*

$$\boldsymbol{u}_{loc}(\boldsymbol{x}) \in \mathcal{U}, \ \forall \boldsymbol{x} \in \mathcal{X}_f. \tag{3.1.9}$$

C3*:* [Invariant terminal constraint set]*. The terminal region* $\mathcal{X}_f$ *from* (3.1.6d) *is positive invariant under* $\boldsymbol{u}_{loc}(\boldsymbol{x})$*.*

C4*:* [Local Lyapunov function existence]*. For all* $\boldsymbol{x} \in \mathcal{X}_f$*, the following condition holds:*

$$\frac{\partial F(\boldsymbol{x})}{\partial \boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{u}_{loc}(\boldsymbol{x})) + \ell(\boldsymbol{x}, \boldsymbol{u}_{loc}(\boldsymbol{x})) \leq 0, \tag{3.1.10}$$

where $F(\cdot)$ *and* $\ell(\cdot)$ *are the terminal and stage costs from* (3.1.5)*, respectively.*

Then, the closed-loop control system $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}_{MPC})$ *with* $f(\cdot)$ *from* (3.1.1) *and* $\boldsymbol{u}_{MPC}$ *from* (3.1.7) *achieves the recursive feasibility and (nominal) asymptotic stability.* □

Proof. The proof is given in [Mayne et al., 2000], pages 797-799. Furthermore, [Chen and Allgöwer, 1998] shows how to apply a linear local controller $\boldsymbol{u}_{\text{loc}}(\boldsymbol{x})$ to design a *quasi-infinite horizon* NMPC scheme satisfying Lemma 3.1.1. □

Example 3.1.2. *Figure 3.1.1 illustrates the proof detailed in [Chen and Allgöwer, 1998, Mayne et al., 2000] for the recursive feasibility property of an NMPC scheme satisfying Lemma 3.1.1 through an application on a simple dynamics* $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$ *as in* (3.1.1) *with its scalar state and input* $(\boldsymbol{x}, \boldsymbol{u}) \in \mathbb{R}$*. The terminal constraint set* $\mathcal{X}_f$ *of the NMPC controller as in* (3.1.6d) *and*

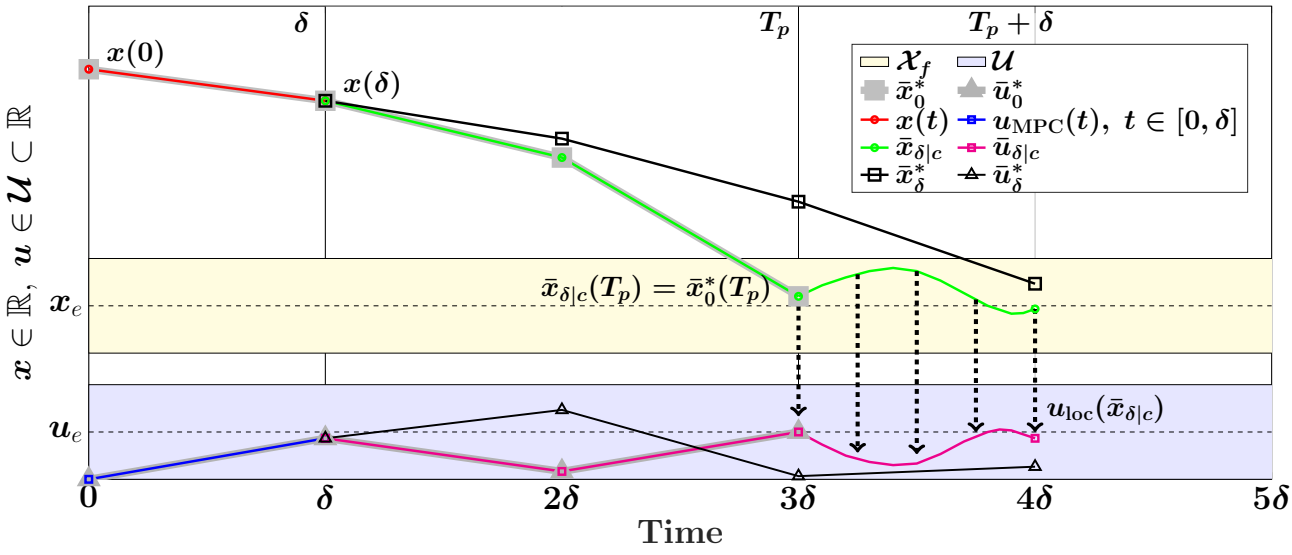


Figure 3.1.1: Illustration of the recursive feasibility property of a scalar system.

the input constraint set $\mathcal{U}$ are given by the blue and yellow polytopic regions with the equilibrium values $(\boldsymbol{x}_e, \boldsymbol{u}_e)$ in their middles, respectively. The simulation scenario is to stabilize the dynamics around $(\boldsymbol{x}_e, \boldsymbol{u}_e)$ from its initial state $\boldsymbol{x}(0)$.
Recursive feasibility implies that the first NMPC iteration successfully executed guarantees the feasibility of the solution for all future steps. Therefore, let us assume that the first NMPC iteration at time $t = 0$ provides the optimal state $\bar{\boldsymbol{x}}_0^*$ and the optimal input $\bar{\boldsymbol{u}}_0^*$ (plotted in thick gray lines with square and triangle marks, respectively). Then, by applying the NMPC input $\boldsymbol{u}_{MPC}$ taken from the optimal input profile $\bar{\boldsymbol{u}}_0^*$ within the first sampling time δ as in (3.1.7) (plotted in blue line with square marks), the state arrives to $\boldsymbol{x}(\delta) = \bar{\boldsymbol{x}}_0^*(\delta)$ as shown by the overlap between the ending point of $\boldsymbol{x}(t)$ (plotted in red line with circle marks) and the second square gray point of the line $\bar{\boldsymbol{x}}_0^*$.
Next, we will show that there exists a feasible solution for the next NMPC iteration starting from $\boldsymbol{x}(\delta)$ at time $t = \delta$. Let us consider the candidate input profile $\bar{\boldsymbol{u}}_{\delta|c}(s)$ (plotted in magenta line with square marks) defined as follows:

$$\bar{\boldsymbol{u}}_{\delta|c}(s) = \begin{cases} \bar{\boldsymbol{u}}_0^*(s), \ \ s \in [\delta, T_p], \\ \boldsymbol{u}_{loc}(\bar{\boldsymbol{x}}_{\delta|c}), \ \ s \in (T_p, T_p + \delta]. \end{cases} \tag{3.1.11}$$

*Since the first part of $\bar{\boldsymbol{u}}_{\delta|c}(s)$ is taken from the previous optimal input profile $\bar{\boldsymbol{u}}_0^*(s)$, it results in the overlapping between the candidate state profile denoted by $\bar{\boldsymbol{x}}_{\delta|c}(s)$ (plotted in green line with circle marks) and the previous optimal state profile $\bar{\boldsymbol{x}}_0^*(s)$ within the interval of $s \in [\delta, T_p]$. Then, $\bar{\boldsymbol{x}}_{\delta|c}(T_p) = \bar{\boldsymbol{x}}_0^*(T_p) \in \mathcal{X}_f$ as noted within the yellow region representing $\mathcal{X}_f$. Next, due to the third condition **C3** of Lemma 3.1.1, $\mathcal{X}_f$ is positive invariant under the local controller $\boldsymbol{u}_{loc}$, hence, the rest of the candidate state profile stays within $\mathcal{X}_f$ (green profile inside the yellow set) so it can vary inside the sampling time because we see the control and state which would result from the continuous local controller. Furthermore, since $\mathcal{X}_f$ is input constraint admissible from the second condition **C2** as in (3.1.9), the candidate input profile $\bar{\boldsymbol{u}}_{\delta|c}(s)$ within the interval $s = [T_p, T_p + \delta]$ always respects the input constraint (as shown by the smoothly varying part of the magenta line within the blue region). Thus, these candidate profiles of state and input are acceptable, hence, guaranteeing the feasibility of the second NMPC iteration.*
Starting from the candidate profiles $\bar{\boldsymbol{x}}_{\delta|c}(s)$ and $\bar{\boldsymbol{u}}_{\delta|c}(s)$, the optimization solver provides the optimal solutions of state, $\bar{\boldsymbol{x}}_\delta^*(s)$ and input $\bar{\boldsymbol{u}}_\delta^*(s)$ for the interval $s = [\delta, T_p + \delta]$ (as plotted by black lines with square and triangle marks, respectively). This also completes the illustrative proof.

The continuous-time NMPC setup given as in (3.1.4)-(3.1.7) allows the discretization step of the prediction model (3.1.6a) (denoted by Δ) to be smaller than the NMPC sampling time δ in (3.1.7) without requiring the explicit formulation as in discrete domain, hence, taking into account inter-sample behavior [Magni and Scattolini, 2004]. More precisely, if the prediction model as in (3.1.6a) is discretized with the step Δ, the discrete implementation of the proposed NMPC scheme in (3.1.4)-(3.1.6) will have N_p prediction steps with

$$N_p = \frac{T_p}{\Delta}, \tag{3.1.12}$$

with T_p the prediction horizon interval. Next, the resulted NMPC input can be taken as N_u first steps (compared to the usage of the only one first step in the standard discrete NMPC design) with:

$$N_u = \frac{\delta}{\Delta}, \tag{3.1.13}$$

with $N_u \in \mathbb{N}^*$ and δ the NMPC sampling time (between two consecutive moments when applying the NMPC input $\boldsymbol{u}_{\text{MPC}}$ (3.1.7)). We emphasize that various works in the literature also consider this NMPC setup in continuous-time [Mayne et al., 2000, Magni and Scattolini, 2004, Chen and Allgöwer, 1998, Reble and Allgöwer, 2012]. In these works, the real implementation of the controller employs a piece-wise constant control input within the sampling time interval (i.e., the input is constrained to keep the same value within every N_u steps) which results in having no differences with the standard discrete NMPC design but at the price of enhancing accuracy of the prediction model.
We decide to employ the continuous-time NMPC formulation not only due to the aforementioned reasons but also due to the ease of integrating the multicopter dynamics (2.1.10) within a continuous NMPC framework. In the next sections, we will particularize the general NMPC design (3.1.4) for a class of "computed-torque like" systems which stand for a broad range of mechanical and robotics systems.

3.2 Use of computed-torque control in NMPC design

In this section, we first present the CTC (Computed-Torque Control) law and its associated dynamical system. Next, we show the construction of an input constraint admissible set under the CTC controller which is obtained by applying Taylor's approximation to the controller. The set allows us to obtain an admissible positive invariant set as required by Lemma 3.1.1 for designing an NMPC scheme with guaranteed stability (i.e., being employed as the terminal constraint set within the NMPC design). Finally, an upper bound is provided for the weighted norm of the CTC controller in terms of the corresponding state within the invariant set which will be used to satisfy the fourth condition (3.1.10) of Lemma 3.1.1.
Let us start by giving the dynamical model of a system admitting a CTC law in its original form and then, adapt it to the general state-space formulation $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$ as in (3.1.1). A "computed-torque like" system is expressed in the second-order dynamics as follows [Craig, 2018, Lewis et al., 2003, Uebel et al., 1992]:

$$\boldsymbol{M}(q)\ddot{q} + \boldsymbol{N}(\dot{q}, q) = \boldsymbol{u}, \tag{3.2.1}$$

with the state $q \in \mathbb{R}^m$ and the actuator input $\boldsymbol{u} \in \mathbb{R}^m$. $\boldsymbol{M}(q) \in \mathbb{R}^{m\times m}$ is a symmetric and positive definite inertia matrix, $\boldsymbol{N}(\dot{q}, q) \in \mathbb{R}^m$ is the vector gathering the nonlinear terms (e.g., Coriolis forces, centrifugal forces [Craig, 2018, Lewis et al., 2003]). By defining the new state vector $\boldsymbol{x} = [q\ \dot{q}]^\top \in \mathbb{R}^{2m}$, the system (3.2.1) is transformed into the state-space formulation given by:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}), \tag{3.2.2}$$

as similar to the general system considered in (3.1.1), Chapter 2, with

$$f(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \boldsymbol{I}_m & \boldsymbol{0}_m \\ \boldsymbol{0}_m & \boldsymbol{M}(q)^{-1} \end{bmatrix} \begin{bmatrix} \dot{q} \\ -\boldsymbol{N}(\dot{q}, q) + \boldsymbol{u} \end{bmatrix}. \tag{3.2.3}$$

The equilibrium point of the system (3.2.1)-(3.2.2) satisfying $f(\boldsymbol{x}_e, \boldsymbol{u}_e) = \boldsymbol{0}$ is defined as:

$$\boldsymbol{x}_e = [q_e\ \boldsymbol{0}]^\top \text{ and } \boldsymbol{u}_e = \boldsymbol{N}(\boldsymbol{0}, q_e), \tag{3.2.4}$$

with $q_e \in \mathbb{R}^m$ the desired output. The state $\boldsymbol{x}$ and the input $\boldsymbol{u}$ are constrained as follows:

$$\boldsymbol{x} \in \mathcal{X},\ \boldsymbol{u} \in \mathcal{U}, \tag{3.2.5}$$

where the state and input constraint sets are denoted by $\mathcal{X}$ and $\mathcal{U}$ and are convex in $\mathbb{R}^{2m}$ and $\mathbb{R}^m$, respectively. $\mathcal{X} \times \mathcal{U}$ contains $(\boldsymbol{x}_e, \boldsymbol{u}_e)$ in its relative interior.

3.2.1 Linearization effect of computed-torque control

The system (3.2.1)-(3.2.2) admits a special feedback linearization law called CTC law [Craig, 2018, Lewis et al., 2003] given as:

$$\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x}, \mu) = \boldsymbol{M}(q)\mu + \boldsymbol{N}(\dot{q}, q), \tag{3.2.6}$$

where $\mu \in \mathbb{R}^m$ gathers the so-called virtual control inputs. The equilibrium value of the virtual input vector μ_e is given by:

$$\mu_e = \mathbf{0}, \tag{3.2.7}$$

which further provides $\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x}_e, \mu_e) = \boldsymbol{u}_e = \boldsymbol{N}(\mathbf{0}, q_e)$ as in (3.2.4). If $\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x}, \mu)$ satisfies the input constraint (3.2.5), it transforms the system (3.2.1)-(3.2.2) into the controllable linear system:

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\mu, \tag{3.2.8}$$

with $A \in \mathbb{R}^{2m\times 2m}$ and $B \in \mathbb{R}^{2m\times m}$ given by:

$$A = \begin{bmatrix} \mathbf{0}_m & \boldsymbol{I}_m \\ \mathbf{0}_m & \mathbf{0}_m \end{bmatrix}, \; B = \begin{bmatrix} \mathbf{0}_m \\ \boldsymbol{I}_m \end{bmatrix}. \tag{3.2.9}$$

Furthermore, we can stabilize the linear system (3.2.8) around its equilibrium point $(\boldsymbol{x}_e, \mu_e)$ from (3.2.4), (3.2.7) by applying a simple linear control design for the virtual input μ as follows:

$$\mu := \mu_K(\boldsymbol{x}) = K(\boldsymbol{x} - \boldsymbol{x}_e) + \mu_e, \tag{3.2.10}$$

in which, $\boldsymbol{x}_e$ is given in (3.2.4), μ_e is from (3.2.7) and the notation ":=" means "to be defined as". Since linear dynamics (3.2.8) are equivalent to n double integrator sub-systems, an appropriate design for the control gain matrix $K \in \mathbb{R}^{n\times 2n}$ is given by:

$$K = [\text{diag}\{K_{p_1}, \ldots, K_{p_n}\} \;\; \text{diag}\{K_{d_1}, \ldots, K_{d_n}\}], \tag{3.2.11}$$

with all the control gains required to be strictly negative, i.e., $\langle K_{p_i}, K_{d_i}\rangle < 0, \forall i \in \{1, \ldots, n\}$. Note that, introducing the virtual input $\mu_K(\boldsymbol{x})$ from (3.2.10) to the system (3.2.8) results in the following linear system:

$$\dot{\boldsymbol{e}} = A_K \boldsymbol{e}, \tag{3.2.12}$$

with $\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}_e$ the tracking error with $\boldsymbol{x}_e$ as in (3.2.4) and $A_K = A + BK$ with A, B, K as in (3.2.8)–(3.2.10). The system (3.2.12) is stable due to the use of all negative control gains $\langle K_{p_i}, K_{d_i}\rangle < 0, \forall i \in \{1, \ldots, n\}$ as in (3.2.11).
Hereinafter, we illustrate the aforementioned content through a typical example of a "computed-torque like" system - an inverted pendulum.

Example 3.2.1. *The dynamical system of the inverted pendulum illustrated in Figure 3.2.1 is given by:*

$$ml^2\ddot{q} - mgl\sin q = \boldsymbol{u}, \tag{3.2.13}$$

with $q \in \mathbb{R}$ the angle between the vertical line and the pendulum, $\boldsymbol{u} \in \mathbb{R}$ the input torque acting on the pendulum, m the mass at the end of the pendulum, g the gravity and l the length of the assumed-massless link. For simplicity, no constraints on state and input are considered. The dynamics (3.2.14) *admits the CTC law given as follows:*

$$\boldsymbol{u}_{CTC}(q, \mu) = ml^2\mu - mgl\sin q, \tag{3.2.14}$$

which linearizes the system (3.2.13) *into a double integrator system $\ddot{q} = \mu$. Furthermore, in case of stabilizing around $q_e = 0$, by applying $\mu = K_p q + K_d\dot{q}$ with any pair of $\langle K_p, K_d\rangle < 0$, the closed-loop controlled system is asymptotically stable.*

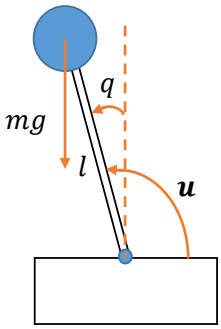


Figure 3.2.1: Schematic of an inverted pendulum.

3.2.2 Satisfaction of input constraints under computed-torque control

In this section, we present the construction of an input constraint admissible set of the CTC law $\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x}, \mu)$ defined in (3.2.6). The first step is to define the shape of the set which is usually chosen either as an ellipsoid [Chen and Allgöwer, 1998, Kohler et al., 2018, Nguyen et al., 2019a], or as a polytope [Cannon et al., 2003, Nguyen et al., 2019b]. The decision is usually made based on the complexity resulted from considering the shape in conjunction with the control law, i.e., the CTC law $\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x}, \mu)$ as in (3.2.6). Since we will employ the input constraint admissible set to construct a positive invariant set later, an ellipsoid set appears as a more promising candidate. Therefore, let us consider a ball $\mathcal{B}$ centered in $(\boldsymbol{x}_e, \mu_e)$ and parameterized by a radius $\varepsilon \in \mathbb{R}^+$ as follows:

$$\mathcal{B}(q_e, \varepsilon) = \Big\{(\boldsymbol{x}, \mu) \in \mathbb{R}^{2m} \times \mathbb{R}^m \Big| \ \|\boldsymbol{x} - \boldsymbol{x}_e\|^2 + \|\mu - \mu_e\|^2 \leq \varepsilon^2\Big\}, \tag{3.2.15}$$

in which, the notation $\mathcal{B}(q_e, \varepsilon)$ is due to the equilibrium value defined as $\boldsymbol{e} = [q_e \ \mathbf{0}]^\top$ and $\mu_e = \mathbf{0}$ from (3.2.4) and (3.2.7). Assuming that the CTC law $\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x}, \mu)$ (3.2.6) is continuous and twice differentiable around the point $(\boldsymbol{x}_e, \mu_e)$ with $\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x}_e, \mu_e) = \boldsymbol{u}_e \in \mathcal{U}$, it is trivial that we can find a "small enough" value of the radius ε such that, for all $(\boldsymbol{x}, \mu)$ within the ball $\mathcal{B}(\cdot)$, the CTC law $\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x}, \mu)$ satisfies the input constraint (3.2.5), i.e., $\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x}, \mu) \in \mathcal{U}$, $\forall (\boldsymbol{x}, \mu) \in \mathcal{B}(q_e, \varepsilon)$ due to the continuity of $\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x}, \mu)$ around $(\boldsymbol{x}_e, \mu_e)$. Then, the problem turns out to estimate how small the radius ε should be such that the set $\mathcal{B}(q_e, \varepsilon)$ as in (3.2.15) is input constraint admissible. The problem is illustrated by the following example.

Example 3.2.2. *Let us consider again Example 3.2.1 on stabilizing the inverted pendulum system* (3.2.13) *around* $q_e = 0$. *The corresponding CTC law* $\boldsymbol{u}_{CTC}(q, \mu)$ *as in* (3.2.14) *depends only on the controlled angle* q *and the virtual input* μ. *Therefore, the ball* $\mathcal{B}$ *from* (3.2.15) *is rewritten as follows:*

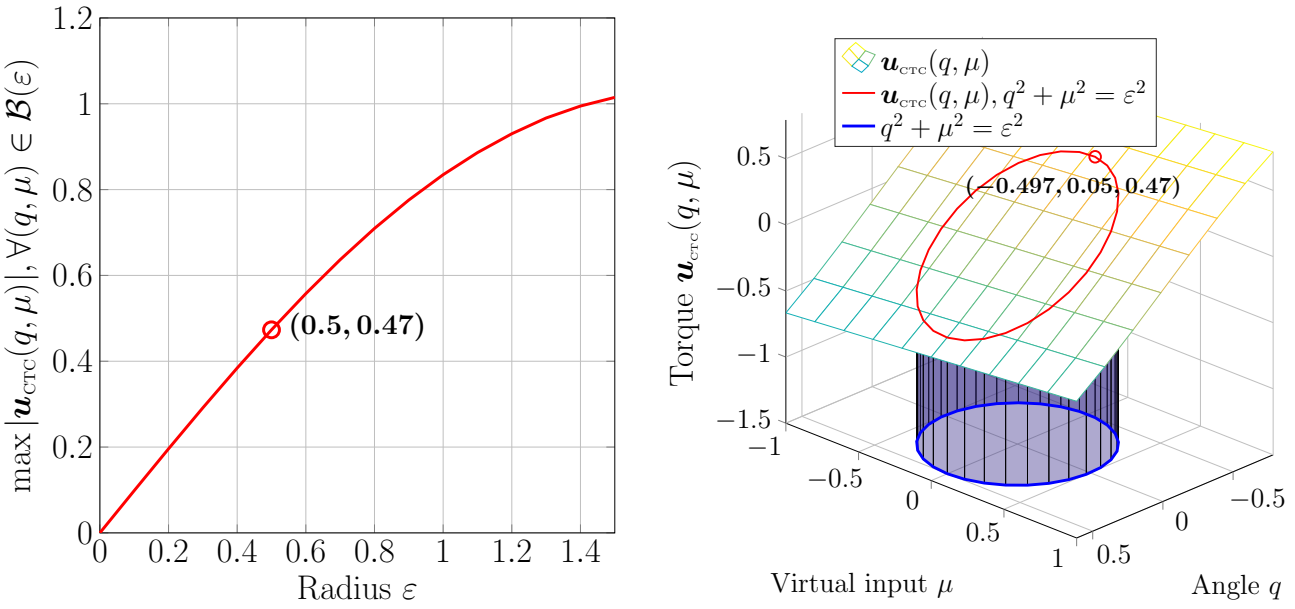
$$\mathcal{B}(\varepsilon) = \Big\{(q, \mu) \in \mathbb{R} \times \mathbb{R} \Big| \ q^2 + \mu^2 \leq \varepsilon^2\Big\}. \tag{3.2.16}$$

We will then illustrate in Figure 3.2.2a the results of the following optimization problem:

$$\begin{aligned} &\max_{q,\mu} |\boldsymbol{u}_{CTC}(q, \mu)| \\ &\text{subject to } q^2 + \mu^2 \leq \varepsilon^2, \end{aligned} \tag{3.2.17}$$

with $\boldsymbol{u}_{CTC}(q, \mu)$ *from* (3.2.14) *and the radius* ε *used as the main variable of the problem. It shows that the maximum value of the torque magnitude* $|\boldsymbol{u}_{CTC}(q, \mu)|$ *for all* $(q, \mu) \in \mathcal{B}(\varepsilon)$ *is increasing when enlarging the set* $\mathcal{B}(\varepsilon)$. *The case of* $\varepsilon = 0.5$ *(as marked by red circles) is detailed in Figure*

3.2.2b which shows the intersection between the mapping $\boldsymbol{u}_{CTC}(q,\mu)$ *and the vertical cylinder with its radius* $\varepsilon = 0.5$.



(a) The maximum values of the torque magnitude obtained inside the ball $\mathcal{B}(\varepsilon)$ with respect to different values of ε.

(b) The mapping $\boldsymbol{u}_{\text{CTC}}(q,\mu)$ and the region limited by $(q,\mu) \in \mathcal{B}(\varepsilon)$ with $\varepsilon = 0.5$.

Figure 3.2.2: Analysis of the relation between $\max|\boldsymbol{u}_{\text{CTC}}(q,\mu)|, \forall(q,\mu) \in \mathcal{B}(\varepsilon)$ and ε.

Through Example 3.2.2, we notice that the "small enough" value of ε such that the CTC law $\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x},\mu)$ from (3.2.6) satisfies the input constraint (3.2.5) can be obtained by solving the optimization problem (3.2.17). However, it does require the global optimal solution of (3.2.17) which even contains the generally nonlinear and multi-variable function $\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x},\mu)$. These obstacles usually prevents us from obtaining the required value of ε. Therefore, in the following, we provide another method for estimating the radius ε based on the Taylor's approximation of $\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x},\mu)$ [Nguyen et al., 2019a]. We first derive an upper bound for $\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x},\mu)$, then, we introduce a condition on ε in order to validate the input constraint (3.2.5) limited by the bound.

We start by addressing the Taylor's approximation [Folland, 1990] of $\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x},\mu)$ from (3.2.6) for all $(\boldsymbol{x},\mu) \in \mathcal{B}(q_e,\varepsilon)$ as in (3.2.15):

$$\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x},\mu) = \boldsymbol{u}_{\text{CTC}}(\boldsymbol{x}_e,\mu_e) + \underbrace{\left.\frac{\partial \boldsymbol{u}_{\text{CTC}}}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_e \\ \mu=\mu_e}}}_{{}^{\boldsymbol{x}}J(q_e)}(\boldsymbol{x}-\boldsymbol{x}_e) + \underbrace{\left.\frac{\partial \boldsymbol{u}_{\text{CTC}}}{\partial \mu}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_e \\ \mu=\mu_e}}}_{{}^{\mu}J(q_e)}(\mu-\mu_e) + \mathcal{R}(\boldsymbol{x},\mu,q_e,\varepsilon), \quad (3.2.18)$$

in which, the Jacobian matrix ${}^{\mu}J(q_e)$ is explicitly given by ${}^{\mu}J(q_e) = \boldsymbol{M}(q_e)$ with $\boldsymbol{M}(\cdot)$ the inertia matrix from (3.2.1) and q_e the desired output from (3.2.4). The remainder vector $\mathcal{R} \in \mathbb{R}^n$ is bounded from Taylor's inequality [Folland, 1990] as follows[2]:

$$|\mathcal{R}(\boldsymbol{x},\mu,q_e,\varepsilon)| \leq \frac{\mathcal{M}(q_e,\varepsilon)}{2!}\left(\|\boldsymbol{x}-\boldsymbol{x}_e\|^2 + \|\mu-\mu_e\|^2\right), \quad (3.2.19)$$

[2]In (3.2.19), $|\mathcal{R}(\boldsymbol{x},\mu,q_e,\varepsilon)|$ is a vector in $\mathbb{R}^m$ containing the norms of all the elements of $\mathcal{R}(\boldsymbol{x},\mu,q_e,\varepsilon)$.

with $\mathcal{M}(q_e, \varepsilon) \in \mathbb{R}^m$ gathering m elements denoted by $\mathcal{M}_i(\cdot) \in \mathbb{R}^+$ with $i \in \{1, \ldots, m\}$. Each element $\mathcal{M}_i(\cdot)$ is defined as:

$$\mathcal{M}_i(q_e, \varepsilon) = \max_{(\boldsymbol{x},\mu)\in\mathcal{B}(q_e,\varepsilon)} |\boldsymbol{H}(\boldsymbol{u}_{\text{CTC},i}(\boldsymbol{x}, \mu))|, \tag{3.2.20}$$

where $\boldsymbol{u}_{\text{CTC},i}(\cdot)$ is the i^{th} element of the vector function $\boldsymbol{u}_{\text{CTC}}(\cdot)$ from (3.2.6). $\boldsymbol{H}(\cdot)$ is the Hessian matrix of a scalar-valued function containing all of its second-order partial derivatives [Meyer, 2000]. Furthermore, by introducing (3.2.19) and the Cauchy-Schwarz inequality [Meyer, 2000] to each element $\boldsymbol{u}_{\text{CTC},i}(\cdot)$ from (3.2.18) ($i \in \{1, \ldots, m\}$), it is straightforward to obtain:

$$|\boldsymbol{u}_{\text{CTC},i}(\boldsymbol{x}, \mu) - \boldsymbol{u}_{e,i}| \leq C_i(q_e)\sqrt{\|\boldsymbol{x} - \boldsymbol{x}_e\|^2 + \|\mu - \mu_e\|^2} + \frac{\mathcal{M}_i(q_e, \varepsilon)}{2}\left(\|\boldsymbol{x} - \boldsymbol{x}_e\|^2 + \|\mu - \mu_e\|^2\right), \tag{3.2.21}$$

in which $C_i(q_e) \in \mathbb{R}^+$ ($i \in \{1, \ldots, m\}$) is defined as:

$$C_i(q_e) = \sqrt{\|{}^{\boldsymbol{x}}J_i(q_e)\|^2 + \|{}^{\mu}J_i(q_e)\|^2}, \tag{3.2.22}$$

with ${}^{\boldsymbol{x}}J_i(q_e)$ and ${}^{\mu}J_i(q_e)$ the i^{th} rows of the Jacobian matrices ${}^{\boldsymbol{x}}J$ and ${}^{\mu}J$ in (3.2.18), respectively. Then, since we are considering the region defined by $\|\boldsymbol{x} - \boldsymbol{x}_e\|^2 + \|\mu - \mu_e\|^2 \leq \varepsilon^2$, (3.2.21) leads to:

$$|\boldsymbol{u}_{\text{CTC},i} - \boldsymbol{u}_{e,i}| \leq \left(C_i(q_e) + \frac{\mathcal{M}_i(q_e, \varepsilon)}{2}\varepsilon\right)\sqrt{\|\boldsymbol{x} - \boldsymbol{x}_e\|^2 + \|\mu - \mu_e\|^2}. \tag{3.2.23}$$

Then, let define $\varepsilon_{\max} \in \mathbb{R}^+$ as the largest possible value of ε satisfying:

$$\begin{aligned} \varepsilon_{\max} &= \max \varepsilon, \\ \text{subject to } C(q_e)\varepsilon + \frac{\mathcal{M}(q_e, \varepsilon)}{2}\varepsilon^2 &\leq \boldsymbol{u}_{\max}, \end{aligned} \tag{3.2.24}$$

in which $C(\cdot) \triangleq [C_1(\cdot), \ \ldots, C_m(\cdot)]^\top$ with $C_i(\cdot)$ from (3.2.22) and $\boldsymbol{u}_{\max} \in \mathbb{R}^m_+$ gathers all the user-defined maximum input values along each input axis. More precisely, $\boldsymbol{u}_{\max}$ is chosen such that:

$$\mathcal{U}^* = \{\boldsymbol{u} \in \mathbb{R}^m | -\boldsymbol{u}_{\max} + \boldsymbol{u}_e \leq \boldsymbol{u} \leq \boldsymbol{u}_{\max} + \boldsymbol{u}_e\} \subseteq \mathcal{U}, \tag{3.2.25}$$

with $\mathcal{U}$ the input constraint set from (3.2.5) and $\mathcal{U}^*$ the alternative input constraint set composed of all saturation constraints. Then, by introducing (3.2.24) to (3.2.23), we have that:

$$\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x}, \mu) \in \mathcal{U}^*, \ \forall(\boldsymbol{x}, \mu) \in \mathcal{B}(q_e, \varepsilon), \ \forall\varepsilon \leq \varepsilon_{\max}, \tag{3.2.26}$$

with $\mathcal{U}^* \subseteq \mathcal{U}$ from (3.2.25), $\mathcal{B}(q_e, \varepsilon)$ from (3.2.15) and $\varepsilon_{\max}$ the largest possible value of ε as in (3.2.24).
Furthermore, by summing the inequalities (3.2.23), we obtain the Lipschitz bound of the CTC law $\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x}, \mu)$ defined (3.2.6) within the set $\mathcal{B}(q_e, \varepsilon)$:

$$\|\boldsymbol{u}_{\text{CTC}} - \boldsymbol{u}_e\|^2 \leq \mathsf{L}_{\text{CTC}}\left(\|\boldsymbol{x} - \boldsymbol{x}_e\|^2 + \|\mu - \mu_e\|^2\right), \tag{3.2.27}$$

with the Lipschitz constant L_{CTC} taken as follows:

$$\mathsf{L}_{\text{CTC}}(q_e, \varepsilon) = \left\|C(q_e) + \frac{\mathcal{M}(q_e, \varepsilon)}{2}\varepsilon\right\|^2, \tag{3.2.28}$$

with $C(q_e)$ as in (3.2.22) and $\mathcal{M}(q_e, \varepsilon)$ as in (3.2.20).

Remark 3.2.3. In the literature, $\varepsilon_{\max}$ is usually assumed to be known [Kohler et al., 2018] since the numerical result of $\varepsilon_{\max}$ can be obtained by progressively increasing an arbitrary small value $\varepsilon \in \mathcal{R}^+$ while guaranteeing that $\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x}, \mu) \in \mathcal{U}$, for all $(\boldsymbol{x}, \mu) \in \mathcal{B}(q_e, \varepsilon)$ as in (3.2.26). However, the method requires heavier computation effort than validating condition (3.2.24) and it does not provide the useful quadratic bound of the controller $\boldsymbol{u}_{\text{CTC}}$ as in (3.2.27).
Futhermore, the Lipschitz constant $\mathsf{L}_{\text{CTC}}(q_e, \varepsilon)$ as in (3.2.28) can be numerically obtained by solving:

$$\begin{aligned} \log \mathsf{L}_{\text{CTC}} = &\max_{\boldsymbol{x},\mu} \left\{ 2 \log \|\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x}, \mu) - \boldsymbol{u}_e\| - \log \left(\|\boldsymbol{x} - \boldsymbol{x}_e\|^2 + \|\mu - \mu_e\|^2 \right) \right\}, \\ &\text{subject to } (\boldsymbol{x}, \mu) \in \mathcal{B}(q_e, \varepsilon), \end{aligned} \tag{3.2.29}$$

with $\boldsymbol{u}_{\text{CTC}}$ as in (3.2.27) and $\mathcal{B}(q_e, \varepsilon)$ as in (3.2.30). However, this numerical approach requires much computation time for solving the nonlinear optimization problem (3.2.29) while it is not the case for the analytical formulation of $\mathsf{L}_{\text{CTC}}(q_e, \varepsilon)$ as given in (3.2.28). □

In this section, we have shown the method to obtain the input constraint admissible set and the Lipstchiz bound of the CTC law $\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x}, \mu)$ defined in (3.2.6). The method is constructed based on the Taylor's approximation of the CTC law and the bound on the remainder term which can be generalized for various nonlinear controllers. The next section will employ the presented results in order to construct a constraint admissible and positive invarian set under the CTC law.

3.2.3 Construction of invariant set associated to the computed-torque control

In this section, we enforce the a priori obtained input constraint admissible set $\mathcal{B}(q_e, \varepsilon)$ from (3.2.15) to become positive invariant. To do so, the control gain matrix K from (3.2.10) needs to be chosen appropriately. Let us start by introducing $\mu = \mu_K(\boldsymbol{x})$ as in (3.2.10) to the set $\mathcal{B}(q_e, \varepsilon)$ as in (3.2.15) (originally described in terms of $\boldsymbol{x}$ and μ). The resulted input constraint admissible set, denoted by $\mathcal{B}_K(q_e, \varepsilon)$, is given as follows:

$$\mathcal{B}_K(q_e, \varepsilon) = \left\{ \boldsymbol{x} \in \mathbb{R}^{2m} \,\middle|\, (\boldsymbol{x} - \boldsymbol{x}_e)^\top (\boldsymbol{I}_{2m} + K^\top K)(\boldsymbol{x} - \boldsymbol{x}_e) \leq \varepsilon^2 \right\}, \tag{3.2.30}$$

with K the control gain matrix as in (3.2.11), $\boldsymbol{I}_{2m}$ the identity matrix of size $2m \times 2m$ and $\varepsilon \in \mathbb{R}_+$ the radius satisfying $\varepsilon \leq \varepsilon_{\max}$ as in (3.2.26).
Next, we introduce a condition on choosing K as in (3.2.11) such that the input constraint admissible set $\mathcal{B}_K(q_e, \varepsilon)$ from (3.2.30) becomes positive invariant.

Lemma 3.2.4. *Let us consider the matrix K defined in* (3.2.11) *where the $2m$ control gains $K_{p_1}, \ldots, K_{p_m}$ and $K_{d_1}, \ldots, K_{d_m}$ satisfy the following conditions:*

$$\begin{cases} K_{p_i} < 0, \; K_{d_i} < 0, \\ 4K_{d_i}^2 > -K_{p_i}(K_{p_i} + 1)^2 - K_{p_i} - \frac{(K_{p_i}+1)^2}{K_{p_i}}, \end{cases} \quad i \in \{1, \ldots, m\}, \tag{3.2.31}$$

and the set $\mathcal{B}_K(q_e, \varepsilon)$ from (3.2.30) *where the radius ε is chosen such that:*

$$\begin{cases} \varepsilon \leq \varepsilon_{max}, \\ \mathcal{B}_K(q_e, \varepsilon) \subseteq \mathcal{X}, \end{cases} \tag{3.2.32}$$

with ε_{max} from (3.2.24). *Then, the set $\mathcal{B}_K(q_e, \varepsilon)$ is constraint admissible and positive invariant under the CTC controller $\boldsymbol{u}_{CTC}(\boldsymbol{x}, \mu_K(\boldsymbol{x}))$ with $\boldsymbol{u}_{CTC}(\boldsymbol{x}, \mu)$ from* (3.2.6) *and $\mu_K(\boldsymbol{x})$ from* (3.2.10).
□

Proof. At first, choosing the radius ε as in (3.2.32) makes $\mathcal{B}_K(q_e, \varepsilon)$ admissible for both input and state constraints. Then, the invariant property is proved by the Lyapunov function $V = (\boldsymbol{x} - \boldsymbol{x}_e)^\top (\boldsymbol{I}_{2m} + K^\top K)(\boldsymbol{x} - \boldsymbol{x}_e)$ as employed for constructing $\mathcal{B}_K(q_e, \varepsilon)$ in (3.2.30). Without loss of generality, let us consider the case of $\boldsymbol{x}_e = \boldsymbol{0}$. Then, by using $\boldsymbol{x} \triangleq [x_1, \ldots, x_{2m}]^\top$ and K as defined in (3.2.11), we have that:

$$V = \sum_{i=1}^{n} \left((x_i^2 + x_{i+m}^2 + (K_{p_i} x_i + K_{d_i} x_{i+m})^2 \right). \tag{3.2.33}$$

Next, from (3.2.9), for $i \in \{1, \ldots, m\}$, we have that $\dot{x}_m = x_{i+m}$ and $\dot{x}_{i+m} = K_{p_i} x_i + K_{d_i} x_{i+m}$ which lead to:

$$\dot{V} = \sum_{i=1}^{m} X_i^\top \Gamma_i X_i, \tag{3.2.34}$$

with $X_i = [x_i \; x_{i+m}]^\top$ and the matrix $\Gamma_i \in \mathbb{R}^{2\times 2}$ given by:

$$\Gamma_i = \begin{bmatrix} 2K_{d_i} K_{p_i}^2 & \Gamma_{i,12} \\ \Gamma_{i,21} & 2K_{d_i}(K_{d_i}^2 + K_{p_i} + 1) \end{bmatrix}, \tag{3.2.35}$$

with $\Gamma_{i,12} = \Gamma_{i,21} = K_{p_i}(2K_{d_i}^2 + K_{p_i} + 1) + 1$. By introducing the conditions (3.2.31) to (3.2.35), the matrix Γ_i is shown to be negative definite, leading to $\dot{V} < 0$. Thus, $\mathcal{B}_K(q_e, \varepsilon)$ from (3.2.30) is positive invariant, completing the proof. □

In this section, we have discussed how to construct a constraint admissible and positive invariant set for a "computed-torque like" system as in (3.2.1). The following section will gather all the presented items into an algorithm which provides an NMPC scheme for the considered system.

3.3 NMPC design with stability induced by a computed-torque controller

Lemma 3.3.1 (NMPC design with stability induced by a local computed-torque control law)**.** *Let us apply the general NMPC setup* (3.1.4)-(3.1.7) *for stabilizing the system* (3.2.2) *admitting the CTC law* (3.2.6)*. Then, the NMPC design will satisfy the four design conditions* ***C1-C4*** *given in Lemma 3.1.1 by using the ingredients defined as follows:*

- *the CTC controller* $\boldsymbol{u}_{CTC}(\boldsymbol{x}, \mu_K(\boldsymbol{x}))$ *from* (3.2.6) *and* (3.2.10) *is used as the local controller* $\boldsymbol{u}_{loc}$ *employed in* (3.1.8)–(3.1.10)*:*

$$\boldsymbol{u}_{loc}(\boldsymbol{x}) := \boldsymbol{u}_{CTC}(\boldsymbol{x}, \mu_K(\boldsymbol{x})), \tag{3.3.1}$$

  in which, := *means "to be defined as".*

- *the terminal constraint set* $\mathcal{X}_f$ *employed in* (3.1.6d) *is taken as the constraint admissible and positive invariant set* $\mathcal{B}_K(q_e, \varepsilon)$ *defined in* (3.2.30)*:*

$$\mathcal{X}_f := \mathcal{B}_K(q_e, \varepsilon). \tag{3.3.2}$$

- *the stage and terminal costs are in their standard quadratic forms:*

$$\ell(\boldsymbol{x},\boldsymbol{u}) := (\boldsymbol{x}-\boldsymbol{x}_e)^\top Q(\boldsymbol{x}-\boldsymbol{x}_e) + (\boldsymbol{u}-\boldsymbol{u}_e)^\top R(\boldsymbol{u}-\boldsymbol{u}_e), \tag{3.3.3}$$

$$F(\boldsymbol{x}) := (\boldsymbol{x}-\boldsymbol{x}_e)^\top P(\boldsymbol{x}-\boldsymbol{x}_e), \tag{3.3.4}$$

in which, $Q \in \mathbb{R}^{n\times n}$ is a positive definite matrix and $R \in \mathbb{R}^{m\times m}$ is a positive semi-definite matrix, all being symmetric. The matrix $P \in \mathbb{R}^{6\times 6}$ is obtained as the unique positive definite matrix solution of the following Lyapunov equation:

$$A_K^\top P + P A_K + Q + R^* = \mathbf{0}, \tag{3.3.5}$$

with A_K the stable matrix from (3.2.12) *and $R^* \in \mathbb{R}^{n\times n}$ the positive definite matrix satisfying:*

$$R^* \succeq \max(\mathrm{eig}(R))\mathsf{L}(\boldsymbol{I}_n + K^\top K), \tag{3.3.6}$$

with the Lipschitz bound L from (3.2.28) *and the control gain matrix K from* (3.2.10).

Using these elements ensures the recursive feasibility and asymptotic stability of the (nominal) closed-loop controlled system. □

Proof. Conditions **C1-C3** given in (3.1.8)-(3.1.9) are validated by employing the constraint admissible and positive invariant set $\mathcal{B}_K(q_e,\varepsilon)$ as in (3.2.30).
Next, for Condition **C4**, by introducing the proposed elements (3.3.1)-(3.3.4) to the requirement (3.1.10), we have that:

$$\frac{\partial F(\boldsymbol{x})}{\partial \boldsymbol{x}} f(\boldsymbol{x},\boldsymbol{u}_{\mathrm{loc}}(\boldsymbol{x})) = (\boldsymbol{x}-\boldsymbol{x}_e)^\top (A_K^\top P + P A_K)(\boldsymbol{x}-\boldsymbol{x}_e), \tag{3.3.7}$$

$$\ell(\boldsymbol{x},\boldsymbol{u}_{\mathrm{loc}}(\boldsymbol{x})) = (\boldsymbol{x}-\boldsymbol{x}_e)^\top Q(\boldsymbol{x}-\boldsymbol{x}_e) + (\boldsymbol{u}_{\mathrm{CTC}}(\cdot)-\boldsymbol{u}_e)^\top R(\boldsymbol{u}_{\mathrm{CTC}}(\cdot)-\boldsymbol{u}_e), \tag{3.3.8}$$

in which, (3.3.7) is due to the feedback linearization effect of the local CTC controller (3.3.1) (c.f. the linear system (3.2.12)). Moreover, the input term in (3.3.8) is bounded as follows:

$$\begin{aligned}(\boldsymbol{u}_{\mathrm{CTC}}(\boldsymbol{x},\mu_K(\boldsymbol{x})) - \boldsymbol{u}_e)^\top R(\boldsymbol{u}_{\mathrm{CTC}}(\boldsymbol{x},\mu_K(\boldsymbol{x})) - \boldsymbol{u}_e) &\le \max(\mathrm{eig}(R))\|\boldsymbol{u}_{\mathrm{CTC}}(\boldsymbol{x},\mu_K(\boldsymbol{x})) - \boldsymbol{u}_e\|^2 \\ &\le \max(\mathrm{eig}(R))\mathsf{L}_{\mathrm{CTC}}(\|\boldsymbol{x}-\boldsymbol{x}_e\|^2 + \|K(\boldsymbol{x}-\boldsymbol{x}_e)\|^2),\end{aligned} \tag{3.3.9}$$

with $\mathsf{L}_{\mathrm{CTC}}$ the Lipschitz constant from (3.2.28), K the control gain matrix from (3.2.11) and R as in (3.3.3) having all non-negative real eigenvalues. Note that, (3.3.9) is due to the quadratic bound of the CTC controller as in (3.2.27). Then, introducing (3.3.7)-(3.3.9) to (3.1.10) ultimately leads to:

$$\begin{aligned}&\frac{\partial F(\boldsymbol{x})}{\partial \boldsymbol{x}} f(\boldsymbol{x},\boldsymbol{u}_{\mathrm{loc}}(\boldsymbol{x})) + \ell(\boldsymbol{x},\boldsymbol{u}_{\mathrm{loc}}(\boldsymbol{x})) \\ &\le (\boldsymbol{x}-\boldsymbol{x}_e)^\top (A_K^\top P + P A_K + Q + \max(\mathrm{eig}(R))\mathsf{L}(\boldsymbol{I}_n + K^\top K))(\boldsymbol{x}-\boldsymbol{x}_e) \\ &\le (\boldsymbol{x}-\boldsymbol{x}_e)^\top (A_K^\top P + P A_K + Q + R^*)(\boldsymbol{x}-\boldsymbol{x}_e) = 0,\end{aligned} \tag{3.3.10}$$

in which, R^* is chosen as in (3.3.6) and P satisfied the Lyapunov equation (3.3.5). This validates Condition **C4** given in (3.1.10) of Lemma 3.1.1, hence, completing the proof. □

Hereinafter, we summarize the design procedure of an NMPC scheme presented above.

Procedure 3.3.2 (NMPC design with stability induced by a CTC law)**.** *The design procedure of the NMPC controller defined in* (3.1.4)–(3.1.7) *employing the CTC controller* (3.2.6) *as the local controller for guaranteeing the closed-loop stability is given as follows:*

1. *Analyze the CTC law* (3.2.6) *of the system then calculate* $C(q_e)$ *as in* (3.2.22) *and formulate* $\mathcal{M}(q_e, \varepsilon)$ *as in* (3.2.20). *Then, find the largest possible radius* ε_{max} *satisfying* (3.2.24).

2. *Define the control gains matrix* K *as in* (3.2.11) *satisfying* (3.2.31).

3. *Choose the symmetric matrices* $Q \in \mathbb{R}^{2m\times 2m}$ *(positive definite) and* $R \in \mathbb{R}^{m\times m}$ *(positive semi-definite) to formulate the stage cost as in* (3.3.3).

4. *Estimate the prediction horizon* T_p *based on the computational limits of the platform.*

5. *Find the radius* ε *satisfying* (3.2.32) *to obtain the constraint admissible and positive invariant set* $\mathcal{B}_K(q_e, \varepsilon)$ *as in* (3.2.30).

6. *Define the matrix* R^* *satisfying* (3.3.6) *with the Lipschitz bound* $\mathsf{L}_{CTC}(q_e, \varepsilon)$ *from* (3.2.28), *then, solve the Lyapunov equation* (3.3.5) *for the terminal weight matrix* P.

For a predefined initial state $\boldsymbol{x}_0$, the procedure requires the users to run the NMPC algorithm once in order to check if the first iteration is feasible. If not, one solution is to progressively increase the prediction horizon until the optimization problem becomes feasible. However, the computation time is greatly affected by any increase of the prediction length. Thus, in order to continue increasing the region of attraction when the prediction horizon T_p is already large (i.e., the computing time reaches the platform limit), one can increase the size of the terminal constraint set $\mathcal{B}_K(q_e, \varepsilon)$ given in (3.2.30) instead. The parameters which affects most the size of the set $\mathcal{B}_K(q_e, \varepsilon)$ is the control gain matrix K defined at Step 2 of the procedure. In general, decreasing the magnitudes of the control gains increase the size of the set $\mathcal{B}_K(q_e, \varepsilon)$ and vice versa. Furthermore, Procedure 3.3.2 can be also applied for tracking control design with only slight modification. If all the information of the reference trajectory is available, i.e., $(q_r, \dot{q}_r, \ddot{q}_r)$ beforehand, then, the radius ε as in (3.2.32) (c.f. Step 5 in Procedure 3.3.2) can be found off-line such that the sets $\mathcal{B}_K(q_r, \varepsilon)$ (i.e., q_r is varying) defined along the trajectory are all constraint admissible and positive invariant.

Remark 3.3.3. At Step 2, if the control gains matrix K as in (3.2.11) does not satisfy the second condition of (3.2.31), the set $\mathcal{B}_K(q_e, \varepsilon_{\max})$ is still input constraint admissible but not positive invariant. Then, another invariant set can be obtained as follows:

$$\mathcal{I}(q_e, r) = \{(\boldsymbol{x} - \boldsymbol{x}_e)^\top P(\boldsymbol{x} - \boldsymbol{x}_e) \leq r^2\}, \tag{3.3.11}$$

with $\boldsymbol{x}_e = [q_e \ 0]^\top$ as in (3.2.4) and P as in (3.3.5). The invariant property of $\mathcal{I}(q_e, r)$ from (3.3.11) is guaranteed by the Lyapunov equation (3.3.5), however, the parameter $r \in \mathbb{R}_+$ needs to be scaled such that:

$$\mathcal{I}(q_e, r) \subseteq \mathcal{B}_K(q_e, \varepsilon_{\max}) \text{ and } \mathcal{I}(q_e, r) \subseteq \mathcal{X}, \tag{3.3.12}$$

with $\mathcal{B}_K(q_e, \varepsilon_{\max})$ as in (3.2.30) being input constraint admissible and $\mathcal{X}$ as in (3.1.6) the state constraint set. Depending on the choice of Q and R^* as in (3.3.5), the resulted shape of $\mathcal{I}(q_e, r)$ from (3.3.11) is probably different from the set $\mathcal{B}_K(q_e, \varepsilon_{\max})$ from (3.2.30), hence, leading to a challenging tuning procedure. We illustrate the idea in Figure 3.3.1 in which we fix the equilibrium at zero and the set $\mathcal{B}_K(q_e, \varepsilon_{\max})$ as in (3.2.30) (yellow ellipsoid) is obtained with $K = [-1 \ -2]$ and $\varepsilon_{\max} = 1$, being both state and input constraint admissible (i.e., $\mathcal{B}_K(q_e, \varepsilon_{\max}) \subseteq \mathcal{X}$ as in (3.1.6)). It is much larger than the invariant set $\mathcal{I}(q_e, r)$ as defined in (3.3.11) (blue ellipsoid) which is obtained by introducing an arbitrarily chosen $Q + R^* = \boldsymbol{I}_2$ to the Lyapunov equation (3.3.5) and then, by scaling the radius r down to 0.29 so that the condition $\mathcal{I}(q_e, r) \subseteq \mathcal{B}_K(q_e, \varepsilon_{\max})$ (3.3.12) is satisfied. Note that, with the optimal choice of $Q + R^* = [4 \ 7; 7 \ 16]$ and $r = 1$, the set $\mathcal{I}(q_e, r)$ as in (3.3.11) will equal the yellow ellipsoid set $\mathcal{B}_K(q_e, \varepsilon_{\max})$ in Figure 3.3.1. □

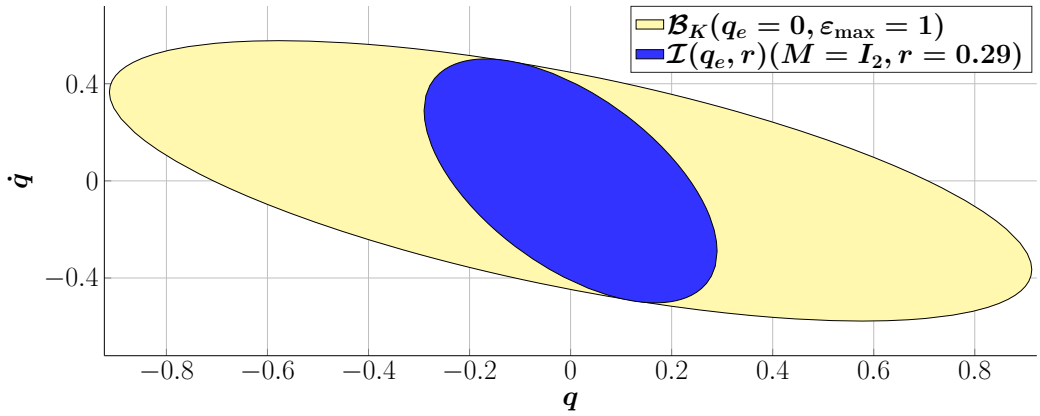


Figure 3.3.1: Illustration of the set $\mathcal{B}_K(q_e, \varepsilon)$ from (3.2.30) and the invariant set $\mathcal{I}(q_e, r)$ from (3.3.11) using $r = 0.29$, $K = [-1 \ -2]$ as in (3.2.11) and $Q + R^* = \boldsymbol{I}_2$ as in (3.3.5).

In the next section, the design procedure is applied on a well-known mechanical system, i.e., to stabilize an inverted pendulum on a cart. The resulted NMPC controller is validated through various simulations and is compared with the *quasi-infinite horizon* NMPC design approach [Chen and Allgöwer, 1998].

3.4 Application on stabilizing a cart-pendulum system

In this section, we consider the problem of stabilizing a well-known mechatronics system, an inverted pendulum robot on a cart [Mazenc and Praly, 1996, Srinivasan et al., 2009]. The angular dynamics of the system are as in (3.2.1) with $q \in \mathbb{R}$, the angle between the vertical line and the pendulum (as defined in Figure 3.2.1). The terms $\boldsymbol{M}(q)$ and $\boldsymbol{N}(q, \dot{q})$ as in (3.2.1) are given by:

$$\boldsymbol{M}(q) = \mu \cos q - \frac{mJ}{\mu \cos q}, \ \boldsymbol{N}(q, \dot{q}) = mg - \mu \dot{q}^2 \sin q, \tag{3.4.1}$$

with $m = 0.3235$, $\mu = 1.3625 \times 10^{-3}$ and $J = 1.5265 \times 10^{-4}$ the physical parameters of the system, $g = 9.81$ the gravity and $\boldsymbol{u} \in \mathbb{R}$ the force applied to the cart (not the torque on the pendulum as employed in Example 3.2.1). The input constraint is $|\boldsymbol{u}| \leq \boldsymbol{u}_{max}$ with $\boldsymbol{u}_{max} = 0.6$ and the state is constrained by $|q| \leq 0.16$, $|\dot{q}| \leq 0.3$. All the parameters are taken from a real experimental setup given in [Srinivasan et al., 2009]. The control problem is to stabilize the pendulum at the upright position, i.e., $q_e = 0$ from its initial state fixed at $q_0 = 0.15$ *rad*. For doing this, we employ two NMPC controllers, one following the proposed Procedure 3.3.2 and the other called *quasi-infinite horizon NMPC* (denoted by qMPC hereinafter) detailed in [Chen and Allgöwer, 1998] in order to analyze and compare the performances. In the following, we briefly introduce the qNMPC design procedure (we employ the second method presented in [Chen and Allgöwer, 1998] which provides less conservative results).

3.4.1 Comparison with quasi-infinite horizon NMPC

Let us consider the general system $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$ and the corresponding state and input constraints given in (3.1.1)–(3.1.3). The qMPC approach stabilizes the system around its equilibrium point $(\boldsymbol{x}_e, \boldsymbol{u}_e)$ from (3.2.4) by using a similar design as in (3.1.5)–(3.1.6) but with the closed-loop stability guaranteed by a local linear controller defined as follows:

$$\boldsymbol{u}_{\text{loc}}(\boldsymbol{x}) \triangleq \boldsymbol{u}_{\text{l,q}}(\boldsymbol{x}) = K_{\text{q}}(\boldsymbol{x} - \boldsymbol{x}_e) + \boldsymbol{u}_e, \tag{3.4.2}$$

with $\boldsymbol{u}_{\text{loc}}(\boldsymbol{x})$ the local controller as employed in (3.1.9)–(3.1.10). The control gain matrix $K_{\text{q}} \in \mathbb{R}^{m\times n}$ is chosen such that the controller $\boldsymbol{u}_{\text{l,q}}(\boldsymbol{x})$ from (3.4.2) stabilizes the linear system:

$$\dot{\boldsymbol{x}} = A_{\text{q}}(\boldsymbol{x} - \boldsymbol{x}_e) + B_{\text{q}}(\boldsymbol{u} - \boldsymbol{u}_e), \tag{3.4.3}$$

which is obtained by linearizing the system (3.1.1) around the equilibrium point with $A_{\text{q}} = (\partial f)/(\partial \boldsymbol{x})(\boldsymbol{x}_e, \boldsymbol{u}_e)$ and $B_{\text{q}} = (\partial f)/(\partial \boldsymbol{u})(\boldsymbol{x}_e, \boldsymbol{u}_e)$. The next step is to define $\kappa \in \mathbb{R}_+$ such that:

$$\kappa < -\max\{\text{Re}\{\text{eig}\,(\underbrace{A_{\text{q}} + B_{\text{q}}K_{\text{q}}}_{\triangleq A_{K_{\text{q}}}})\}\}, \tag{3.4.4}$$

with $\text{Re}\{\text{eig}(A_{K_{\text{q}}})\}$ giving the real parts of all the eigenvalues of $A_{K_{\text{q}}}$. Then, the terminal weighting matrix P_{q} (employed similarly as P in (3.3.4)) is obtained by solving the following Lyapunov equation:

$$\left(A_{K_{\text{q}}} + \kappa \boldsymbol{I}_n\right)^\top P_{\text{q}} + P_{\text{q}}\left(A_{K_{\text{q}}} + \kappa \boldsymbol{I}_n\right) = -Q - K_{\text{q}}^\top R K_{\text{q}}, \tag{3.4.5}$$

with Q and R the weighting matrices as employed in (3.3.3). The qMPC design also uses the terminal weight matrix P_{q} to formulate the terminal constraint set given as follows:

$$\Omega_\alpha(\boldsymbol{x}_e) = \{\boldsymbol{x} \in \mathbb{R}^n | (\boldsymbol{x} - \boldsymbol{x}_e)^\top P_{\text{q}}(\boldsymbol{x} - \boldsymbol{x}_e) \leq \alpha\}, \tag{3.4.6}$$

in which, the parameter $\alpha \in \mathbb{R}_+$ is chosen as the largest possible value such that the followings hold:

$$\Omega_\alpha(\boldsymbol{x}_e) \subseteq \mathcal{X}, \tag{3.4.7}$$
$$\boldsymbol{u}_{\text{l,q}}(\boldsymbol{x}) \in \mathcal{U},\ \forall \boldsymbol{x} \in \Omega_\alpha(\boldsymbol{x}_e), \tag{3.4.8}$$
$$(\boldsymbol{x} - \boldsymbol{x}_e)^\top P_{\text{q}}[f(\boldsymbol{x}, \boldsymbol{u}_{\text{l,q}}(\boldsymbol{x})) - A_{K_{\text{q}}}(\boldsymbol{x} - \boldsymbol{x}_e)] \leq \kappa(\boldsymbol{x} - \boldsymbol{x}_e)^\top P_{\text{q}}(\boldsymbol{x} - \boldsymbol{x}_e),\ \forall \boldsymbol{x} \in \Omega_\alpha(\boldsymbol{x}_e), \tag{3.4.9}$$

with $\mathcal{X}, \mathcal{U}$ the state and input constraint sets in (3.1.3), $\boldsymbol{u}_{\text{l,q}}(\boldsymbol{x})$ the linear controller as in (3.4.2), $f(\cdot)$ the system dynamics as in (3.1.1), $A_{K_{\text{q}}}$, κ as in (3.4.4) and P_{q} from (3.4.5).

Remark 3.4.1. The quasi-infinite horizon NMPC design proposed in [Chen and Allgöwer, 1998] makes use of a local linear feedback controller, hence, being capable of dealing with any general nonlinear system as in (3.1.1). However, employing a linear controller for a nonlinear system is not always appropriate and obviously restricts the corresponding invariant set Ω_α as in (3.4.6). The complexity of finding an acceptable value of the "radius" α satisfying (3.4.7)–(3.4.9) is significant in comparison with the proposed approach, i.e. finding the control gain matrix K as in (3.2.31) to guarantee the invariant property and scale the radius ε as in (3.2.32) for the invariant set $\mathcal{B}_K(q_e, \varepsilon)$ given in (3.5.24) to be constraint admissible. We consider these results evolutionary and differ from the qMPC approach [Chen and Allgöwer, 1998] as delineated in Table 3.4.1.

3.4.2 Simulation results

For the designing of the proposed NMPC scheme, we follow Procedure 3.3.2 to establish the optimization problem (3.1.4)–(3.1.6). In order to compare and analyze the performances of different controllers, we consider three scenarios with the common goal of stabilizing the inverted pendulum (3.4.1) where we change the control law as follows:

Scenario 1: The CTC controller $\boldsymbol{u}_{\text{CTC}}(\boldsymbol{x}, K_1\boldsymbol{x})$ from (3.2.6) with the control gain matrix K_1 chosen as in (3.2.31) such that the resulted invariant set $\mathcal{X}_{f_1} \triangleq \mathcal{B}_{K_1}(q_e, \varepsilon)$ (3.5.24) contains the initial state $\boldsymbol{x}_0$, thus, showing its positive invariant property.

Table 3.4.1: Differences in desing principles between quasi-infinite horizon NMPC and the proposed NMPC scheme.

Ingredients	Quasi-infininite horizon NMPC [Chen and Allgöwer, 1998]	NMPC design using local CTC controller
Conditions **C1**-**C4** in Lemma 3.1.1	Origin	Usage
Local controller	Linear controller (3.4.2) $\boldsymbol{u}_{\mathrm{l,q}}(\boldsymbol{x}) = K_{\mathrm{q}}(\boldsymbol{x} - \boldsymbol{x}_e) + \boldsymbol{u}_e$	CTC controller $\boldsymbol{u}_{\mathrm{CTC}}(\boldsymbol{x}, K\boldsymbol{x})$ (3.2.6)
Weighting matrix P	Obtained from (3.4.5), required as prerequisite for further analysis	Obtained from (3.3.5) as final step of the design procedure
Input constraint satisfaction	Scaling a predefined ellipsoid region Ω_α (3.4.6) by choosing α satisfying (3.4.7)-(3.4.9)	Approximating $\boldsymbol{u}_{\mathrm{CTC}}(\boldsymbol{x})$ by using Taylor's theory
State constraint		Scaling $\boldsymbol{\varepsilon}$ (3.2.32)
Invariant set		Choosing K (3.2.31)
Condition **C4** (3.1.10) satisfaction	1) $\|\boldsymbol{u}_{\mathrm{l,q}} - \boldsymbol{u}_e\|_R^2 = \|\boldsymbol{x} - \boldsymbol{x}_e\|^2_{K_{\mathrm{q}}^\top R K_{\mathrm{q}}}$ 2) $\frac{d}{dt}\|\boldsymbol{x} - \boldsymbol{x}_e\|^2_{P_{\mathrm{q}}}$ is bounded by using (3.4.9) 3) Terminal weighting matrix P_{q} satisfies Lyapunov equation (3.4.5)	1) $\|\boldsymbol{u}_{\mathrm{CTC}} - \boldsymbol{u}_e\|_R^2 \leq \|\boldsymbol{x} - \boldsymbol{x}_e\|^2_{R^*}$ 2) $\frac{d}{dt}\|\boldsymbol{x} - \boldsymbol{x}_e\|^2_P = \|\boldsymbol{x} - \boldsymbol{x}_e\|^2_{A_K^\top P + P A_K}$ 3) Terminal weighting matrix P satisfies Lyapunov equation (3.3.5)

Scenario 2: The proposed NMPC design (3.1.4)–(3.1.6) with the gain matrix K_2 chosen as in (3.2.31), different from K_1 used in Scenario 1 such that the terminal region $\mathcal{X}_{f_2} \triangleq \mathcal{B}_{K_2}(q_e, \varepsilon)$ (3.5.24) does not contain the initial state $\boldsymbol{x}_0$. Note that, we keep the same value of the radius ε under both Scenarios 1 and 2. The aim is to show the ease to tune the orientation of the terminal constraint set by changing the gain matrix since scaling ε is straightforward.

Scenario 3: The qMPC controller recalled in Section 3.4.1.

All the design parameters concerning the three Scenarios are gathered in Table 3.4.2.

Regarding the prediction model as in (3.1.6a), we employ the Runge-Kutta 4-order discretization method to discretize the cart-pendulum dynamics (3.4.1) with a step $\Delta = 0.1$ seconds. The NMPC sampling time is also fixed at $\delta = 0.1$ seconds making the formulation identical with a standard discrete MPC setup. The optimization problem (3.1.4) is solved by using solver IPOPT [Wächter and Biegler, 2006] within Matlab version R2015a. Note that in the presented figures, the variables concerning Scenario 1 are plotted in blue, for Scenario 2 in red and for Scenario 3 in green. Furthermore, in the figures labels, we denote our proposed approach by NMPC and the quasi-infinite horizon NMPC by qMPC.

Figure 3.4.1 illustrates the three terminal regions under three scenarios, $\mathcal{X}_{f_1} = \mathcal{B}_{K_1}(\mathbf{0}, \varepsilon)$ (blue ellipsoid), $\mathcal{X}_{f_2} = \mathcal{B}_{K_2}(\mathbf{0}, \varepsilon)$ (red ellipsoid) with $B_{K_1}(\cdot), B_{K_2}(\cdot)$ from (3.5.24) and Ω_α from (3.4.6) (green ellipsoid). We observe that the proposed terminal regions, $\mathcal{X}_{f_1}$ and $\mathcal{X}_{f_2}$, are significantly larger than the terminal region Ω_α. Furthermore, by tuning the gain matrix from $K_1 = [-0.5 \quad -0.55]$ to $K_2 = [-1 \quad -0.6]$, the invariant set $\mathcal{B}_{K_1}(\mathbf{0}, \varepsilon)$ under Scenario 1 is easily modified into $\mathcal{B}_{K_2}(\mathbf{0}, \varepsilon)$ under Scenario 2, both staying within the constraint set $\mathcal{X}$ (black rectangle). These results illustrate the ease of modifying the terminal region of the proposed NMPC method which is not the case for the qMPC design under Scenario 3. Indeed, it is difficult to predict the shape of the invariant set Ω_α from (3.4.6) under the linear controller $\boldsymbol{u}_{\mathrm{l,q}}$ in

Table 3.4.2: Design parameters for stabilizing the pendulum

	Parameters	Values
Common parameters	NMPC sampling time δ from (3.1.7)	0.1 sec
	Model discretization step Δ from (3.1.12)	0.1 sec
	Weighting matrices Q, R as in (3.1.5)	diag$\{5, 1\}$, 1
	Jacobian matrices ${}^{\boldsymbol{x}}J$, ${}^{\mu}J$ as in (3.2.18)	[3.1735 0], -0.0349
	$\mathcal{C}$ as in (3.2.22)	3.1737
	$M_{\boldsymbol{\varepsilon}}$ as in (3.2.20)	1.1882
	Radius $\varepsilon = \varepsilon_{\max}$ as in (3.2.24)	0.1793
Scenario 1	Gain matrix K_1 as in (3.2.31)	$[-0.5 \ -0.55]$
Scenario 2	Prediction horizon T_{p_2} as in (3.1.5)	0.4 *sec*
	Gain matrix K_2 as in (3.2.31)	$[-1 \ -0.6]$
	R^* as in (3.3.6)	[21.52 6.46; 6.46 14.63]
	Terminal weighting matrix P as in (3.3.5)	[36.63 13.26; 13.26 35.13]
Scenario 3	Prediction horizon T_{p_3} as in (3.1.5)	0.6 *sec*
	Gain matrix K_{q} as in (3.4.4)	[7.0557 1.2216]
	κ as in (3.4.4)	3
	Terminal weighting matrix P_{q} as in (3.4.5)	[29.90 1.05; 1.05 0.072]
	α as in (3.4.7)-(3.4.9)	0.003

(3.4.2) since the matrix P_{q} is obtained as the solution of the Lyapunov equation (3.4.5) and there is no available information of α until the final design step (3.4.7)–(3.4.9). These observations again confirm that applying a linear controller for a general system (and, in particular, for the considered inverted pendulum) restricts the size of the resulted invariant set and furthermore, causes difficulties in employing it as a terminal constraint set in the NMPC design. By using a larger terminal region $\mathcal{X}_{f_2}$, our proposed NMPC controller can be executed with any prediction horizon larger than 0.2 seconds (we employ $T_2 = 0.4$ seconds for better performances), while the qMPC in Scenario 3 does not accept any prediction horizon smaller than $T_3 = 0.6$ seconds as shown in Table 3.4.2.

The positive invariant property of the set $\mathcal{X}_{f_1}$ (as proved in Lemma 3.2.4) is illustrated in Figure 3.4.1 by the dash-dotted blue cyclic trajectory under Scenario 1 resulted from the CTC controller $\boldsymbol{u}_{\mathrm{CTC}}(\boldsymbol{x}, K_1\boldsymbol{x})$ (3.2.6) (with the input values given in dashed-dotted blue line in Figure 3.4.2b) which always lies inside the invariant set $\mathcal{X}_{f_1}$.

Regarding the performances of the two NMPC controllers, as observed from Figure 3.4.2a, both our proposed NMPC scheme under Scenario 2 (red lines) and the qMPC controller under Scenario 3 (green lines) obtain the similar convergence times of 2 seconds. This is mostly due to the fact that both qMPC and NMPC designs employ the same matrices Q, R as shown in Table 3.4.2. We note that, the performance of the proposed NMPC controller can be enhanced by choosing more appropriate matrices Q and R (e.g., larger values in Q). All the states and inputs of the three controllers under the three scenarios satisfy the system constraint as shown in Figures 3.4.1–3.4.2b which are due to the design of the constraint admissible set $\mathcal{X}_{f_1}$ under Scenario 1 and the usage of the NMPC controllers under the two other scenarios. Next, Figure 3.4.3 presents the computation times per step for Scenarios 2 and 3. Regarding the proposed NMPC controller (red line with circle marks), after the first several iterations, the computation times stabilize around 0.15 seconds, which is significantly less than those corresponding to the qMPC under Scenario 3 (green line with triangle marks). These differences imply the advantage

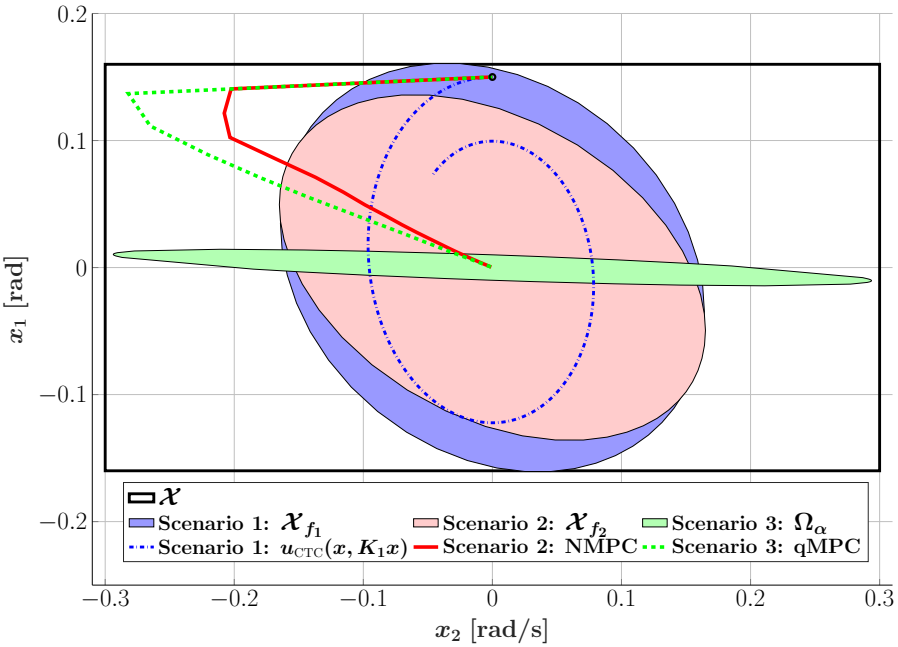
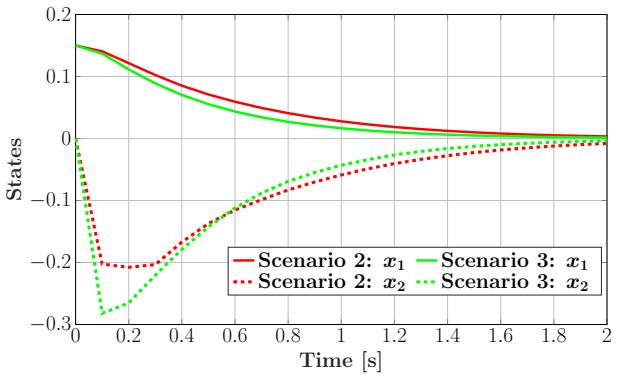
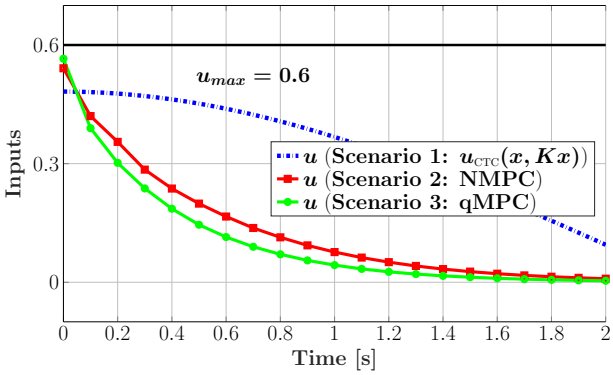


Figure 3.4.1: Terminal regions $\mathcal{X}_{f_1}, \mathcal{X}_{f_2}$ and Ω_α and state trajectories under different scenarios.



(a) Convergence of states.



(b) Values of the input u.

Figure 3.4.2: States and inputs under different scenarios.

of having the larger terminal region $\mathcal{X}_{f_2}$ (red ellipsoid in Figure 3.4.1), as resulted from applying the CTC controller $u_{\text{CTC}}(x)$ (3.2.6) within the NMPC scheme when compared to the usage of the local linear controller $u_{\text{l,q}}(x)$ as in (3.4.2) for the qMPC strategy.

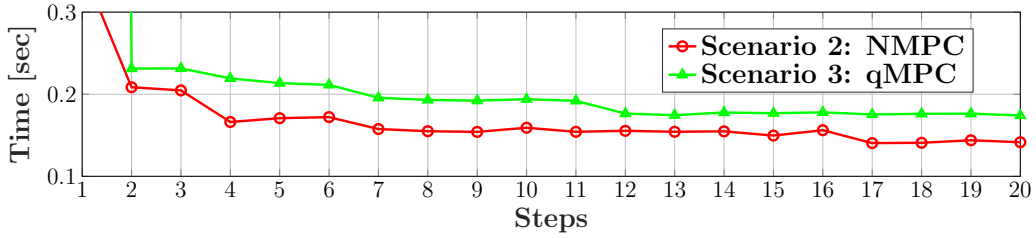


Figure 3.4.3: Computing time under the two Scenarios 2 and 3.

By this example on stabilizing the cart-pendulum system, we have illustrated the NMPC design presented in this Chapter which is particularized for a "computed-torque like" system by using its available CTC controller. Our proposed construction including: i) the upper bound of the CTC law through Taylor's approximation as detailed in Section 3.2.2; ii) the bound of

the input term as given in (3.2.28); and iii) the invariant set construction by employing the linear system as presented in Section 3.2.3, can be generalized for any feedback linearization controller. Hence, using these results, various applications of nonlinear controllers for stabilizing NMPC schemes can be conducted.

In the following section, we apply the proposed NMPC scheme to design the *attitude controller* at the low control level for the multicopter system (c.f. Figure 2.3.1). More precisely, the controller provides the torque for the rotation dynamics (2.1.1) to track the angle set-points received from the high control level. This significantly increases the complexity of setting up the controller due to the fact that the terminal constraint set $\mathcal{B}_K(q_e, \varepsilon)$ from (3.5.24) and the terminal weighting matrix P from (3.3.5) are constructed based on the desired equilibrium point q_e. A similar problem also occurs for the quasi-infinite horizon NMPC design as it is constructed based on the system linearized around the equilibrium. Therefore, the next section will present how we reduce the dependence of the controller setup on the desired equilibrium q_e in order to overcome the aforementioned difficulties.

3.5 Application on attitude control for a multicopter system

This section presents an NMPC attitude controller [Nguyen et al., 2020b] which solves the control problem presented in Section 2.3.2: to stabilize the rotation dynamics around the desired equilibrium point defined as:

$$\eta_e = \eta_d,\ \omega_e = \mathbf{0},\ \tau_e = \mathbf{0}, \tag{3.5.1}$$

in which the angle reference $\eta_d = [\phi_d\ \theta_d\ \psi_r]^\top$ is defined in (2.3.5) and is supposed to be piece-wise constant.

For the reader to easily follow, we will briefly recapitulate the rotation dynamics and the required constraints which are required within the NMPC setup. At first, the rotation dynamics are partly taken from the full dynamics of the multicopter system (2.1.11):

$$\dot{\eta} = \overline{W}\omega, \tag{3.5.2a}$$
$$\dot{\omega} = J^{-1}\left(-\omega \times (J\omega) + \tau\right), \tag{3.5.2b}$$

with $\eta = [\phi\ \theta\ \psi]^\top \in \mathbb{R}^3$ gathering the three Euler angles, $\omega = [\omega_x\ \omega_y\ \omega_z]^\top \in \mathbb{R}^3$ the angle rate vector, $\tau \in \mathbb{R}^3$ the torque vector[3] and $J = \mathrm{diag}\{J_x, J_y, J_z\}$ the inertial tensor. The matrix $\overline{W} = W^{-1} \in \mathbb{R}^{3\times 3}$ with W as in (2.1.8) is explicitly given by:

$$\overline{W} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix}. \tag{3.5.3}$$

Furthermore, system (3.5.2) is subject to its state and input constraints as follows:

$$\mathcal{X}_{\mathrm{rot}} = \Big\{ \langle |\phi|, |\theta| \rangle \leq \epsilon_{\max},\ \langle |\omega_x|, |\omega_y| \rangle \leq \omega_{\max} \Big\}, \tag{3.5.4}$$
$$\mathcal{U}_{\mathrm{rot}} = \Big\{ |\tau| \leq \tau_{\max} \Big\}, \tag{3.5.5}$$

[3]The input constraint (3.5.5) will be enforced by the NMPC controller, hence, leading to no difference between the desired torque law τ_d and the input torque τ as being distinguished in (2.3.6).

with $\epsilon_{\max} \in (0, \pi/2)$ the given maximum angle value, $\omega_{\max} \in \mathbb{R}_+$ the maximum angle rate and $\tau_{\max} \in \mathbb{R}^3_+$ gathering the maximum values of three torques on three axes as defined in (2.1.12)-(2.1.13). Note that, the condition $\langle |\phi|, |\theta| \rangle \leq \epsilon_{\max} < \pi/2$ is sufficient to avoid singularities of the matrix $\overline{W}$ (3.5.3) which happens at the perpendicular position, i.e., $\phi = \pi/2$ and $\theta = \pi/2$. In the followings, we apply the NMPC design particularized for the "computed-torque like" systems proposed in Section 3.4 for the rotation dynamics (3.5.2). We give the details about how to construct the NMPC parameters required in Procedure 3.3.2. This has the scope of reducing their dependence on the reference η_d received from the high control level in order to decrease the complexity of the controller.

3.5.1 Parameters design for the NMPC attitude controller

In order to apply the proposed NMPC design with stability induced by a CTC law, we firstly transform the dynamics (3.5.2) into their "computed-torque like" representation by using the relation $\omega = W\dot{\eta}$ from (2.1.7):

$$JW\ddot{\eta} + J\dot{W}\dot{\eta} + (W\dot{\eta}) \times (JW\dot{\eta}) = \tau, \tag{3.5.6}$$

with J as in (3.5.2) and $W \in \mathbb{R}^{3\times 3}$ depending on η as given in (2.1.8) and recapitulated below:

$$W(\eta) = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix}. \tag{3.5.7}$$

Note that the controlled state is the three angles $\eta = [\phi\ \theta\ \psi]^\top \in \mathbb{R}^3$ and the control input is the torque vector $\tau \in \mathbb{R}^3$ (instead of the general state q and input $\boldsymbol{u}$ as employed in (3.2.1)). The CTC law of the dynamics (3.5.6) is given by [Nguyen et al., 2017b, Nguyen et al., 2020b]:

$$\tau_{\text{CTC}}(\eta, \dot{\eta}, \mu_\eta) = JW\mu_\eta + J\dot{W}\dot{\eta} + (W\dot{\eta}) \times (JW\dot{\eta}), \tag{3.5.8}$$

with $\mu_\eta \in \mathbb{R}^3$ the virtual input vector. From (3.5.1), the equilibrium values are:

$$\eta_e = \eta_d,\ \dot{\eta}_e = \mathbf{0},\ \tau_e = \mathbf{0},\ \mu_{\eta_e} = \mathbf{0}, \tag{3.5.9}$$

with η_d the desired angles received from the higher control level. Following the design procedure given in Section 3.4, the first step is to find the maximum radius $\varepsilon_{\max}$ such that the set $\mathcal{B}(\eta_e, \varepsilon_{\max})$ from (3.2.15) is input constraint admissible with respect to the constraints $|\tau_{\text{CTC}}(\eta, \dot{\eta}, \mu_\eta)| \leq \tau_{\max}$ as in (3.5.5). To better emphasize the result, the set $\mathcal{B}(\cdot)$ from (3.2.15) is re-written explicitly in terms of the depending variables $(\eta, \dot{\eta}, \mu_\eta)$ as follows:

$$\mathcal{B}(\eta_d, \varepsilon) = \left\{ (\eta, \dot{\eta}, \mu_\eta) \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3 \,\middle|\, \|\eta - \eta_d\|^2 + \|\dot{\eta}\|^2 + \|\mu\|^2 \leq \varepsilon^2 \right\}. \tag{3.5.10}$$

The Taylor's approximation of the CTC law $\tau_{\text{CTC}}(\eta, \dot{\eta}, \mu_\eta)$ (3.5.8) around the equilibrium (3.5.9) within the ball $\mathcal{B}(\cdot)$ is given by (c.f. the general Taylor's approximation detailed in (3.2.18)):

$$\tau_{\text{CTC}}(\eta, \dot{\eta}, \mu_\eta) = \tau_e + {}^{\eta}J(\eta - \eta_d) + {}^{\dot{\eta}}J(\dot{\eta} - \dot{\eta}_e) + {}^{\mu_\eta}J(\mu_\eta - \mu_{\eta_e}) + \mathcal{R}(\eta, \dot{\eta}, \mu_\eta, \eta_d, \varepsilon), \tag{3.5.11}$$

with the Jacobian matrices (all in $\mathbb{R}^{3\times 3}$ space) given by:

$${}^{\eta}J = \left.\frac{\partial \tau_{\text{CTC}}}{\partial \eta}\right|_{(\eta_d, \mathbf{0}, \mathbf{0})} = \mathbf{0},\ {}^{\dot{\eta}}J = \left.\frac{\partial \tau_{\text{CTC}}}{\partial \dot{\eta}}\right|_{(\eta_d, \mathbf{0}, \mathbf{0})} = \mathbf{0},\ {}^{\mu_\eta}J = \left.\frac{\partial \tau_{\text{CTC}}}{\partial \mu_\eta}\right|_{(\eta_d, \mathbf{0}, \mathbf{0})} = JW(\eta_d), \tag{3.5.12}$$

in which $W(\eta_d)$ is obtained by introducing $\eta = \eta_d$ to (3.5.7). Then, by applying the triangle and Cauchy-Schwarz inequalities, the CTC law $\tau_{\text{CTC}}(\eta, \dot{\eta}, \mu_\eta)$ from (3.5.11) is bounded by:

$$|\tau_{\text{CTC}}(\eta, \dot{\eta}, \mu_\eta)| \leq |JW(\eta_d)|\|\mu_\eta\|^2 + |\mathcal{R}(\eta, \dot{\eta}, \mu_\eta, \eta_d, \varepsilon)|, \tag{3.5.13}$$

in which, $|JW(\eta_d)| \leq C$ with the vector $C \in \mathbb{R}^3$ given by:

$$C = \begin{bmatrix} J_x\sqrt{1+\sin^2\epsilon_{\max}} & J_y & J_z \end{bmatrix}^\top, \tag{3.5.14}$$

with $\epsilon_{\max}$ the maximum angle value as in (3.5.4). Furthermore, the remainder term $\mathcal{R}(\eta, \dot{\eta}, \mu_\eta, \eta_d, \varepsilon)$ is bounded as in (3.2.19), however, this approach requires to solve (3.2.20) for $\mathcal{M}(\eta_d)$ at each step. Therefore, in the following, we introduce an alternative way to calculate the constant term $\mathcal{M} \in \mathbb{R}^3$ which becomes independent from the desired reference angle η_d.

Proposition 3.5.1 ([Nguyen et al., 2020b]). *The remainder term $\mathcal{R}(\eta, \dot{\eta}, \mu_\eta, \eta_d, \varepsilon)$ of the Taylor's approximation of the CTC law $\tau_{CTC}(\eta, \dot{\eta}, \mu_\eta)$ as in (3.5.11) is bounded as follows*[4]*:*

$$|\mathcal{R}(\eta, \dot{\eta}, \mu_\eta, \eta_d, \varepsilon)| \leq \frac{\mathcal{M}}{2}\left(\|\eta - \eta_d\|^2 + \|\dot{\eta}\|^2 + \|\mu_\eta\|^2\right), \tag{3.5.15}$$

with the vector $\mathcal{M} \in \mathbb{R}^3$ given as follows:

$$\mathcal{M} = \begin{bmatrix} J_x + 2|J_z - J_y| \\ 2J_y + 2\sqrt{1+\sin^2\epsilon_{max}}|J_z - J_x| \\ 2J_z + 2\sqrt{1+\sin^2\epsilon_{max}}|J_x - J_y| \end{bmatrix}, \tag{3.5.16}$$

with ϵ_{max} the maximum angle value in (3.5.4) and (J_x, J_y, J_z) the moments of inertia along the three axes of the multicopter system as in (3.5.6).

Proof. The proof is constructed by exploiting the explicit formulation of the remainder term $R_\varepsilon \triangleq [R_{\varepsilon,1}\ R_{\varepsilon,2}\ R_{\varepsilon,3}]^\top$ derived from (3.5.11):

$$R_{\varepsilon,1} = -J_x\Big((\mathrm{s}\,\theta - \mathrm{s}\,\theta_d)\mu_\psi + \dot{\theta}\dot{\psi}\,\mathrm{c}\,\theta\Big) + (J_z - J_y)(\dot{\theta}\,\mathrm{c}\,\phi + \dot{\psi}\,\mathrm{s}\,\phi\,\mathrm{c}\,\theta)(-\dot{\theta}\,\mathrm{s}\,\phi + \dot{\psi}\,\mathrm{c}\,\phi\,\mathrm{c}\,\theta), \tag{3.5.17}$$

$$\begin{aligned} R_{\varepsilon,2} =& J_y\Big((\mathrm{c}\,\phi - \mathrm{c}\,\phi_d)\mu_\theta + (\mathrm{s}\,\phi\,\mathrm{c}\,\theta - \mathrm{s}\,\phi_d\,\mathrm{c}\,\theta_d)\mu_\psi - \dot{\phi}\dot{\theta}\,\mathrm{s}\,\phi + \dot{\psi}(\dot{\theta}\,\mathrm{c}\,\phi\,\mathrm{c}\,\theta - \dot{\theta}\,\mathrm{s}\,\phi\,\mathrm{s}\,\theta)\Big) \\ &+ (J_x - J_z)\Big(\dot{\phi} - \mathrm{s}\,\theta\dot{\psi}\Big)\Big(-\mathrm{s}\,\phi\dot{\theta} + \mathrm{c}\,\phi\,\mathrm{c}\,\theta\dot{\psi}\Big), \end{aligned} \tag{3.5.18}$$

$$\begin{aligned} R_{\varepsilon,3} =& J_z\Big(-(\mathrm{s}\,\phi - \mathrm{s}\,\phi_d)\mu_\theta + (\mathrm{c}\,\phi\,\mathrm{c}\,\theta - \mathrm{c}\,\phi_d\,\mathrm{c}\,\theta_d)\mu_\psi - \dot{\phi}\dot{\theta}\,\mathrm{c}\,\phi - \dot{\psi}\dot{\phi}\,\mathrm{s}\,\phi\,\mathrm{c}\,\theta - \dot{\psi}\dot{\theta}\,\mathrm{c}\,\phi\,\mathrm{s}\,\theta\Big) \\ &+ (J_y - J_x)(\dot{\phi} - \mathrm{s}\,\theta\dot{\psi})(\mathrm{c}\,\phi\dot{\theta} + \mathrm{s}\,\phi\,\mathrm{c}\,\theta\dot{\psi}), \end{aligned} \tag{3.5.19}$$

with $\mathrm{s}(\cdot)$ and $\mathrm{c}(\cdot)$ representing the $\sin(\cdot)$ and $\cos(\cdot)$ functions, respectively. As detailed in Appendix B, from (3.5.17)–(3.5.19), we have that:

$$\begin{aligned} |R_{\varepsilon,1}| &\leq \frac{J_x}{2}\Big((\theta - \theta_d)^2 + \mu_\psi^2 + \dot{\phi}^2 + \dot{\psi}^2\Big) + |J_z - J_y|(\dot{\theta}^2 + \dot{\psi}^2)^2 \\ &\leq \left(\frac{J_x}{2} + J_z - J_y\right)\left(\|\eta - \eta_d\|^2 + \|\dot{\eta}\|^2 + \|\mu_\eta\|^2\right), \end{aligned} \tag{3.5.20}$$

$$\begin{aligned} |R_{\varepsilon,2}| &\leq \frac{J_y}{2}\left(2(\phi - \phi_d)^2 + (\theta - \theta_d)^2 + \mu_\theta^2 + 2\mu_\psi^2 + \|\dot{\eta}\|^2\right) + |J_z - J_x|\sqrt{1+\mathrm{s}^2\,\epsilon_{\max}}\,\|\dot{\eta}\|^2, \\ &\leq \left(J_y + |J_z - J_x|\sqrt{1+\mathrm{s}^2\,\epsilon_{\max}}\right)\left(\|\eta - \eta_d\|^2 + \|\dot{\eta}\|^2 + \|\mu_\eta\|^2\right), \end{aligned} \tag{3.5.21}$$

[4]Note that, $|\boldsymbol{x}| = [|x_1|\ |x_2|\ldots|x_n|]^\top$ for all $\boldsymbol{x} \triangleq [x_1\ x_2\ldots x_2]^\top \in \mathbb{R}^n$.

$$|R_{\varepsilon,3}| \leq \frac{J_z}{2}\left(2(\phi-\phi_d)^2+(\theta-\theta_d)^2+\mu_\theta^2+2\mu_\psi^2+\|\dot{\eta}\|^2\right)+|J_y-J_x|\sqrt{1+\mathrm{s}^2\,\epsilon_{\max}}\,\|\dot{\eta}\|^2,$$
$$\leq \left(J_z+|J_x-J_y|\sqrt{1+\mathrm{s}^2\,\epsilon_{\max}}\right)\left(\|\eta-\eta_d\|^2+\|\dot{\eta}\|^2+\|\mu_\eta\|^2\right), \tag{3.5.22}$$

with $\epsilon_{\max}$ the maximum angle value as constrained in (3.5.4). Then, by re-formulating (3.5.20)–(3.5.22) into (3.5.15), the vector $\mathcal{M}$ is obtained as given in (3.5.16). □

Therefore, by introducing (3.5.15) to (3.5.13), the maximum radius $\varepsilon_{\max}$ which ensures that the set $\mathcal{B}(\eta_d,\varepsilon_{\max})$ as in (3.2.16) is input constraint admissible is calculated by finding the largest possible $\varepsilon_{\max}>0$ such that:

$$C\varepsilon_{\max}+\frac{\mathcal{M}}{2}\varepsilon_{\max}^2 \leq \tau_{\max}, \tag{3.5.23}$$

with C as in (3.5.14), $\mathcal{M}$ given in (3.5.16) and $\tau_{\max}$ the maximum torque values as in (3.5.5). The next step is to choose the control gain matrix $K\in\mathbb{R}^{3\times 6}$ as in (3.2.11) satisfying the conditions (3.2.31) in order to construct the positive invariant set $\mathcal{B}_K(\eta_d,\varepsilon)$ similarly to (3.2.30) and which is explicitly given by:

$$\mathcal{B}_K(\eta_d,\varepsilon)=\left\{(\eta,\dot{\eta})\in\mathbb{R}^3\times\mathbb{R}^3\,\middle|\,\|\eta-\eta_d\|^2+\|\dot{\eta}\|^2+\begin{bmatrix}\eta-\eta_d\\ \dot{\eta}\end{bmatrix}^\top K^\top K\begin{bmatrix}\eta-\eta_d\\ \dot{\eta}\end{bmatrix}\leq\varepsilon^2\right\}. \tag{3.5.24}$$

Then, at each time step, the radius ε needs to be scaled corresponding to the desired angles η_d received from the high control level such that the conditions (3.2.32) are satisfied in order to make the set $\mathcal{B}_K$ in (3.5.24) constraint admissible. However, the second requirement of (3.2.32), i.e., $\mathcal{B}_K(\eta_d,\varepsilon)\subseteq\mathcal{X}_{\text{rot}}$ (the state constraint set $\mathcal{X}_{\text{rot}}=\left\{\langle|\phi|,|\theta|\rangle\leq\epsilon_{\max},\ \langle|\omega_x|,|\omega_y|\rangle\leq\omega_{\max}\right\}$ as in (3.5.4)) brings difficulties due to the nonlinear relation $\omega=W\dot{\eta}$ from (2.1.7). Therefore, in the following, we propose an alternative condition on $\dot{\eta}$ which ensures the satisfaction of $\langle|\omega_x|,|\omega_y|\rangle\leq\omega_{\max}$ from (3.5.4).

Proposition 3.5.2 ([Nguyen et al., 2020b]). *The state conditions on angular velocities:* $\langle|\omega_x|,|\omega_y|\rangle\leq\omega_{max}$ *as in* (3.5.4) *are satisfied if the following holds:*

$$\|\dot{\eta}\|\leq\frac{\omega_{max}}{\sqrt{1+\sin^2\epsilon_{max}}}, \tag{3.5.25}$$

with ϵ_{max} *the maximum angle value from* (3.5.4).

Proof. From $\omega=W\dot{\eta}$ given in (2.1.7), we have that:

$$\omega_x=\dot{\phi}-\dot{\psi}\sin\theta, \tag{3.5.26}$$
$$\omega_y=\dot{\theta}\cos\phi+\dot{\psi}\sin\phi\cos\theta. \tag{3.5.27}$$

Next, applying Cauchy-Schwarz inequality to (3.5.26)–(3.5.27) leads to:

$$|\omega_x|\leq\sqrt{(1+\sin^2\theta)(\dot{\phi}^2+\dot{\psi}^2)}\leq\sqrt{1+\sin^2\epsilon_{\max}}\|\dot{\eta}\|, \tag{3.5.28}$$
$$|\omega_y|\leq\sqrt{(\cos^2\phi+\sin^2\phi\cos^2\theta)(\dot{\theta}^2+\dot{\psi}^2)}\leq\|\dot{\eta}\|, \tag{3.5.29}$$

in which, the second inequality of (3.5.28) is due to $|\theta|\leq\epsilon_{\max}$ as constrained in (3.5.4) and to the fact that sin is monotonously increasing on the interval $[0,\pi/2]$. Next, (3.5.29) comes by using $\cos^2\phi+\sin^2\phi\cos^2\theta\leq\cos^2\phi+\sin^2\phi=1$. Therefore, introducing the proposed condition (3.5.25) to (3.5.28)–(3.5.29) ultimately leads to $\langle|\omega_x|,|\omega_y|\rangle\leq\omega_{\max}$. □

Next, we propose a method to efficiently choose the radius ε such that the requirements in (3.2.32) are satisfied.

Proposition 3.5.3 ([Nguyen et al., 2020b]). *The invariant set $\mathcal{B}_K(\eta_d, \varepsilon)$ given in (3.5.24) is admissible for both the input and state constraints given in (3.5.4)–(3.5.5) for any control gain matrix K satisfying (3.2.31) if the radius ε is chosen such that:*

$$\varepsilon \leq \min\left\{\varepsilon_{max}, \epsilon_{max} - \max\{|\phi_d|, |\theta_d|\}, \frac{\omega_{max}}{\sqrt{1+\sin^2\epsilon_{max}}}\right\}, \tag{3.5.30}$$

with ε_{max} the maximum radius as in (3.5.23), ϵ_{max}, ω_{max} the maximum angle and angular velocity as in (3.5.4) and ϕ_d, θ_d the desired roll, pitch angles sent from high control level as in (3.5.1).

Proof. At first, $\varepsilon \leq \varepsilon_{\max}$ is as required in (3.2.32) for ensuring the input constraint admissible property. Next, introducing $\varepsilon \leq \epsilon_{\max} - \max\{\phi_d, \theta_d\}$ to the formulation of $\mathcal{B}_K(\eta_d, \varepsilon)$ as in (3.5.24) leads to:

$$\|\eta - \eta_d\| \leq \epsilon_{\max} - \max\{|\phi_d|, |\theta_d|\}, \tag{3.5.31}$$

which further provides:

$$\langle |\phi - \phi_d|, |\theta - \theta_d| \rangle \leq \epsilon_{\max} - \max\{|\phi_d|, |\theta_d|\}. \tag{3.5.32}$$

This leads to the satisfaction of the angle constraints as required in (3.5.4):

$$|\phi| \leq |\phi - \phi_d| + |\phi_d| \leq |\phi - \phi_d| + \max\{|\phi_d|, |\theta_d|\} \leq \epsilon_{\max}, \tag{3.5.33}$$

$$|\theta| \leq |\theta - \theta_d| + |\theta_d| \leq |\theta - \theta_d| + \max\{|\phi_d|, |\theta_d|\} \leq \epsilon_{\max}. \tag{3.5.34}$$

Next, constraining $\varepsilon \leq \omega_{\max}/\sqrt{1+\sin^2\epsilon_{\max}}$ is to obtain the condition $\|\dot{\eta}\| \leq \omega_{\max}/\sqrt{1+\sin^2\epsilon_{\max}}$ as in (3.5.25), hence, Proposition 3.5.2 provides $\langle |\omega_x|, |\omega_y| \rangle \leq \omega_{\max}$ as required in (3.5.4), completing the proof. □

By Proposition 3.5.3, we achieve the construction of the constraint admissible and positive invariant set $\mathcal{B}_K(\eta_d, \varepsilon)$ given in (3.5.24) as required for designing the NMPC controller. Next, we provide the details on how to choose the terminal weighting matrix P as employed in (3.3.4). The general approach is to solve the Lyapunov equation (3.3.5), however, the Lipschitz bound L (more precisely, L_{CTC} from (3.2.28) since the CTC law is used as the local controller) is dependent of the desired angle η_d and the radius ε as detailed in (3.2.28). Then, the symmetric matrix R^* as in (3.3.6) satisfying $R^* \succeq \max(\text{eig}(R))\mathsf{L}_{\text{CTC}}(\boldsymbol{I}_n + K^\top K)$ is re-computed at each time step which increases complexity when solving on-line the NMPC controller. Thus, we further detail how to choose the matrix R^* only one time but ensuring the condition (3.3.6) for all the desired angle η_d and the radius ε (which are expected to change at each time step).

Proposition 3.5.4 ([Nguyen et al., 2020b]). *Let us consider the Lipschitz bound $\mathsf{L}_{CTC}(\eta_d, \varepsilon)$ as defined in (3.2.28). For all the desired angles $|\eta_d| \leq \epsilon_{max}$ as in (3.5.4) and the radius $\varepsilon \leq \varepsilon_{max}$ as in (3.2.32), we have that:*

$$\mathsf{L}_{CTC}(\eta_d, \varepsilon) \leq \mathsf{L}_{CTC,max}, \tag{3.5.35}$$

in which, $\mathsf{L}_{CTC,max}$ is given by:

$$\mathsf{L}_{CTC,max} = \left\| C + \frac{\mathcal{M}}{2}\varepsilon_{max} \right\|^2, \tag{3.5.36}$$

with the vectors C given in (3.5.14) and $\mathcal{M}$ defined in (3.5.16). If the symmetric matrix R^* employed in (3.3.5) is chosen such that:

$$R^* \succeq \max(\text{eig}(R))\mathsf{L}_{CTC,max}(\boldsymbol{I}_n + K^\top K), \tag{3.5.37}$$

with R the weighting matrix as in (3.3.3), K the control gain matrix as in (3.2.11), then, condition (3.3.6) is satisfied for all the simulation steps.

Proof. The proof is straightforward to obtain due to the positiveness of all the elements of the vectors C as in (3.5.14) and $\mathcal{M}$ as in (3.5.16). Therefore, $\varepsilon \leq \varepsilon_{\max}$ leads to (3.5.35). Then, by introducing (3.5.35) to (3.5.37), we obtain:

$$R^* \succeq \max(\text{eig}(R))\mathsf{L}_{\text{CTC,max}}(\boldsymbol{I}_n + K^\top K) \succeq \max(\text{eig}(R))\mathsf{L}_{\text{CTC}}(\eta_d, \varepsilon)(\boldsymbol{I}_n + K^\top K), \tag{3.5.38}$$

which validates the requirement (3.3.6), hence, completing the proof. □

Remark 3.5.5. Employing $\mathsf{L}_{\text{CTC,max}}$ as in (3.5.37) does not affect the terminal constraint set (i.e., the set $\mathcal{B}_K(\eta_d, \varepsilon)$ given in (3.5.24)) but only the terminal weighting matrix P obtained from (3.3.5). This will not increase the conservativeness of the NMPC results since the matrix R^* can be freely chosen with the only requirement being the satisfaction of (3.3.6) at each NMPC steps. Therefore, by using R^* as defined in (3.5.37), we can avoid changing R^* and hence, are able to keep the same value of the terminal weighting matrix P obtained from (3.3.5) during the simulation time. This reduces the complexity of the on-line process and also the computation time per step.

3.5.2 Simulation results of the NMPC attitude controller

In this section, we present the implementation process and the simulation results of the NMPC controller designed for the rotation dynamics (3.5.6) as proposed in Section 3.5. The requirement is to track the angle references $\eta_r = [\phi_r \ \theta_r \ \psi_r]^\top$ given in Figure 2.2.5a which are obtained by the flatness-based trajectory generation procedure detailed in Section 2.2. In order to make the reference piece-wise constant, similarly to those obtained from the high control level running at lower frequency (further details can be found in Section 2.3.2), we apply the zero-order hold method with the holding time of 0.1 seconds for discretizing the smooth reference η_r (c.f. Figure 2.2.5a), hence, obtaining the roll reference ϕ_d (plotted in thin red line in Figure 3.5.1b) and also the pitch reference θ_d (plotted in thin green line) while keeping $\psi_r = 0$. The NMPC controller is simulated at the sampling time of $\delta = 0.01$ seconds while the prediction horizon is chosen as $T_p = 0.05$ seconds, thus, having 5 steps. The prediction model (3.1.6a) employed in the NMPC controller is the Forward Euler discrete model of the rotation dynamics (3.5.6) with discretization step $\Delta = 0.01$ seconds while the simulation model (to which the input torques are applied) is still in the continuous form (i.e., numerically integrated with ode45). The optimization problem in (3.1.4) is implemented with the Pyomo framework [Hart et al., 2011] and solved by with solver IPOPT [Wächter and Biegler, 2006] in Python.

At the off-line preparation stage, we find the largest possible radius $\varepsilon_{\max}$ as in (3.5.23) which further allows us to obtain the constant Lipschitz bound $\mathsf{L}_{\text{CTC, max}}$ defined in (3.5.36). Next, we choose the control gain matrix K as in (3.2.11) with the control gains satisfying the conditions (3.2.31). The next step is to choose the positive definite weighting matrices $Q \in \mathbb{R}^{6\times6}$ and $R \in \mathbb{R}^{3\times3}$, define the symmetric matrix $R^* \in \mathbb{R}^{6\times6}$ satisfying (3.5.37) and then, solve the Lyapunov equation (3.3.5) for the terminal weighting matrix $P \in \mathbb{R}^6$. All the parameters related to this off-line preparation process are gathered in Table 3.5.1. Note that the physical parameters of the rotation dynamics are given in (2.2.47)–(2.2.49) (i.e., $J = 10^{-6}\,\text{diag}\{14, 14, 22\}$ as in (3.5.6), $\epsilon_{\max} = 5°$ as in (3.5.14), $\omega_{\max} = 0.2$ as in (3.5.25) and $\tau_{\max} = [43\ 43\ 17]^\top 10^{-4}$ as in (3.5.23)).

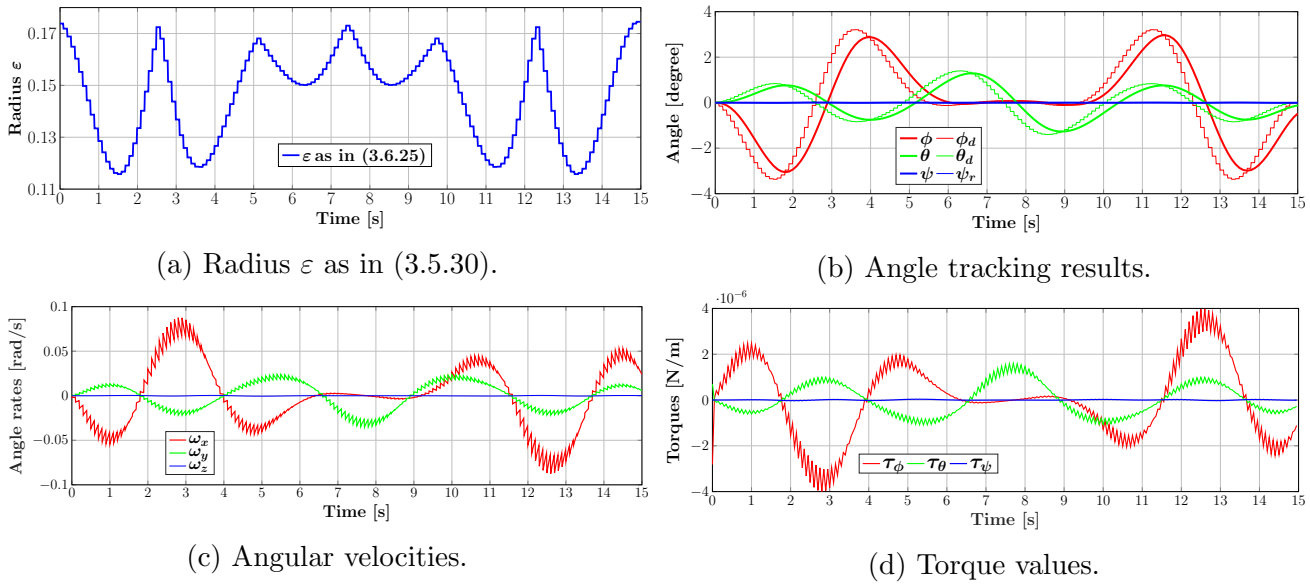
At each on-line simulation step, after taking the angle references $\eta_d = (\phi_d, \theta_d, \psi_r)^\top$ as shown by thin step lines in Figure 3.5.1b, the controller first chooses the radius ε as given in (3.5.30) (plotted in Figure 3.5.1a) in order to fully obtain the terminal constraint set $\mathcal{B}_K(\eta_d, \varepsilon)$ defined in (3.5.24). Then, the NMPC controller solves the optimization problem (3.1.4) and provides the input torques $(\tau_\phi, \tau_\theta, \tau_\psi)$ as shown in Figure 3.5.1d. The angle tracking results are shown

in Figure 3.5.1b in which the simulation angles plotted by thick lines (red for ϕ, green for θ and blue for ψ) closely track their piece-wise constant references given in thin lines with according colors. The resulting angular velocities are given in Figure 3.5.1c in which we observe a slight chattering phenomenon (can also be seen from the torques plotted in Figure 3.5.1d). This chattering problem is caused by the controller's effort to stabilize the angular velocities and the torques at their zero equilibrium values as in (3.5.9). By conducting multiple simulations, we find that this issue is aggravated by increasing the value of the weighting matrix R. Thus, we have reduced this phenomenon by decreasing the value of R and also the gain of the angular velocities within the cost function $\ell(\cdot)$ from (3.3.3) as shown in Table 3.5.1 (i.e., 0.1 for ω and 0.01 for τ in comparison with 1 for the angle η). Also, the issue can be further mitigated by adding a penalty on the input variation to the cost $\ell(\cdot)$ from (3.3.3) as proposed in [Badgwell and Qin, 2015]. It also should be clarified that all the states and inputs validates their constraints given in (3.5.4)–(3.5.5) due to the usage of the NMPC algorithm and the limits are not plotted in Figures 3.5.1b–3.5.1d to reduce the clutter since they are much larger than the states and inputs values.

Parameters	Value
Prediction horizon T_p as in (3.1.4)	0.05 seconds
NMPC sampling time δ as in (3.1.7)	0.01 seconds
Model discretizing step Δ as in (3.1.12)	0.01 seconds
C as in (3.5.14)	$[14.05\ 14\ 22]^\top 10^{-6}$
$\mathcal{M}$ as in (3.5.16)	$[30\ 44.6\ 44]^\top 10^{-6}$
$\varepsilon_{\max}$ as in (3.5.23)	8.3047
$\mathsf{L}_{\text{CTC, max}}$ as in (3.5.36)	10^{-7}
K as in (3.2.11) and (3.2.31)	$[-0.1\boldsymbol{I}_3\ -0.2\boldsymbol{I}_3]$
Q and R as in (3.3.3)	$\text{diag}\{1,1,1,0.1,0.1,0.1\}$ and $0.01\boldsymbol{I}_3$
R^* as in (3.5.37)	$\begin{bmatrix} 101\boldsymbol{I}_3 & 2\boldsymbol{I}_3 \\ 2\boldsymbol{I}_3 & 104\boldsymbol{I}_3 \end{bmatrix} 10^{-11}$
P as in (3.3.5)	$\begin{bmatrix} 3.525\boldsymbol{I}_3 & 5\boldsymbol{I}_3 \\ 5\boldsymbol{I}_3 & 25.25\boldsymbol{I}_3 \end{bmatrix}$

Table 3.5.1: Parameters prepared off-line of the NMPC controller of the rotation dynamics (3.5.6).

Beside the tracking results, the computing time per step is also important when discussing the implementing details for the NMPC controller. In Table 3.5.2, we gather all the information of the computing time through the whole simulation, i.e., 1500 steps. The average computing time is 54 milliseconds while the minimum value reaches 30.9 milliseconds. Note that, the maximum computing time of 93.7 milliseconds as noted in Table 3.5.2 happens at the first simulation time which usually requires more time than the rest for the solver to set up the algorithm. For the first 100 steps, the computing time is plotted in red line with circle marks in Figure 3.5.2 in comparison with the mean value given in solid blue line. We notice that the computing time is larger than the chosen sampling time $\delta = 0.1$ seconds as given in Table 3.5.1, hence, this setup (including both the formulation of the optimization problem (3.1.4) and



(a) Radius ε as in (3.5.30).

(b) Angle tracking results.

(c) Angular velocities.

(d) Torque values.

Figure 3.5.1: Simulation result on angle tracking of the proposed NMPC controller.

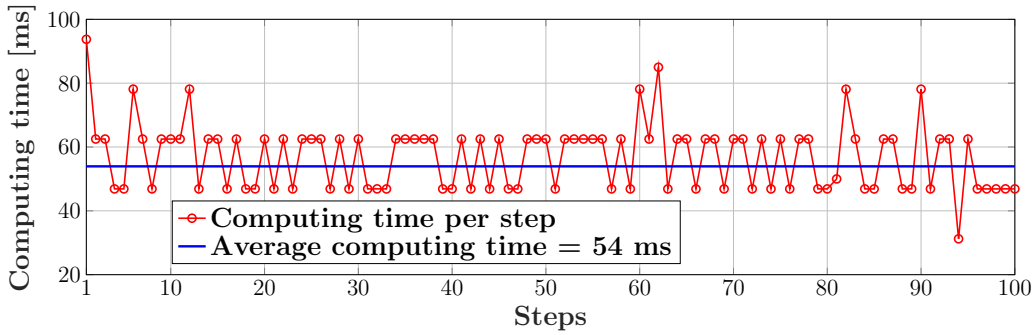


Figure 3.5.2: Computing time per step of the first 100 steps under simulation.

the employed hardware) is not ready for a real implementation. However, we are confident that various speeding-up approaches can be applied in order to mitigate the issue. E.g., [Zanelli et al., 2018] succeeds in embedding the optimization problem (3.1.4) into a low-power micro controller by re-formulating the problem into its approximated quadratic formulation and then, solving it by using a modified interior-point method. Also, different solving methods can be employed to speed-up the controller such as multiple shooting [Gros et al., 2012] or convex lifting [Gulan et al., 2017].

	Mean	Min	Max	Standard deviation
Value $[ms]$	54	30.9	93.7	10

Table 3.5.2: Information on the computing time of the NMPC controller designed for angle tracking of the rotation dynamics. Solver: IPOPT [Wächter and Biegler, 2006] in Python 3.0. Simulation steps: 1500.

3.6 Concluding remarks and open questions

Even though the name "Attitude control through using NMPC with guaranteed stability" indicates as premier contribution the design of an NMPC controller for attitude set-point angle tracking of the multicopter system, the content of this chapter is larger. It firstly details how to use the existing computed-torque controller of various robotics/mechanical systems (admitting Lagrangian dynamics), as the local controller to guarantee the closed-loop stability and feasibility of the corresponding NMPC design. In order to do so, the principles of designing NMPC with terminal stabilizing constraint given in [Chen and Allgöwer, 1998, Mayne et al., 2000] are applied:

- construct an input constraint admissible set of the CTC controller by using Taylor's approximation theory and also derive to its Lipschitz bound.
- propose the condition for choosing the control gain matrix such that the aforementioned input constraint admissible set becomes positive invariant, therefore, avoiding the conservativeness and the difficulties implied by tuning the parameters when using an arbitrary invariant set (c.f. Figure 3.3.1).
- validate the contributions through a simulation on a cart-pendulum system and through a comparison with quasi-infinite horizon NMPC [Chen and Allgöwer, 1998].
- apply the proposed design to stabilize the rotation dynamic of the multicopter system (i.e., the attitude control problem). The novelties lie in the several alternative designs for elements such as the Lipschitz constant and the radius of the terminal region. We reduce the dependence of the design on the desired set-points received from the high control level since they are not known in advance, and hence, cause much complexity for on-line computation when applying the originally proposed design without any modification.
- validate the NMPC attitude controller under simulation and obtain the average computing time per step of 54 milliseconds.

Some open questions which should be considered for improving and strengthening the contributions are given below:

- How to solve the proposed NMPC problem using a low-power microprocessor since the attitude controller is usually implemented on board? How are the computing time and the control performance affected by various compromises imposed by the processor architecture (e.g., solving an approximated optimization problem)?
- The inertial tensor J in (3.5.6) is difficult to obtain. What is the effect of any uncertainties in J for the controlled system? Can these be estimated and their effect bounded?

Chapter 4

Position control through NMPC designs with guaranteed stability

Recently, the multicopter platforms usually have their built-in controllers which control the rotors to track the four inputs consisting of the thrust level and the three Euler angles [Giernacki et al., 2017, Nguyen et al., 2018b, Nguyen et al., 2018a]. Therefore, controlling the multicopter vehicles requires only the maneuvering of their translation dynamics [Nguyen et al., 2018a]. This is already a challenging task as the dynamics are not only strongly nonlinear but also subject to many operating constraints [Nguyen et al., 2018b, Cao and Lynch, 2016, Lu et al., 2017, Giernacki et al., 2017]. For these reasons many researchers develop sophisticated feedback linearization controllers (e.g. the controller $u_{\text{FL}}(\cdot)$ from (2.4.1)) in order to satisfy the system constraints [Nguyen et al., 2018b, Cao and Lynch, 2016, Lu et al., 2017].
Model Predictive Control (MPC), as already introduced in the previous chapter, is a popular candidate for easily handling the constraints [Mayne et al., 2000, Badgwell and Qin, 2015]. With recently advances in microprocessor technology, it becomes possible to implement MPC algorithms for controlling the UAVs [Gros et al., 2012, Mueller and D'Andrea, 2013, Nguyen et al., 2017a, Zanelli et al., 2018]. However, most of the works in the literature disregard the stability problem in the MPC designs due to the extremely nonlinear dynamics of the drones, thus possibly leading to instability or even to infeasible solutions during execution. Therefore, in view of these shortcomings and also motivated by the expanding use of drones, we propose here three different NMPC designs for stabilizing the translational dynamics of the multicopter vehicles with the corresponding closed-loop stability properties guaranteed by:

- a standard ellipsoid terminal invariant set under the aforementioned FL controller (2.4.1): the set can be enlarged unlimitedly (but subject to the state constraints that might exist), covering an arbitrarily chosen compact set containing the initial states and hence, the NMPC design achieves the *semi-global* stability property (c.f. Definition 4.2.3);

- a relaxed invariant set under the FL controller: the invariant set is replaced by a pair of admissible and attractive sets such that the state trajectory is guaranteed to remain inside the later and to return at predefined periodic time instants into the later. Due to this relaxation, the terminal region can be obtained as a box. This reduces the complexity of the NMPC optimization problem while still guarantee the closed-loop stability;

- a "long-enough" prediction horizon: the NMPC design ensures the closed-loop stability by defining an appropriate prediction horizon length corresponding to a specific domain of attraction [Grüne, 2012, Kohler et al., 2018]. We prove that the existing design principles given in [Grüne, 2012, Kohler et al., 2018] can be satisfied by using the results obtained

from applying the aforementioned FL controller as a feasible initial guess of the NMPC optimization problem. Henceforth, we propose a thorough analysis on how to optimally tune the prediction horizon's length which turns out to be orders of magnitude less than when employing a linear controller (150 steps versus thousands).

The outline of the chapter is as follows. Firstly, in Section 4.1, we recapitulate the translation dynamics of the multicopter system and some preliminary results concerning the FL controller introduced in Section 2.4. Next, Section 4.2 details the two NMPC designs with different terminal constraint sets: i) an ellipsoid invariant set which guarantees semi-global stability and ii) a relaxed invariant set described through box-type linear constraints which also ensures closed-loop stability. Then, Section 4.3 shows that the FL controller further allows us to design an *unconstrained* NMPC scheme without using the terminal stabilizing constraints but with its stability guaranteed by an appropriately-chosen prediction horizon length. All of the proposed approaches are validated through extensive simulations and comparison with the classic quasi-infinite horizon NMPC design [Chen and Allgöwer, 1998]. Finally, the experimental validations of the two NMPC designs with terminal stabilizing constraints over the nano-drone Crazyflie 2.0 platform (c.f. Section 2.5.1) are presented in Section 4.5.

4.1 Preliminary results on the feedback linearization law

Let us briefly recapitulate the translation dynamics $\dot{\mathsf{p}} = \mathsf{f}_{\mathsf{p}}(\mathsf{p}, u, \psi)$ defined as in (2.3.1) by re-formulating it as follows:

$$\begin{bmatrix}\dot{\xi}\\ \dot{v}\end{bmatrix} = \underbrace{\begin{bmatrix}0_{3\times3} & \boldsymbol{I}_3\\ 0_{3\times3} & 0_{3\times3}\end{bmatrix}}_{A}\underbrace{\begin{bmatrix}\xi\\ v\end{bmatrix}}_{\mathsf{p}} + \begin{bmatrix}0_{3\times3}\\ h(T,\eta),\end{bmatrix} \tag{4.1.1}$$

in which, the input term $h(T,\eta)$ is defined in (2.3.2), the state vector $\mathsf{p} \in \mathbb{R}^6$ gathers the positions $\xi = [x\ y\ z]^\top$ and velocities $v = [v_x\ v_y\ v_z]^\top$ along the three axes, the input vector $u \triangleq [T\ \phi\ \theta]$ consists of the thrust and the roll, pitch angles while the yaw angle $\psi \in [-\pi, \pi]$ is an, assumed known, constant affecting the system (usually considered to be zero as in [Cowling et al., 2007, Lu et al., 2017] but here we consider an arbitrary value of ψ). Note that, we use the notation A in (4.1.1) similarly to the use of A in (3.2.9) since they share a similar construction with the only difference being their dimensions, i.e., A in (3.2.9) is a general case of the one in (4.1.1).
The translation system (4.3.6) is also subject to input constraints partly taken from those of the multicopter system (2.1.12)–(2.1.13):

$$u \in \mathcal{U}_{\mathrm{p}} = \big\{0 \le T_k \le T_{\max},\ |\phi_k| \le \epsilon_{\max},\ |\theta_k| \le \epsilon_{\max}\big\}, \tag{4.1.2}$$

with $T_{\max}$, $\epsilon_{\max}$ the maximum thrust and angle from (2.1.12)–(2.1.13). Note that, we only consider the input u of the system (4.3.6) without distinguishing the actual input and its desired control law as in (4.3.6) in order to simplify the notations. However, we impose the input constraint $u \in \mathcal{U}_{\mathrm{p}}$ as in (4.1.2) to the system (4.1.1) then enforce it by using the NMPC strategy, hence, still respecting the saturation effect as considered in (2.3.3).
Within this chapter, we address the problem of stabilizing the dynamics (4.1.1) at the reference position $\xi_r = [x_r\ y_r\ z_r]^\top$. Since we do not consider any state constraints for the dynamics, it is without loss of generality to transform the initial problem into once concerned with stabilizing the dynamics (4.1.1) around the zero equilibrium point:

$$\mathsf{p}_e = \mathbf{0},\ u_e = \begin{bmatrix}mg\ 0\ 0\end{bmatrix}^\top, \tag{4.1.3}$$

with m the system mass and g the gravity as in (2.1.9).
The system (4.1.1) admits the feedback linearization (FL) law $u_{\text{FL}}(\mu_\xi, \psi)$ as defined in (2.4.1) which is re-written hereinafter for completeness:

$$T_{\text{FL}}(\mu_\xi) = m\sqrt{\mu_x^2 + \mu_y^2 + (\mu_z + g)^2}, \tag{4.1.4a}$$

$$\phi_{\text{FL}}(\mu_\xi, \psi) = \arcsin\left(\frac{\mu_x \sin\psi - \mu_y \cos\psi}{\sqrt{\mu_x^2 + \mu_y^2 + (\mu_z + g)^2}}\right), \tag{4.1.4b}$$

$$\theta_{\text{FL}}(\mu_\xi, \psi) = \arctan\left(\frac{\mu_x \cos\psi + \mu_y \sin\psi}{\mu_z + g}\right), \tag{4.1.4c}$$

with $u_{\text{FL}}(\mu_\xi, \psi) \triangleq [T_{\text{FL}}(\mu_\xi)\ \phi_{\text{FL}}(\mu_\xi, \psi)\ \theta_{\text{FL}}(\mu_\xi, \psi)]^\top$ the FL-based control, $\mu_\xi \triangleq [\mu_x\ \mu_y\ \mu_z]^\top$ the virtual input vector and ψ the yaw angle as in (4.1.1). Below is also a recap of several important remarks on the FL-based control (4.1.1) which have been discussed in Section 2.4:

- By Proposition 2.4.1, under the FL-based control $u_{\text{FL}}(\mu_\xi, \psi)$ from (4.1.4), if $\mu_z \geq -g$, then, the nonlinear dynamics (4.3.6) are linearized into the three decoupled double integrators (2.4.2). They are gathered into a matrix form as follows:

$$\dot{\mathsf{p}} = A\mathsf{p} + B\mu_\xi, \tag{4.1.5}$$

  with[1] A as in (4.1.1) and $B = [0_{3\times 3}\ \boldsymbol{I}_3]^\top$.

- By Proposition 2.4.2, the input constraint admissible set w.r.t. the FL-based control $u_{\text{FL}}(\mu_\xi, \psi)$ given in (4.1.4) is as follows:

$$\mathcal{X}_{\text{FL}} = \Big\{|\mu_x| \leq U_x,\ |\mu_y| \leq U_y,\ |\mu_z| \leq U_z\Big\}, \tag{4.1.6}$$

  with (U_x, U_y, U_z) the positive constants defined as in (2.4.5)–(2.4.7). Furthermore, since we do not consider state constraint for the system (4.1.1), the set $\mathcal{X}_{\text{FL}}$ also plays the role of the admissible set.

Next, we show that within the admissible set $\mathcal{X}_{\text{FL}}$ defined in (4.1.6), the FL law $u_{\text{FL}}(\mu_\xi, \psi)$ as in (4.1.4) admits a Lipschitz bound (i.e., similar to the case of the computed-torque controller given in (3.2.27)). This property will be employed to prove the stability of the NMPC designs introduced in the following sections.

Proposition 4.1.1. *For all $\mu_\xi \triangleq [\mu_x\ \mu_y\ \mu_z]^\top \in \mathcal{X}_{FL}$ as in (4.1.6), each element of the FL law $u_{FL}(\mu_\xi, \psi)$ defined in (4.1.4) is bounded as follows:*

$$|T_{FL}(\mu_\xi) - mg| \leq m\|\mu_\xi\|, \tag{4.1.7}$$

$$\big\langle|\phi_{FL}(\mu_\xi, \psi)|,\ |\theta_{FL}(\mu_\xi, \psi)|\big\rangle \leq \frac{\sqrt{\mu_x^2 + \mu_y^2}}{g - U_z},\ \forall\psi \in \mathbb{R}, \tag{4.1.8}$$

with m, the system mass, g, the gravity and $0 < U_z < g$ the positive constant defined as in (2.4.5). As a result, the value of $\|u_{FL}(\mu_\xi, \psi) - u_e\|$ with u_e given in (4.1.3) admits the following Lipschitz bound:

$$\|u_{FL}(\mu_\xi, \psi) - u_e\|^2 \leq \mathsf{L}\|\mu_\xi\|^2, \tag{4.1.9}$$

[1]We keep the same notations: A and B in (4.1.5) of the dynamics (4.1.1) as similar to A and B in (3.2.9) of the general system (3.1.1) since they provide the same meaning.

with the Lipschitz constant L *given by:*

$$\mathsf{L} = m + \frac{2}{g - U_z}. \tag{4.1.10}$$

Proof. Let us start by proving (4.1.7). Considering $T_{\text{FL}}(\mu_\xi)$ as in (4.1.4a), we have that:

$$(T_{\text{FL}}(\mu_\xi) - mg)^2 = m^2(\mu_x^2 + \mu_y^2 + \mu_z^2) + 2m^2 g\left(\mu_z + g - \sqrt{\mu_x^2 + \mu_y^2 + (\mu_z + g)^2}\right). \tag{4.1.11}$$

Next, if $u_z + g > 0$, we have that:

$$\sqrt{\mu_x^2 + \mu_y^2 + (\mu_z + g)^2} \geq |\mu_z + g| = \mu_z + g, \tag{4.1.12}$$

otherwise, if $\mu_z + g \leq 0$, it is straightforward to obtain:

$$\mu_z + g - \sqrt{\mu_x^2 + \mu_y^2 + (\mu_z + g)^2} \leq 0. \tag{4.1.13}$$

Thus, introducing (4.1.12) and (4.1.13) to (4.1.11) leads to:

$$(T_{\text{FL}}(\mu_\xi) - mg)^2 \leq m^2(\mu_x^2 + \mu_y^2 + \mu_z^2), \tag{4.1.14}$$

which is equivalent to (4.1.7).
Next, to prove (4.1.8), using the differential flatness properties given in Proposition 2.2.4, we have that:

$$\left\langle |\phi_{\text{FL}}(\mu_\xi, \psi)|,\ |\theta_{\text{FL}}(\mu_\xi, \psi)| \right\rangle \leq \underbrace{\arctan\left(\sqrt{\frac{\mu_x^2 + \mu_y^2}{(\mu_z + g)^2}}\right)}_{\Upsilon_\epsilon(\mu_\xi)}, \forall \psi \in \mathbb{R}, \tag{4.1.15}$$

in which, $\Upsilon_\epsilon(\cdot)$ is defined in (2.2.15) and its tangent is given in (2.2.17). Next, we have that $|\mu_z| < U_z$ due to $\mu_\xi \in \mathcal{X}_{\text{FL}}$ as in (4.1.6) and $U_z < g$ (2.4.5) which ultimately leads to $|\mu_z + g| \geq g - U_z$. Then, introducing this result and $\arctan(x) \leq x,\ \forall x > 0$ to (4.1.15) results in (4.1.8). The norm as in (4.1.9) is straightforward to obtain by using (4.1.7)–(4.1.8), hence, completing the proof. □

Remark 4.1.2. A preliminary version of Proposition 4.1.1 has been published in [Nguyen et al., 2019b] where the inequality (4.1.7) is given in a more conservative form: $|T_{\text{FL}}(\mu_\xi) - mg| \leq 3m\|\mu_\xi\|$. The new result as in (4.1.7) improves significantly the corresponding NMPC designs in comparison with the one in [Nguyen et al., 2019b]. □

In the next section, we will employ the presented results for designing two NMPC schemes with stability guaranteed by two different terminal constraint sets. The sets posses different properties (invariant and δ-invariant) but both are constructed from the FL law (4.1.4).

4.2 NMPC designs with terminal constraint for position control

In this section, we detail the NMPC design with terminal stabilizing constraints (c.f. the NMPC scheme in (3.1.4)) for the system (4.1.1). We will make use of two different constructions of the terminal constraint set: i) a standard ellipsoid invariant set and ii) a relaxed invariant set, called δ-invariant set with δ, the sampling time. The latter construction allows the state to escape

from the set but enforces it to return to the set periodically, hence, being a generalization of the standard invariant set notion [Doban and Lazar, 2018, Lazar and Spinu, 2015]. Both the invariant and δ-invariant sets are designed under the FL law (4.1.4) in order to benefit from the simpler linear system (4.1.5) instead of dealing with the original nonlinear dynamics (4.1.1).
In the following, we particularize the general NMPC optimization problem given in (3.1.4) for the system (4.1.1) for completeness sake. For stabilizing the system (4.1.1) around the zero equilibrium point (4.1.3), the NMPC controller at time t is designed in the following form:

$$\bar{u}_t^*(\cdot) = \arg\min_{\bar{u}_t(\cdot)} \left\{ \int_t^{t+T_p} \ell(\bar{\mathsf{p}}_t(s), \bar{u}_t(s))ds + F(\bar{\mathsf{p}}_t(t+T_p)) \right\}, \tag{4.2.1}$$

subject to:

$$\dot{\bar{\mathsf{p}}}_t = \mathsf{f}_{\mathsf{p}}(\bar{\mathsf{p}}_t, \bar{u}_t), \text{ the dynamics as in (4.1.1)}, \tag{4.2.2a}$$
$$\bar{u}_t(s) \in \mathcal{U}_{\mathrm{p}}, \ s \in [t, t+T_p], \text{ the input constraints as in (4.1.2)}, \tag{4.2.2b}$$
$$\bar{\mathsf{p}}_t(t) = \mathsf{p}(t), \text{ the initial condition}, \tag{4.2.2c}$$
$$\bar{\mathsf{p}}_t(t+T_p) \in \mathcal{X}_f, \text{ the terminal stabilizing constraint}, \tag{4.2.2d}$$

in which, $\bar{\mathsf{p}}_t(s)$ and $\bar{u}_t(s)$ stand for the predicted state and input at time s ($t \leq s \leq t+T_p$) while $\bar{u}_t(\cdot)$ as in (4.2.1) gives the whole predicted input trajectory along the prediction horizon interval $[t, t+T_p]$, all corresponding to the optimization problem (4.2.1) at time t. The stage cost $\ell(\cdot)$ and terminal cost $F(\cdot)$ will be defined in their standard quadratic form hereinafter while the terminal constraint set $\mathcal{X}_f \in \mathbb{R}^6$ used in (4.2.2d) will be taken either as the standard invariant set or as the δ-invariant set in the two next sections, respectively. The NMPC input is taken from the optimal input profile of (4.2.1) within the first sampling time δ, similar to (3.1.7):

$$u_{\mathrm{MPC}}(s) = \bar{u}_t^*(s), \ \forall s \in [t, t+\delta], \tag{4.2.3}$$

with $\bar{u}_t^*(\cdot)$ the optimal input trajectory resulted from (4.2.1).
Even though we will employ two different designs for the terminal constraint set $\mathcal{X}_f$ from (4.2.2d), the two NMPC designs share the same stage and terminal costs defined as follows (useful from the viewpoint of the subsequent comparisons: only the terminal set construction differs and hence its influence on the performance can be easily isolated):

$$\ell(\mathsf{p}, u) := \|\mathsf{p}\|_Q^2 + \|u - u_e\|_R^2, \tag{4.2.4}$$
$$F(\mathsf{p}) := \|\mathsf{p}\|_P^2, \tag{4.2.5}$$

in which, $\|\mathsf{p}\|_Q^2 = \mathsf{p}^\top Q \mathsf{p}$ with $Q \in \mathbb{R}^{6\times 6}$ a symmetric positive definite matrix; similar reasoning is applied for $\|u-u_e\|_R^2$ and $\|\mathsf{p}\|_P^2$ with the two matrices $R \in \mathbb{R}^3$ (symmetric positive semi-definite) and $P \in \mathbb{R}^{6\times 6}$ (symmetric positive definite). The matrices Q, R are defined by the user while the terminal weighting matrix P in (4.2.5) is obtained by solving the following Lyapunov equation:

$$A_K^\top P + P A_K + M = \mathbf{0}, \tag{4.2.6}$$

in which, the elements are given as follows:

- the matrix $A_K \in \mathbb{R}^{6\times 6}$ describes the closed-loop behavior of the linear system (4.1.5):

$$\dot{\mathsf{p}} = \underbrace{(A + BK)}_{A_K} \mathsf{p}, \tag{4.2.7}$$

with the matrix A, B as in (4.1.5) and the gain matrix $K \in \mathbb{R}^{3\times 6}$ given by:

$$K = \left[\text{diag}\{K_{p_x}, K_{p_y}, K_{p_z}\} \ \text{diag}\{K_{d_x}, K_{d_y}, K_{d_z}\}\right], \tag{4.2.8}$$

with all the control gains required to be negative to ensure the stability of (4.2.7). The linear system (4.2.7) is resulted from using the feedback linearization controller $u_{\text{FL}}(K\mathsf{p}, \psi)$ from (4.1.4) as the local controller (denoted by $\boldsymbol{u}_{\text{loc}}(\boldsymbol{x})$ for the general case as in (3.1.9)) within the NMPC design:

$$u_{\text{loc}}(\mathsf{p}) := u_{\text{FL}}(K\mathsf{p}, \psi). \tag{4.2.9}$$

with $u_{\text{loc}}(\mathsf{p})$ the local controller particularized for the system (4.1.1), K given in (4.2.8) and ψ the yaw angle received as the feedback of the system.

- The symmetric matrix $M \in \mathbb{R}^{6\times 6}$ is chosen such that:

$$M \succeq Q + \mathsf{L}\max(\text{eig}(R))K^\top K, \tag{4.2.10}$$

with the Lipschitz bound L from (4.1.10), $\max(\text{eig}(R))$ the maximum eigenvalues of the weighting matrix R from (4.2.1) and the control gain matrix K as in (4.2.8).

The aforementioned calculation process for P as in (4.2.6) may raise questions on why the two NMPC designs with two different constructions of the terminal constraint sets can be stabilized by using the same method for finding the terminal weighting matrix P. The explicit answers will be given in the corresponding sections while in the following, we provide a hint which shows that using P from (4.2.6) can guarantee condition **C4** (3.1.10) of Lemma 3.1.1 in the whole admissible set $\mathcal{X}_{\text{FL}}$ as in (4.1.6), thus, also ensuring (3.1.10) for any subsets of $\mathcal{X}_{\text{FL}}$ (the terminal constraints are always required to be an admissible set as in (3.1.8)–(3.1.9)).

Proposition 4.2.1. *The requirement* (3.1.10) *(as required by Lemma 3.1.1) is ensured for all* $\mathsf{p} \in \mathcal{X}_{FL}$ *in* (4.1.6) *by employing the terminal quadratic cost* $F(\mathsf{p}) := \|\mathsf{p}\|_P^2$ *as in* (4.2.5) *with* $P \in \mathbb{R}^{6\times 6}$ *satisfying* (4.2.6) *and the local FL controller* $u_{loc}(\mathsf{p}) := u_{FL}(K\mathsf{p}, \psi)$ *as in* (4.2.9).

Proof. The proof is constructed similar to the one of Lemma 3.3.1 but with a slight difference. I.e., since the set $\mathcal{X}_{\text{FL}}$ is admissible, we can apply $\dot{\mathsf{p}} = \mathsf{f}_{\mathsf{p}}(\mathsf{p}, u_{\text{FL}}(K\mathsf{p}, \psi), \psi) = A_K\mathsf{p}$ as given in (4.2.7) for all $\mathsf{p} \in \mathcal{X}_{\text{FL}}$. Then, with the current notations, condition **C4** (3.1.10) is re-formulated as follows:

$$\underbrace{\mathsf{p}^\top A_K^\top P\, \mathsf{p} + \mathsf{p}^\top P A_K \mathsf{p} + \|\mathsf{p}\|_Q^2 + \|u_{\text{FL}}(K\mathsf{p}, \psi) - u_e\|_R^2}_{\text{LHS of (4.2.11)}} \leq 0, \tag{4.2.11}$$

with A_K as in (4.2.7), P the terminal weighting matrix from (4.2.6), Q, R the weighting matrices in (4.2.1) and K the control gain matrix in (4.2.8). By introducing the bound of $\|u_{\text{FL}}(K\mathsf{p}, \psi) - u_e\|$ as in (4.1.9) to the left-hand side of (4.2.11), we have that:

$$\begin{aligned} \text{LHS of (4.2.11)} &\leq \mathsf{p}^\top \left(A_K^\top P + P A_K + Q + \max(\text{eig}(R))\mathsf{L}K^\top K\right)\mathsf{p} \\ &\leq \mathsf{p}^\top \left(A_K^\top P + P A_K + M\right)\mathsf{p} = 0, \end{aligned} \tag{4.2.12}$$

with the constant L as in (4.1.10), M as in (4.2.10) and P satisfying (4.2.6), hence, completing the proof. □

Choosing the symmetric matrix M as in (4.2.10) causes difficulties since the matrix lies in $\mathbb{R}^{6\times 6}$ space, hence, having 21 tuning variables. Therefore, we will prove that it is possible to employ a diagonal matrix M with only 6 tuning variables but still satisfying requirement (4.2.10).

Proposition 4.2.2. *Let us parameterize the positive definite matrix $M \in \mathbb{R}^{6\times 6}$ employed in* (4.2.10) *as follows (simultaneously, restricting it to a diagonal form):*

$$M = \text{diag}\{m_x, m_y, m_z, m_{v_x}, m_{v_y}, m_{v_z}\}, \tag{4.2.13}$$

then, there always exist some values of $(m_x, m_y, m_z, m_{v_x}, m_{v_y}, m_{v_z}) > 0$ such that $M \succeq Q + \mathsf{L}\max(\text{eig}(R))K^\top K$, as required in (4.2.10)*, holds.*

Proof. At first, let us define the matrix $Q^* = Q + \mathsf{L}\max(\text{eig}(R))K^\top K$ with the elements defined in (4.2.10). Then, Q^* has a symmetrical structure defined as follows:

$$Q^* = \begin{bmatrix} \text{diag}\{Q^*_{1_x}, Q^*_{1_y}, Q^*_{1_z}\} & \text{diag}\{Q^*_{3_x}, Q^*_{3_y}, Q^*_{3_z}\} \\ \text{diag}\{Q^*_{3_x}, Q^*_{3_y}, Q^*_{3_z}\} & \text{diag}\{Q^*_{2_x}, Q^*_{2_y}, Q^*_{2_z}\} \end{bmatrix}, \tag{4.2.14}$$

which is due to the symmetry of Q as in (4.2.4) and of the term $K^\top K$ with K defined in (4.2.8). Next, in order to examine the positive definiteness of the matrix $M - Q^*$, we introduce an arbitrary vector $\mathsf{p} = [x\ y\ z\ v_x\ v_y\ v_z]^\top \in \mathbb{R}^6$ to $\mathsf{p}^\top(M - Q^*)\mathsf{p}$ with M, Q^* as in (4.2.13)–(4.2.14):

$$\mathsf{p}^\top(M - Q^*)\mathsf{p} = \sum_{q\in\{x,y,z\}} \left((m_q - Q^*_{1_q})q^2 + (m_{v_q} - Q^*_{2_q})v_q^2 - 2Q^*_{3_q} q v_q\right). \tag{4.2.15}$$

Therefore, from (4.2.15), we can obtain $\mathsf{p}^\top(M - Q^*)\mathsf{p} \geq 0,\ \forall \mathsf{p} \in \mathbb{R}^6$ by constraining the main coefficients $(m_q - Q^*_{1_q}, m_{v_q} - Q^*_{2_q})$ and the discriminant $4{Q^*_{3_q}}^2 - 4(m_q - Q^*_{1_q})(m_{v_q} - Q^*_{2_q})$ to be positive as follows:

$$\begin{cases} m_q - Q^*_{1_q} > 0, \\ m_{v_q} - Q^*_{2_q} > 0, \\ (m_q - Q^*_{1_q})(m_{v_q} - Q^*_{2_q}) \geq {Q^*_{3_q}}^2, \end{cases} \quad \forall q \in \{x, y, z\}. \tag{4.2.16}$$

One possible and simple solution for the inequalities (4.2.16) is to ensure $m_q - Q^*_{1_q} > Q^*_{3_q}$ and $m_{v_q} - Q^*_{2_q} > Q^*_{3_q}$, hence, leading to the following conditions on choosing M as in (4.2.13):

$$\begin{cases} m_q > Q^*_{1_q} + Q^*_{3_q}, \\ m_{v_q} > Q^*_{2_q} + Q^*_{3_q}, \end{cases} \quad \forall q \in \{x, y, z\}. \tag{4.2.17}$$

This also completes the proof. □

4.2.1 NMPC design with semi-global stability guarantee

In this section, we address an NMPC design with semi-global stability guarantees for the dynamics (4.1.1). The design makes use of a terminal invariant set and follows the NMPC design principles given in Lemma 3.1.1, hence, it is able to guarantee the recursive feasibility and asymptotic stability of the closed-loop controlled system (c.f. Figure 3.1.1). However, the recursive feasibility implies the feasibility of the controller only after the first successful iteration. The reason is due to the terminal constraint as given in (3.1.6d) which enforces the final predicted state to enter the terminal invariant set by the end of the prediction horizon. This obviously prevents starting from an arbitrary initial state far from the terminal region, hence, causing a limited domain of attraction.

One trivial solution is to increase the prediction horizon length, but, this obviously results in a heavier computational burden. Thus, in order to continue increasing the region of attraction

when the prediction horizon is already large, we can increase the size of the terminal constraint set instead. However, due to the complexity of the existing NMPC designs with guaranteed stability, tuning the set is not always easy. One illustrative example is the *quasi-infinite horizon* NMPC design [Chen and Allgöwer, 1998] which has been summarized in Section 3.4.1. Within the design, the size of the terminal constraint set has to be scaled to ensure not only the admissible properties (3.4.7)–(3.4.8) but also the stability (3.4.9), hence, any modifications on the tuning parameters results in various re-checks. Furthermore, the design is not initially-state-oriented, in the sense that, we cannot directly obtain the parameters which allow us to stabilize the system from a predefined initial state but always have to check for the first iteration. This is also a common limitation of various existing NMPC designs with guaranteed stability [Cannon et al., 2003, Magni and Scattolini, 2004, Simon et al., 2013].
Noticing these shortcomings, we propose in this section an NMPC design for the system (4.1.1) whose terminal constraint set is easy to tune, and furthermore, can be enlarged accordingly to the initial condition and unlimitedly. Hence, the NMPC controller achieves the semi-global asymptotic stability (c.f. Definition 4.2.3).

Definition 4.2.3 (Semi-global stabilization [Braslavsky and Middleton, 1996])**.** *A system is semi-global stabilizable to the considered equilibrium point $\boldsymbol{x}_e$ by means of a class $\mathcal{F}$ of feedback control laws, if, for any a priori determined compact set $\mathcal{X}_0$ of initial conditions, there exists a control law in $\mathcal{F}$ that makes $\boldsymbol{x}_e$ asymptotically stable with a domain of attraction that contains $\mathcal{X}_0$.*

Regarding Definition 4.2.3, we adopt it for designing our NMPC controller with a relaxation: we undervalue the region of attraction by the terminal constraint set. More precisely, since the region of attraction increases accordingly to the prediction horizon length and it is impossible to determine how large the prediction horizon should be for a specific computing system, we, ultimately, present how to tune the control parameters such that the corresponding terminal invariant set contains the predefined initial state. However, bear in mind that the set can be easily modified according to the user's desire (e.g, slightly reducing the size of the set to improve the convergence speed).

4.2.1.1 NMPC design with terminal invariant set for position control

The NMPC design makes use of the local FL controller $u_{\text{FL}}(K\mathsf{p}, \psi)$ as defined in (4.1.4) and (4.2.8). In the following, we provide the most important element of the design - the construction of the positive invariant set under the FL controller $u_{\text{FL}}(K\mathsf{p}, \psi)$.

Proposition 4.2.4 ([Nguyen et al., 2019b])**.** *Let us consider the set $\mathcal{S}(P, r)$ defined as follows:*

$$\mathcal{S}(P, r) = \{\mathsf{p} \in \mathbb{R}^6 |\ \mathsf{p}^\top P\, \mathsf{p} \leq \min(\text{eig}(P)) r^2\}, \tag{4.2.18}$$

with $P \in \mathbb{R}^{6\times 6}$ obtained from (4.2.6) by using a symmetric positive definite matrix $M \in \mathbb{R}^{6\times 6}$ satisfying (4.2.10) (e.g. as in (4.2.13)). The parameter $r \in \mathbb{R}_+$ is chosen as follows:

$$r = \min\{r_x, r_y, r_z\}, \tag{4.2.19}$$

in which, r_q with $q \in \{x, y, z\}$ is defined as:

$$r_q = \frac{U_q}{\sqrt{K_{p_q}^2 + K_{d_q}^2}}, \tag{4.2.20}$$

with (U_x, U_y, U_z) *the positive constants as in* (2.4.5)–(2.4.7) *and* (K_{p_q}, K_{d_q}) *the control gains as in* (4.2.8). *Then, the set* $\mathcal{S}(P,r)$ *defined in* (4.2.18) *is constraint admissible and positive invariant under the FL controller* $u_{FL}(K\mathsf{p}, \psi)$ *from* (4.2.9).

Proof. Using a symmetric positive definite M ensures the symmetry and positive definiteness of the matrix P obtained as the solution of the Lyapunov equation (4.2.6). Therefore, all the eigenvalues of P are positive scalars which further leads to:

$$\min(\text{eig}(P))\|\mathsf{p}\|^2 \leq \|\mathsf{p}\|_P^2 \leq \max(\text{eig}(P))\|\mathsf{p}\|^2. \tag{4.2.21}$$

Then, for all $\mathsf{p} \in \mathcal{S}(P,r)$ as defined in (4.2.18), we have that:

$$\min(\text{eig}(P))\|\mathsf{p}\|^2 \leq \|\mathsf{p}\|_P^2 \leq \min(\text{eig}(P))r^2, \tag{4.2.22}$$

with r as in (4.2.19). This leads to $\|\mathsf{p}\|^2 \leq r^2$ which further implies that:

$$q^2 + v_q^2 \leq r_q^2, \ \forall q \in \{x, y, z\}, \tag{4.2.23}$$

with r_q as in (4.2.20).
Next, the virtual input design $\mu = K\mathsf{p}$ employed in the FL controller $u_{\text{FL}}(K\mathsf{p}, \psi)$ as in (4.2.9) with K from (4.2.8) has its explicit formulation along each axis provided by:

$$\mu_q = K_{p_q} q + K_{d_q} v_q, \ \forall q \in \{x, y, z\}, \tag{4.2.24}$$

with $\mu = [\mu_x \ \mu_y \ \mu_z]^\top$ as defined in (4.1.6). Applying the Cauchy-Schwarz inequality to the virtual input from (4.2.24) leads to:

$$|\mu_q| \leq \sqrt{\left(K_{p_q}^2 + K_{d_q}^2\right)\left(q^2 + v_q^2\right)}. \tag{4.2.25}$$

Introducing (4.2.23) to (4.2.25) provides:

$$|\mu_q| \leq \sqrt{\left(K_{p_q}^2 + K_{d_q}^2\right)\frac{U_q^2}{K_{p_q}^2 + K_{d_q}^2}} = U_q, \ \forall q \in \{x, y, z\}, \tag{4.2.26}$$

for all $\mathsf{p} \in \mathcal{S}(P,r)$ defined in (4.2.18), which further provides:

$$\mathcal{S}(P,r) \subset \mathcal{X}_{\text{FL}}, \tag{4.2.27}$$

with $\mathcal{X}_{\text{FL}}$ the input constraint admissible set from (4.1.6). Therefore, the set $\mathcal{S}(P,r)$ is input constraint admissible w.r.t. the FL controller $u_{\text{FL}}(K\mathsf{p}, \psi)$ given in (4.2.9). This further implies that within $\mathcal{S}(P,r)$, the controlled system is equivalent to $\dot{\mathsf{p}} = A_K\mathsf{p}$ as in (4.2.7). Then, the invariant property of $\mathcal{S}(P,r)$ as in (4.2.18) is proved by considering the following Lyapunov function $V = \mathsf{p}^\top P\,\mathsf{p}$ whose derivative is given by:

$$\dot{V} = \dot{\mathsf{p}}^\top P\,\mathsf{p} + \mathsf{p}^\top P\,\dot{\mathsf{p}} = \mathsf{p}^\top\left(A_K^\top P + P A_K\right)\mathsf{p} = -\mathsf{p}^\top M\mathsf{p} < 0, \tag{4.2.28}$$

with P, M the symmetric positive definite matrices satisfying the Lyapunov equation (4.2.6). Since $\dot{V} < 0$ as in (4.2.28), the set $\mathcal{S}(P,r)$ as defined in (4.2.18) is positive invariant, hence, completing the proof. □

Next, by using the aforementioned invariant set, we will complete the NMPC design given in (4.2.1)–(4.2.5) (i.e, still lacking of the terminal constraint set $\mathcal{X}_f$ as in (4.2.2d)). The stability proof is also given hereinafter.

Proposition 4.2.5 ([Nguyen et al., 2019b]). *Let us consider the NMPC design given in* (4.2.1)–(4.2.5) *for stabilizing the system* (4.1.1). *By using the terminal invariant set* $\mathcal{S}(P,r)$ *as defined in* (4.2.18)*:*

$$\mathcal{X}_f := \mathcal{S}(P,r), \tag{4.2.29}$$

with $P \in \mathbb{R}^{6\times 6}$ *the terminal weighting matrix obtained from* (4.2.6) *and* r *as in* (4.2.19)*, the closed-loop controlled system achieves (nominal) asymptotic stability.*

Proof. The stability is proven by the satisfaction of the four NMPC design principles given in Lemma 3.1.1. The first three conditions **C1**–**C3** as in (3.1.8)–(3.1.9) are obviously validated by using $\mathcal{X}_f := \mathcal{S}(P,r)$ with the set $\mathcal{S}(P,r)$ being both constraint admissible and positive invariant as detailed in Proposition 4.2.4. The final condition **C4** as in (3.1.10) is also satisfied within the set $\mathcal{S}(P,r) \subset \mathcal{X}_{\text{FL}}$ (as proven in (4.2.27)) by applying Proposition 4.2.1. □

Remark 4.2.6. In [Nguyen et al., 2019b], we present Propositions 4.2.4 and 4.2.5 in discrete domain. The only difference is that the matrix $P \in \mathbb{R}^{6\times 6}$ employed to construct the invariant set $\mathcal{S}(P,r)$ as in (4.2.18) is obtained from the discrete version of the Lyapunov equation (4.2.6) while the rest is without any modifications.
Furthermore, the NMPC controller given in Proposition 4.2.5 is fully capable of handling standard state constraints of the system (4.1.1), e.g.:

$$\mathsf{p} \in \mathcal{X}_{\text{p}}, \tag{4.2.30}$$

with $\mathcal{X}_{\text{p}}$, the non-empty set containing the equilibrium point p_e as in (4.1.3). Then, the radius r as in (4.2.19) has to be chosen such that:

$$\begin{cases} r \leq \min\{r_x, r_y, r_z\}, \\ \mathcal{S}(P,r) \subset \mathcal{X}_{\text{p}}, \end{cases} \tag{4.2.31}$$

with (r_x, r_y, r_z) as defined in (4.2.20) and $\mathcal{X}_{\text{p}}$ the state constraint set from (4.2.30). However, considering the state constraint as in (4.2.30) also impends proving the semi-global stability property of the design given in Proposition 4.2.5 (as will be detailed hereinafter). The reason is that the shape of $\mathcal{X}_{\text{p}}$ from (4.2.30) is arbitrary (subject to the user demand, physical constraints, etc.) while the terminal constraint set $\mathcal{S}(P,r)$ as in (4.2.18) is an ellipsoid in $\mathbb{R}^6$. Thus, it is impossible to tune $\mathcal{S}(P,r)$ such that it covers an arbitrarily predefined point in $\mathcal{X}_{\text{p}}$ without exceeding $\mathcal{X}_{\text{p}}$ due to some restricted areas at corners or edges of the undetermined set $\mathcal{X}_{\text{p}}$ from (4.2.30). In any case, we emphasize again that reducing the radius r in (4.2.19) is always feasible and should be considered according to specific applications. □

4.2.1.2 Unlimited expandability of terminal invariant set

This section proves the semi-global stability property of the proposed NMPC design (4.2.1)–(4.2.3) using the terminal invariant set $\mathcal{S}(P,r)$ as in (4.2.18). We firstly show that the set $\mathcal{S}(P,r)$ can be enlarged covering an arbitrarily chosen initial state $\mathsf{p}_0 \in \mathbb{R}^6$, then, the result is extended for a compact set $\mathcal{X}_0$ containing all the feasible initial states.

Proposition 4.2.7. *For any arbitrarily predefined state* $\mathsf{p}_0 \in \mathbb{R}^6$*, there always exist some values of the control gain matrix* K *defined as in* (4.2.8)*, and of the matrix* M *as in* (4.2.13) *such that, the corresponding invariant set* $\mathcal{S}(P,r)$ *as in* (4.2.18) *contains* p_0*. I.e.:*

$$\mathsf{p}_0 \in \mathcal{S}(P,r) = \{\mathsf{p} \in \mathbb{R}^6 |\ \mathsf{p}^\top P\, \mathsf{p} \leq \min(\text{eig}(P))r^2\}, \tag{4.2.32}$$

with P *obtained from* (4.2.6) *and* r *as in* (4.2.19)*.*

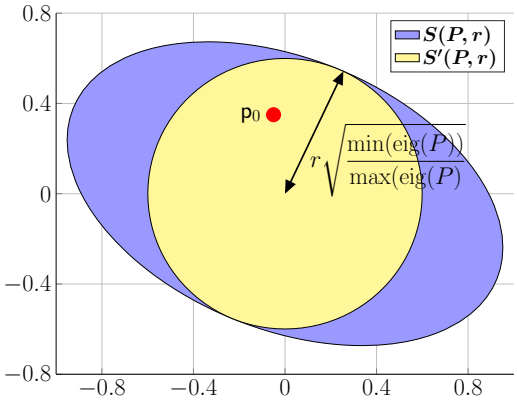


Figure 4.2.1: Illustration in 2D space of the two sets $\mathcal{S}(P,r)$ and $\mathcal{S}'(P,r)$ as in (4.2.32)–(4.2.33).

In the rest of the section, we concentrate on providing the proof of the aforementioned proposition. Since it will require many materials (some of them are given as new propositions and corollaries), we do not limit the content in a *proof* paragraph and for those who want to check fast the results, the solutions are given in (4.2.60)–(4.2.61). The insight of the proof is to consider the set $\mathcal{S}'(P,r)$ defined as follows:

$$\mathcal{S}'(P,r) = \left\{ \mathsf{p} \in \mathbb{R}^6 |\ \|\mathsf{p}\|^2 \leq \frac{\min(\text{eig}(P))}{\max(\text{eig}(P))} r^2 \right\}, \tag{4.2.33}$$

with P, r the same parameters as in (4.2.32) which always admits:

$$\mathcal{S}'(P,r) \subset \mathcal{S}(P,r), \tag{4.2.34}$$

with $\mathcal{S}(P,r)$ as in (4.2.32). This is due to the fact that $\mathsf{p}^\top P\, \mathsf{p} \leq \max(\text{eig}(P))\|\mathsf{p}\|^2 \leq \min(\text{eig}(P))r^2$ for all $\mathsf{p} \in \mathcal{S}'(P,r)$ defined in (4.2.33). We provide an illustration for the relation (4.2.34) in Figure 4.2.1 in the two-dimensional case where we arbitrarily choose $P = [2\ 1;\ 1\ 4]$ and $r = 1$. We can see that the set $\mathcal{S}'(P,r)$ (yellow circle with radius $r\sqrt{\min(\text{eig}(P))/\max(\text{eig}(P))}$ as in (4.2.33)) stays within the blue ellipsoid set $\mathcal{S}(P,r)$ defined in (4.2.18). Furthermore, the figure also provides our further direction which is to prove that there always exists the values of K as in (4.2.8) and M as in (4.2.13) such that an arbitrarily chosen initial state p_0 (illustrated by the red dot) belongs to the set $\mathcal{S}'(P,r)$ from (4.2.34), i.e.:

$$\frac{\min(\text{eig}(P))}{\max(\text{eig}(P))} r^2 \geq \|\mathsf{p}_0\|^2, \tag{4.2.35}$$

with P and r as employed in (4.2.32). In order to prove that assertion, an explicit formulation of P (which is obtained by solving the Lyapunov equation (4.2.6)) described in terms of the tuning parameters: K as in (4.2.8) and M as in (4.2.13) will be useful.

Proposition 4.2.8. *Let us consider the Lyapunov equation $A_K^\top P + P A_K + M = \mathbf{0}$ as in (4.2.6) with the stable matrix A_K as in (4.2.7) and the diagonal matrix M as in (4.2.13). Then, the matrix solution P of (4.2.6) is explicitly given by:*

$$P = \begin{bmatrix} \text{diag}\{P_{1_x}, P_{1_y}, P_{1_z}\} & \text{diag}\{P_{3_x}, P_{3_y}, P_{3_z}\} \\ \text{diag}\{P_{3_x}, P_{3_y}, P_{3_z}\} & \text{diag}\{P_{2_x}, P_{2_y}, P_{2_z}\} \end{bmatrix}, \tag{4.2.36}$$

with the positive scalars P_{i_q} *(*$i \in \{1,2,3\}$*,* $q \in \{x,y,z\}$*) calculated as follows:*

$$P_{1_q} = \frac{1}{2}\left(\frac{K_{d_q}}{K_{p_q}} - \frac{1}{K_{d_q}}\right) m_q + \frac{K_{p_q}}{2K_{d_q}} m_{v_q}, \tag{4.2.37}$$

$$P_{2_q} = \frac{m_q}{2K_{p_q}K_{d_q}} - \frac{m_{v_q}}{2K_{d_q}}, \tag{4.2.38}$$

$$P_{3_q} = -\frac{m_q}{2K_{p_q}}, \tag{4.2.39}$$

with (m_q, m_{v_q}) *positive parameters as in* (4.2.13) *and* (K_{p_q}, K_{d_q}) *negative control gains as in* (4.2.8) *(*$q \in \{x,y,z\}$*). Furthermore, the values of* $(P_{1_q}, P_{2_q}, P_{3_q})$ *given in* (4.2.37)–(4.2.39) *admit the following Lyapunov equation:*

$$A_{K_q}^\top P_q + P_q A_{K_q} + M_q = \mathbf{0}, \tag{4.2.40}$$

in which, the elements are defined as follows:

$$A_{K_q} = \begin{bmatrix} 0 & 1 \\ K_{p_q} & K_{d_q} \end{bmatrix}, \quad P_q = \begin{bmatrix} P_{1_q} & P_{3_q} \\ P_{3_q} & P_{2_q} \end{bmatrix}, \quad M_q = \begin{bmatrix} m_q & 0 \\ 0 & m_{v_q} \end{bmatrix}, \tag{4.2.41}$$

with (K_{p_q}, K_{d_q}) *as in* (4.2.8), $(P_{1_q}, P_{2_q}, P_{3_q})$ *as in* (4.2.36) *and* (m_q, m_{v_q}) *as in* (4.2.13) *with* $q \in \{x,y,z\}$.

Proof. The Lyapunov equation (4.2.6) is actually a full-rank system of linear equations in terms of P. I.e., if we introduce a complete parametrization of $P \in \mathbb{R}^{6\times 6}$ (36 variables) to the equation (4.2.6) with A_K as in (4.2.41) and M as in (4.2.13), then, we obtain 36 linear equations which definitely provide the solution for P. Furthermore, since M is symmetric, we obtain the symmetrical structure of P beforehand, hence, the number of variables is reduced to 21. Explicitly solving them leads to the results presented in (4.2.36)–(4.2.39).
We will illustrate the aforementioned solving process by applying it to the Lyapunov equation (4.2.40) which is explicitly given as follows:

$$\begin{bmatrix} 0 & K_{p_q} \\ 1 & K_{d_q} \end{bmatrix} \begin{bmatrix} P_{1_q} & P_{3_q} \\ P_{3_q} & P_{2_q} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ K_{p_q} & K_{d_q} \end{bmatrix} + \begin{bmatrix} m_q & 0 \\ 0 & m_{v_q} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \tag{4.2.42}$$

and then transformed into the full-rank system of linear equations:

$$\begin{bmatrix} 0 & K_{p_q} & K_{p_q} & 0 \\ 1 & K_{d_q} & 0 & K_{p_q} \\ 1 & 0 & K_{d_q} & K_{p_q} \\ 0 & 1 & 1 & 2K_d \end{bmatrix} \begin{bmatrix} P_{1_q} \\ P_{3_q} \\ P_{3_q} \\ P_{2_q} \end{bmatrix} = - \begin{bmatrix} m_q \\ 0 \\ 0 \\ m_{v_q} \end{bmatrix}, \tag{4.2.43}$$

with (m_q, m_{v_q}) as in (4.2.13), (K_{p_q}, K_{d_q}) as in (4.2.8) and $(P_{1_q}, P_{2_q}, P_{3_q})$ the variable to be solved ($q \in \{x,y,z\}$). Due to the symmetry of M_q, as defined in (4.2.40), the two middle equations is in fact equivalent. Then, solving (4.2.43) provides again the solutions given in (4.2.37)–(4.2.39), hence, completing the proof. □

After obtaining the parametrization of the matrix P as in (4.2.36), we further provide the formulations of its eigenvalues since they are employed in (4.2.32)–(4.2.33) and directly involved into the problem (4.2.35).

Corollary 4.2.9. *The eigenvalues of the matrix $P \in \mathbb{R}^{6\times 6}$ as in* (4.2.36) *consist of all the eigenvalues of the three matrices P_x, P_y, P_z defined in* (4.2.41)*:*

$$\text{eig}(P) = \big\{ \text{eig}(P_x),\ \text{eig}(P_y),\ \text{eig}(P_z)\big\}, \tag{4.2.44}$$

in which, the eigenvalues of the matrix P_q with $q \in \{x, y, z\}$ are given by:

$$\text{eig}(P_q) = \left\{ \frac{1}{2}\left(P_{1_q} + P_{2_q} \pm \sqrt{(P_{1_q} - P_{2_q})^2 + 4P_{3_q}^2} \right) \right\}, \tag{4.2.45}$$

with $(P_{1_q}, P_{2_q}, P_{3_q})$ as in (4.2.37)–(4.2.39)*.*

Proof. By defining $\mathsf{e} \in \mathbb{R}$ one eigenvalue and $\zeta \triangleq [\zeta_x\ \zeta_{v_x}]^\top \in \mathbb{R}^2$ as the corresponding eigenvector of the matrix $P_x \in \mathbb{R}^{2\times 2}$ from (4.2.41), we have that:

$$\begin{bmatrix} P_{1_x} & P_{3_x} \\ P_{3_x} & P_{2_x} \end{bmatrix} \begin{bmatrix} \zeta_x \\ \zeta_{v_x} \end{bmatrix} = \mathsf{e} \begin{bmatrix} \zeta_x \\ \zeta_{v_x} \end{bmatrix}. \tag{4.2.46}$$

Next, let us define a new vector $\zeta^* \in \mathbb{R}^6$ as follows:

$$\zeta^* = \big[\zeta_x\ 0\ 0\ \zeta_{v_x}\ 0\ 0\big]^\top, \tag{4.2.47}$$

with (ζ_x, ζ_{v_x}) two elements of eigenvector ζ as in (4.2.46). Then, it is straight forward to obtain:

$$P\zeta^* = \big[(P_{1_x}\zeta_x + P_{3_x}\zeta_{v_x})\ 0\ 0\ (P_{3_x}\zeta_{v_x} + P_{2_x}\zeta_x)\ 0\ 0\big]^\top, \tag{4.2.48}$$

with P as in (4.2.36). From (4.2.46), we also have $P_{1_x}\zeta_x + P_{3_x}\zeta_{v_x} = \mathsf{e}\zeta_x$ and $P_{3_x}\zeta_{v_x} + P_{2_x}\zeta_x = \mathsf{e}\zeta_{v_x}$, hence, it leads to $P\zeta^* = \mathsf{e}\zeta^*$. Therefore, e as in (4.2.46) and ζ^* as in (4.2.47) are one eigenvalue and the corresponding eigenvector of the matrix P from (4.2.36). Similar reasoning is also applied for the other eigenvalues of P_x and the matrices P_y, P_z from (4.2.41). The three matrices P_x, P_y, P_z from (4.2.41) are constructed independently, their eigenvalues are also independent with each other while the matrix $P \in \mathbb{R}^{6\times 6}$ can have only six eigenvalues, hence, the statement (4.2.44) is validated.
Next, the eigenvalues of the matrix P_q ($q \in \{x, y, z\}$) as given in (4.2.45) are straightforward to obtain by using the explicit formulation of P_q as in (4.2.41). This also completes the proof. □

By Proposition 4.2.8 and Corollary 4.2.9, we are now able to parametrize completely the proposed problem (4.2.35) in terms of the tuning parameters, i.e., the control gain matrix K as in (4.2.8) and the diagonal matrix M as in (4.2.13). In the following, we will point out a feasible choice of these parameters such that (4.2.35) holds with an arbitrarily chosen initial state p_0. The approach treats the motions along the three axes equally by choosing K from (4.2.8) and M from (4.2.13) as follows:

$$K_{p_x} = K_{p_y} = K_{p_z} = k_p, \tag{4.2.49}$$
$$K_{d_x} = K_{d_y} = K_{d_z} = k_d, \tag{4.2.50}$$
$$m_x = m_y = m_z = m_{v_x} = m_{v_y} = m_{v_z} = m_1, \tag{4.2.51}$$

with $(k_p, k_d) \in \mathbb{R}_-$ and $m_1 \in \mathbb{R}_+$ the new tuning parameters (m_1 is different from the system mass m employed in (4.1.4a)). Note that, due to the conditions (4.2.17) on choosing M as in (4.2.13), m_1 has to satisfy:

$$m_1 > \max\big\{Q_{1_q}^* + Q_{3_q}^*, Q_{2_q}^* + Q_{3_q}^*\big\},\ \forall q \in \{x, y, z\}, \tag{4.2.52}$$

with $(Q_{1_q}^*, Q_{2_q}^*, Q_{3_q}^*)$ defined in (4.2.14). By using the parametrization defined in (4.2.49)–(4.2.51), it is straightforward to obtain:

$$P_x = P_y = P_z, \tag{4.2.53}$$

$$r = \frac{\min\{U_x, U_y, U_z\}}{\sqrt{k_p^2 + k_d^2}}, \tag{4.2.54}$$

with (P_x, P_y, P_z) the matrices in $\mathbb{R}^{2\times 2}$ defined in (4.2.41) and r as in (4.2.19). By introducing (4.2.53) to the relation $\text{eig}(P) = \{\text{eig}(P_x), \text{eig}(P_y), \text{eig}(P_z)\}$ as in (4.2.44), we further achieve:

$$\min(\text{eig}(P)) = \min(\text{eig}(P_q)) \text{ and } \max(\text{eig}(P)) = \max(\text{eig}(P_q)), \tag{4.2.55}$$

with P_q representing for one among the three equal matrices (P_x, P_y, P_z). Then, by introducing the explicit formulation of $\text{eig}(P_q)$ as in (4.2.45) to (4.2.55), we obtain:

$$\min(\text{eig}(P)) = \frac{m_1}{4}\left(\frac{k_d}{k_p} + \frac{1}{k_p k_d} + \frac{k_p - 2}{k_d}\right) - \frac{m_1}{2}\sqrt{\frac{1}{4}\left(\frac{k_d}{k_p} - \frac{1}{k_p k_d} + \frac{k_p}{k_d}\right)^2 + \frac{1}{k_p^2}}, \tag{4.2.56}$$

$$\max(\text{eig}(P)) = \frac{m_1}{4}\left(\frac{k_d}{k_p} + \frac{1}{k_p k_d} + \frac{k_p - 2}{k_d}\right) + \frac{m_1}{2}\sqrt{\frac{1}{4}\left(\frac{k_d}{k_p} - \frac{1}{k_p k_d} + \frac{k_p}{k_d}\right)^2 + \frac{1}{k_p^2}}, \tag{4.2.57}$$

with (k_p, k_d, m_1) the three tuning parameters as in (4.2.49)–(4.2.51). Therefore, by using the eigenvalues of P as in (4.2.56)–(4.2.57) and the radius r as in (4.2.54), the problem (4.2.35) turns out into finding only the negative scalars (k_p, k_d) such that the following holds:

$$\underbrace{\frac{\min\{U_x^2, U_y^2, U_z^2\}\left((k_p-1)^2 + k_d^2 - \sqrt{(k_p^2 + k_d^2 - 1)^2 + 4k_d^2}\right)}{\left(k_p^2 + k_d^2\right)\left((k_p-1)^2 + k_d^2 + \sqrt{(k_p^2 + k_d^2 - 1)^2 + 4k_d^2}\right)}}_{r^2 \min(\text{eig}(P))/\max(\text{eig}(P))} \geq \|\mathsf{p}_0\|^2, \tag{4.2.58}$$

with $\mathsf{p}_0 \in \mathbb{R}^6$ an arbitrarily chosen initial state and (U_x, U_y, U_z) three positive scalars as in (4.1.6). We will not provide the explicit design law for the control gains (k_p, k_d) to satisfy the condition (4.2.58). However, we prove hereinafter the proof for guaranteeing the solution of (k_p, k_d) employed in (4.2.58). We will prove that the left-hand-side of (4.2.58) increases to infinity as long as the control gains (k_p, k_d) approaches minus zero:

$$\begin{aligned}
&\lim_{\substack{k_p\to -0\\ k_d\to -0}} \frac{(k_p-1)^2 + k_d^2 - \sqrt{(k_p^2 + k_d^2 - 1)^2 + 4k_d^2}}{\left(k_p^2 + k_d^2\right)\left((k_p-1)^2 + k_d^2 + \sqrt{(k_p^2 + k_d^2 - 1)^2 + 4k_d^2}\right)} \\
&= \lim_{\substack{k_p\to -0\\ k_d\to -0}} \frac{-4k_p\left(k_p^2 + k_d^2\right) + 8k_p^2 - 4k_p}{\left(k_p^2 + k_d^2\right)\underbrace{\left((k_p-1)^2 + k_d^2 + \sqrt{(k_p^2 + k_d^2 - 1)^2 + 4k_d^2}\right)^2}_{=4}} \\
&= \underbrace{\lim_{\substack{k_p\to -0\\ k_d\to -0}} \frac{-4k_p\left(k_p^2 + k_d^2\right)}{4\left(k_p^2 + k_d^2\right)}}_{=0} + \lim_{\substack{k_p\to -0\\ k_d\to -0}} \frac{8k_p^2 - 4k_p}{4\left(k_p^2 + k_d^2\right)} = \lim_{\substack{k_p\to -0\\ k_d\to -0}} \frac{2 - \frac{1}{k_p}}{1 + \frac{k_d^2}{k_p^2}} = +\infty,
\end{aligned} \tag{4.2.59}$$

if $\frac{k_d^2}{k_p^2} \not\to +\infty$ or even if $\frac{k_d^2}{k_p^2} \to +\infty$ but slower than $\frac{-1}{k_p}$.
Thus, by (4.2.59), we guarantee the solution of the condition (4.2.10), hence, be able to ensure the requirement $\mathsf{p}_0 \in \mathcal{S}'(P,r)$ as in (4.2.35) and also $\mathsf{p}_0 \in \mathcal{S}(P,r)$ as in (4.2.32). As a result, Proposition 4.2.7 is validated by choosing the tuning parameters as follows:

$$M := m_1 \boldsymbol{I}_6, \tag{4.2.60}$$

$$K := \begin{bmatrix} k_p \boldsymbol{I}_3 & k_d \boldsymbol{I}_3 \end{bmatrix}, \tag{4.2.61}$$

with m_1 the positive scalar satisfying (4.2.52) and (k_p, k_d) the negative gains satisfying (4.2.58). This also completes the proof for Proposition 4.2.7.

Corollary 4.2.10 (Extended version of Proposition 4.2.7 for a compact set $\mathcal{X}_0$)**.** *For any arbitrarily predefined compact set $\mathcal{X}_0 \in \mathbb{R}^6$ containing all the considered initial states of the system (4.1.1), there always exist some values of the negative control gains (k_p, k_d) as in (4.2.61) and the positive scalar m_1 as in (4.2.60) such that the invariant set $\mathcal{S}(P,r)$ as in (4.2.18) contains $\mathcal{X}_0$, i.e.:*

$$\mathcal{X}_0 \subset \mathcal{S}(P,r), \tag{4.2.62}$$

with P obtained as in (4.2.36) and r as in (4.2.54).

Proof. Since $\mathcal{X}_0$ is a compact set, it is possible to obtain a value D which is larger than any norm of all the elements within the set:

$$D \geq \|\mathsf{p}_0\|, \ \forall \mathsf{p}_0 \in \mathcal{X}_0. \tag{4.2.63}$$

Note that, D can be taken as a exact maximum value of all the norms (if applicable). Then, by using the result in (4.2.58), the control gains (k_p, k_d) can be chosen such that:

$$\frac{\min\{U_x^2, U_y^2, U_z^2\}\left((k_p-1)^2 + k_d^2 - \sqrt{(k_p^2+k_d^2-1)^2+4k_d^2}\right)}{\left(k_p^2+k_d^2\right)\left((k_p-1)^2 + k_d^2 + \sqrt{(k_p^2+k_d^2-1)^2+4k_d^2}\right)} \geq D^2, \tag{4.2.64}$$

with D as in (4.2.63) and (U_x, U_y, U_z) as in (4.1.6). The gains (k_p, k_d) as in (4.2.64) and the matrix M as in (4.2.60) will guarantee $\mathsf{p}_0 \in \mathcal{S}(P,r)$ as in (4.2.32) for all initial states p_0 having $\|\mathsf{p}_0\| \leq D$ (c.f. Figure 4.2.1). Then, due to the definition of D as in (4.2.63), Corollary 4.2.10 is validated, completing the proof. □

Example 4.2.11. *In this example, we illustrate Corollary 4.2.10 by considering a compact set $\mathcal{X}_0 \in \mathbb{R}^6$ containing all the static initial states defined as follows:*

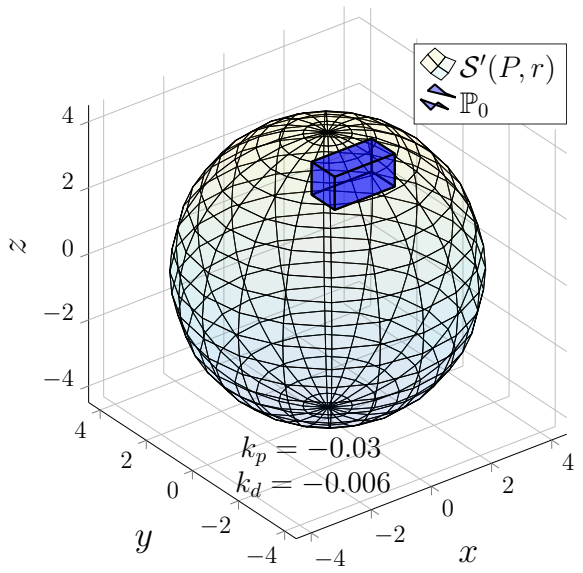
$$\mathcal{X}_0 = \left\{ [\xi_0 \ v_0]^\top \in \mathbb{R}^6 \middle| \ \xi_0 \in \mathbb{P}_0, \ v_0 = \mathbf{0} \right\}, \tag{4.2.65}$$

with the polytope $\mathbb{P}_0 \in \mathbb{R}^3$ defined as follows:

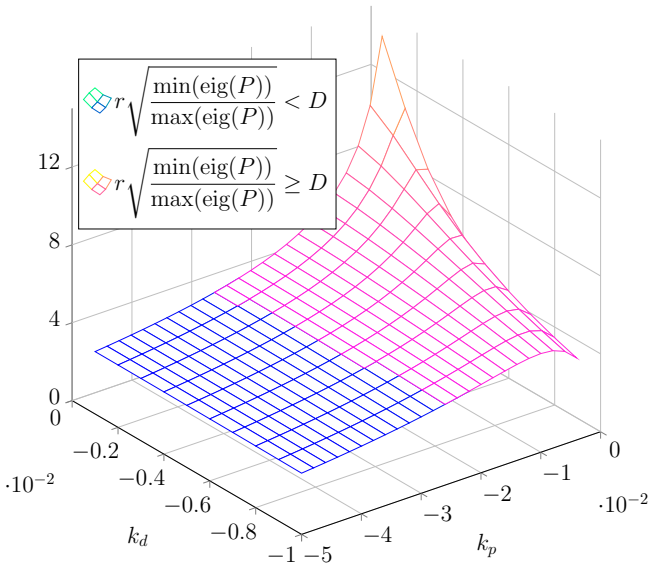
$$\mathbb{P}_0 = \text{Conv}\left(\begin{bmatrix}1\\1\\1\end{bmatrix}, \begin{bmatrix}3\\1\\1\end{bmatrix}, \begin{bmatrix}3\\2\\1\end{bmatrix}, \begin{bmatrix}1\\2\\1\end{bmatrix}, \begin{bmatrix}1\\1\\2\end{bmatrix}, \begin{bmatrix}3\\1\\2\end{bmatrix}, \begin{bmatrix}3\\2\\2\end{bmatrix}, \begin{bmatrix}1\\2\\2\end{bmatrix} \right), \tag{4.2.66}$$

which is illustrated as the blue rectangular box in Figure 4.2.2a. Then, the furthest distance D defined in (4.2.63) (i.e., having largest norm) of all the elements within the set $\mathcal{X}_0$ as in (4.2.65) is given by:

$$D = \sqrt{3^2+2^2+2^2} = 4.12. \tag{4.2.67}$$



(a) Illustratation in 3D space of the set $\mathcal{X}_0$ covering the set $\mathcal{S}'(P,r)$ defined in (4.2.33).



(b) Radius of $\mathcal{S}'(P,r)$ explicitly given in (4.2.64) w.r.t. different values of (k_p, k_d) (4.2.49)–(4.2.50).

Figure 4.2.2: Example on designing the terminal invariant set $\mathcal{S}(P,r)$ as in (4.2.18) containing a compact set $\mathcal{X}_0$.

Next, we will define the control gains (k_p, k_d) satisfying condition (4.2.64) with $D = 4.12$ and $\min\{U_x, U_y, U_z\} = 0.7168$ as given in Table 2.5.1. We show the values of the radius of the set $\mathcal{S}'(P,r)$ as in (4.2.33) w.r.t. different values of the control gains (k_p, k_d) in Figure 4.2.2b where the red part contains all the feasible choices, i.e., the radius is larger than $D = 4.12$ as required in (4.2.64). From these feasible choices, we choose the control gains $(k_p, k_d) = (-0.03, -0.006)$ and the corresponding set $\mathcal{S}'(P,r)$ defined in (4.2.33) is illustrated in Figure 4.2.2a by the meshed sphere in 3D space. Note that the sphere is defined by $x^2+y^2+z^2 \leq r^2 \min(\text{eig}(P))/\max(\text{eig}(P))$, hence, obviously being a subset of $\mathcal{S}'(P,r)$ as in (4.2.33). Therefore, the real invariant set $\mathcal{S}(P,q)$ defined in (4.2.18) and also the region of attraction of the NMPC design (4.2.1) using $\mathcal{S}(P,q)$ as its terminal constraint set (c.f. Proposition 4.2.5) surely contains all the compact set $\mathcal{X}_0$ as in (4.2.65). This validates the semi-global stability properties of the proposed NMPC design.

Procedure 4.2.12 (NMPC design using terminal invariant set with semi-global stability)**.** *Hereinafter, we summarize the design procedure of the NMPC controller (4.2.1)–(4.2.3) using the terminal invariant terminal set for stabilizing the multicopter's translation dynamics (4.1.1). The steps to follow are:*

1. *Choose the symmetric matrices $Q \in \mathbb{R}^{6\times 6}$ (positive definite) and $R \in \mathbb{R}^{3\times 3}$ (positive semi-definite) to formulate the stage cost as in (4.2.4).*

2. *Choose the negative control gains matrix (K_{p_q}, K_{d_q}) $(q \in \{x, y, z\})$ as in (4.2.8) and define the symmetric matrix M satisfying (4.2.10).*

3. *Solve the Lyapunov function (4.2.6) for the weighting matrix P which allows to formulate the terminal cost as in (4.2.5) and the terminal invariant set $\mathcal{S}(P,r)$ as in (4.2.79).*

4. *Choose the prediction horizon T_p as in (4.2.1).*

Remark 4.2.13. The most important step in Procedure 4.2.12 is at step 2 where the control gains (K_{p_q}, K_{d_q}) $(q \in \{x, y, z\})$ and the matrix M are defined. We have shown in Section 4.2.1.2

that the set $\mathcal{S}(P,r)$ as in (4.2.79) can be enlarged unlimitedly in order to cover any compact set $\mathcal{X}_0$ as in (4.2.62) containing all feasible initial states. However, a real efficient implementation will require only the domain of attraction to cover $\mathcal{X}_0$, hence, we have to balance between the size of the terminal invariant set $\mathcal{S}(P,r)$ (mostly affecting the convergence speed) and the prediction horizon T_p (mostly affecting the computing time). The tuning is particular for a specific scenario including both the construction of $\mathcal{X}_0$ and the computing hardware, and hence, cannot be explicitly instructed in this thesis. □

4.2.1.3 Simulation validation of NMPC design with terminal invariant set

In this section, we consider the simulation model (4.1.1)–(4.2.2) of a Crazyflie 2.0 nano-quadcopter platform [Nguyen et al., 2018b] with the parameters given in (2.2.47)–(2.2.49) and recapitulated hereinafter:

$$m = 0.028 \text{ kg}\ , \epsilon_{\max} = 10^\circ,\ T_{\max} = 0.55 \text{ N}. \tag{4.2.68}$$

The NMPC sampling time and the discretization step are chosen as the same value of $\delta = 0.1$ seconds (as employed in (4.2.3)). We will also employ the quasi-infinite horizon NMPC design (which makes use of a similar terminal invariant set but under the linear controller (3.4.2)) [Chen and Allgöwer, 1998] for comparison. Recalling the quasi-infinite horizon NMPC design procedure summarized in Section 3.4.1, the control gain matrix K_{q} as in (3.4.2) is chosen as follows:

$$K_{\mathrm{q}} = \begin{bmatrix} 0 & 0 & mk_p & 0 & 0 & mk_d \\ \sin(\psi)\frac{m}{g}k_p & -\cos(\psi)\frac{m}{g}k_p & 0 & \sin(\psi)\frac{m}{g}k_d & -\cos(\psi)\frac{m}{g}k_d & 0 \\ \cos(\psi)\frac{m}{g}k_p & \sin(\psi)\frac{m}{g}k_p & 0 & \cos(\psi)\frac{m}{g}k_d & \cos(\psi)\frac{m}{g}k_d & \end{bmatrix}, \tag{4.2.69}$$

with (k_p, k_d) negative control gains (as similar to (4.2.49)–(4.2.50)), m the system mass, g the gravity and ψ the yaw angle as in (4.1.1). The goal of choosing K_{q} as in (4.2.69) is to obtain the simple closed-loop matrix $A_{K_{\mathrm{q}}}$ (defined in (3.4.4)) as follows:

$$A_{K_{\mathrm{q}}} = \begin{bmatrix} 0_{3\times 3} & \boldsymbol{I}_3 \\ k_p\boldsymbol{I}_3 & k_d\boldsymbol{I}_3 \end{bmatrix}, \tag{4.2.70}$$

which is identical to the closed-loop system (4.2.7) under the FL controller (4.1.4).
We fix the initial state at $\mathsf{p}_0 = \begin{bmatrix} -0.1 & 0.3 & -0.2 & 0.8 & 0 & 0 \end{bmatrix}^\top$ and the yaw angle at $\psi = 0$. Three scenarios will be considered:

- *Scenario 1:* Using the proposed NMPC controller with the terminal invariant set $\mathcal{S}(P_1, r_1)$ such that $\mathsf{p}_0 \notin \mathcal{S}(P_1, r_1)$ with P_1, r_1 given in Table 4.2.1.
- *Scenario 2:* Using the proposed NMPC controller with the terminal invariant set $\mathcal{S}(P_2, r_2)$ such that $\mathsf{p}_0 \in \mathcal{S}(P_2, r_2)$ with P_2, r_2 given in Table 4.2.1.
- *Scenario 3:* Using the quasi-infinite horizon NMPC (qMPC) controller with the largest-possible terminal invariant set Ω_α as in (3.4.6).

The parameters of the proposed NMPC design (4.2.1)–(4.2.3) corresponding to the two first scenarios are gathered into Table 4.2.1. For Scenario 1, we arbitrarily choose $k_p = k_d = -2$ and then increase the prediction horizon up to $T_p = 0.7$ seconds while for Scenario 2, the terminal constraint set $\mathcal{S}(P_2, r_2)$ containing p_0 allows us to employ the shorter prediction horizon of $T_p = 0.2$ seconds. Next, the parameters of the qMPC controller are given in Table 4.2.2 in which the control gains (k_p, k_d) as in (4.2.69) are chosen to be equal to those under Scenario

Parameters	Scenario 1	Scenario 2
Q, R as in (4.2.4)	$\boldsymbol{I}_6$, $0.1\boldsymbol{I}_3$	
(k_p, k_d) as in (4.2.49)–(4.2.50)	$(-2, -2)$	$(-1, -1)$
M as in (4.2.13)	$2\boldsymbol{I}_6$	
P as in (4.2.6)	$P_1 = \begin{bmatrix} 2.5\boldsymbol{I}_3 & 0.5\boldsymbol{I}_3 \\ 0.5\boldsymbol{I}_3 & 0.75\boldsymbol{I}_3 \end{bmatrix}$	$P_2 = \begin{bmatrix} 3\boldsymbol{I}_3 & \boldsymbol{I}_3 \\ \boldsymbol{I}_3 & 2\boldsymbol{I}_3 \end{bmatrix}$
r in (4.2.19)	$r_1 = 0.2534$	$r_2 = 0.5069$
T_p as in (4.2.1)	0.7 seconds (7 steps)	0.2 seconds (2 steps)

Table 4.2.1: Parameters of the NMPC controllers with terminal invariant sets under the local FL controller.

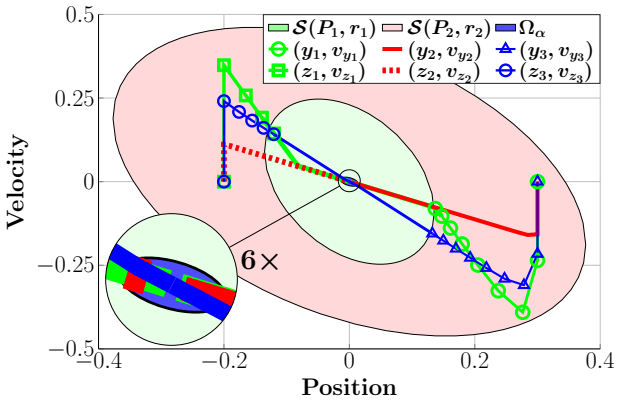
Parameters	Values
(k_p, k_d) as in (4.2.69)	$(-2, -2)$
κ as in (3.4.4)	0.9
P_{q} as in (3.4.5)	$\begin{bmatrix} 15.3960\boldsymbol{I}_3 & 7.1782\boldsymbol{I}_3 \\ 7.1782\boldsymbol{I}_3 & 6.9803\boldsymbol{I}_3 \end{bmatrix}$
α as in (3.4.6)	2.5×10^{-4}
T_p as in (3.1.5)	1.2 seconds (12 steps)

Table 4.2.2: Parameters of the quasi-infinite horizon NMPC controller.

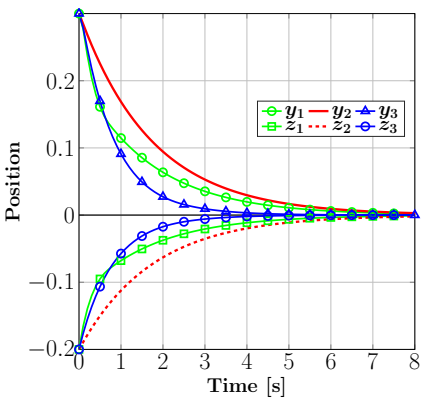
1 given in Table 4.2.1 for the two resulted linear systems (i.e., (4.2.7) and (4.2.70)) to have the same convergence speed. We provide the trajectories of (y, v_y) and (z, v_z) in comparison with the three terminal invariant sets in Figure 4.2.3a and the position's convergence results in Figure 4.2.3b which are plotted in green, red, blue for Scenarios 1, 2 and 3, respectively. Note that, since the sets are all in six-dimensional space, we have intentionally constructed them symmetrically such that the three 2D subspaces, obtained by slicing along (x, v_x), (y, v_y) and (z, v_z) are coincident with each other (c.f. the parameters given in Tables 4.2.1–4.2.2). Hence, they will be illustrated by their images on these three 2D spaces. E.g. the set $\mathcal{S}(P, r) \in \mathbb{R}^6$ as in (4.2.79) will be illustrated by the 2D set $P_{1_x}x^2 + P_{2_x}v_x^2 + 2P_{3_x}xv_x \leq \min(\mathrm{eig}(P))r^2$ since $P_{i_x} = P_{i_y} = P_{i_z},\ \forall i \in \{1, 2, 3\}$.

From Figure 4.2.3a, the semi-global stability property of the proposed NMPC design (c.f. Section 4.2.1.2) is proved emperically. It is straighforward to increase the size of the terminal constraint set from the set $\mathcal{S}(P_1, r_1)$ under Scenario 1 (green ellipsoid) to the larger set $\mathcal{S}(P_2, r_2)$ under Scenario 2 (red ellipsoid) by only increasing the control gains from $k_p = k_d = -2$ to $k_p = k_d = -1$ as shown in Table 4.2.1. In comparison, the tuning procedure of the qMPC controller is very exceedingly convoluted (c.f. Section 3.4.1), we have to define the control gains (k_p, k_d) as in (4.2.69), then, choose κ satisfying (3.4.4) without any specific information on the effect of these parameter [Chen and Allgöwer, 1998]. Furthermore, the next step is to tune the radius α of the terminal region Ω_α as in (3.4.6) such that the three conditions (3.4.7)–(3.4.9) are satisfied at once. Bear in mind that (3.4.7)–(3.4.9) are three nonlinear inequalities subject to the quadratic constraint (3.4.6) and hence, the value of α is hard to obtain. We give in Table 4.2.2 the best solution of α which we can find and the resulted invariant set Ω_α is illustrated by the blue ellipsoid in Figure 4.2.3a which can be seen only in the 6× zooming inset. The only advantage of using this small terminal constraint set is the fastest convergence speed (i.e., 2.6

seconds) as can be seen from blue lines in Figure 4.2.3b. However, the price to pay manifests itself in the large input effort as can be seen from Figure 4.2.4a (the blue lines have the largest amplitudes) and in the high computational burden as can be seen from Figure 4.2.4b (blue line with the mean value of 77 milliseconds) due to the required prediction horizon of 12 steps (c.f. Table 4.2.2). We emphasize that the maximum computing time of the qMPC controller reaches 109 millisecond while the predefined NMPC sampling time is $\delta = 100$ milliseconds, hence, being infeasible for real implementation.
On the contrary, our proposed method, with its larger terminal invariant sets $\mathcal{S}(P_1, r_1)$ and $\mathcal{S}(P_2, r_2)$ (c.f. Figure 4.2.3a) provide slower convergence speeds, i.e., 4.4 and 5.2 seconds as observed from Figure 4.2.3b (plotted in green and red lines for the two scenarios, respectively). However, the larger terminal invariant sets $\mathcal{S}(P_1, r_1)$ and $\mathcal{S}(P_2, r_2)$ allow us to reduce the prediction horizon to 7 steps and 2 steps (c.f. Table 4.2.1), hence, significantly reducing the computing time to average values at 55 and 42 milliseconds, respectively (as plotted in green and red lines in Figure 4.2.4b). The results strongly confirm the effectiveness of the proposed approach on using the FL controller (4.1.4) to design the NMPC scheme with guaranteed stability.

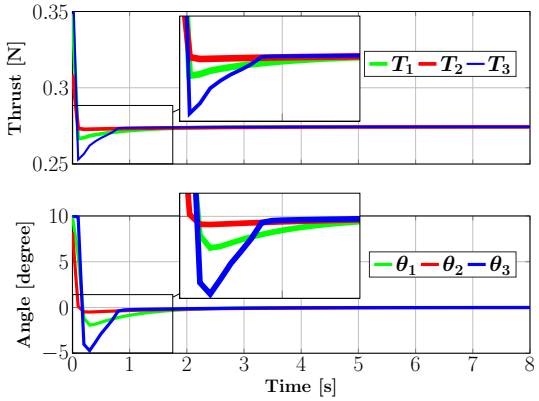


(a) Trajectories of (y, v_y) and (z, v_z) and terminal regions $\mathcal{S}(P_1, r_1)$, $\mathcal{S}(P_2, r_2)$ and Ω_α.

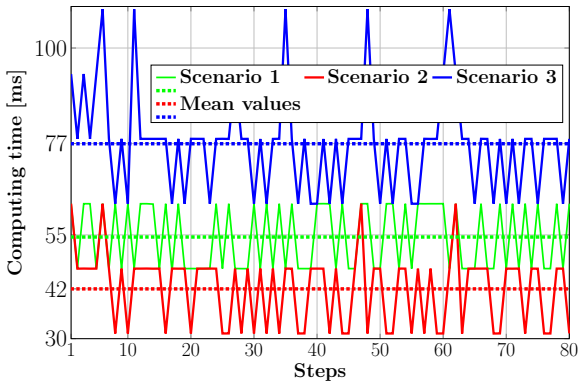


(b) Convergences of $y(t)$ and $z(t)$.

Figure 4.2.3: State convergences under NMPC controllers using terminal invariant sets.



(a) Results of thrust T and pitch angle θ.



(b) Computing time per step.

Figure 4.2.4: Inputs and computing time of three NMPC controllers using terminal invariant sets.

In the next section, we will continue with a new NMPC design using a relaxed invariant set as its terminal region. The construction allows to employ linear box-type terminal constraints, and hence, to further reduce the complexity of the optimization problem (4.2.1) and also the computation time.

4.2.2 NMPC design with relaxed invariant terminal constraint set

We propose in this section the design of the terminal constraint set $\mathcal{X}_f$ as in (4.2.2d) of the NMPC controller (4.2.1)–(4.2.3) which is described through simple box-type linear constraints. The proposed design makes use of the continuous-time δ-invariant set [Doban and Lazar, 2018] (with δ the sampling time of the NMPC controller as in (4.2.3)). The δ-invariant property is a relaxation of the standard invariant notion which allows the state trajectory to escape from the set and come back at predefined, periodic, time instants. Note that, the similar notion in discrete time is called a finite-step invariant set. I.e., in [Lazar and Spinu, 2015], a chain of polytopic finite-step invariant sets is used as the sequence of terminal sets for stabilizing an NMPC scheme. This allowed to reduce the complexity of the optimization problem since one polytopic terminal set will provide lower complexity than the standard invariant set (e.g., $\mathcal{S}(P,r)$ as in (4.2.18)) as also confirmed by the approach of approximating a standard ellipsoidal invariant set by a polytopic set [Cannon et al., 2003]. To easily illustrate the idea, the terminal constraint set $\mathcal{X}_f$ considered within this section admits the following formulation:

$$\mathcal{X}_f = \left\{ \boldsymbol{x} \in \mathbb{R}^6 \middle| \begin{cases} |x| \leq X_x, \ |v_x| \leq V_x \\ |y| \leq X_y, \ |v_y| \leq V_y \\ |z| \leq X_z, \ |v_x| \leq V_z \end{cases} \right\}, \tag{4.2.71}$$

with $X_q, V_q > 0$ the position and velocity limits along the three axes, chosen according to the design procedure presented in the following sections. Hence, we are introducing novel terminal constraints (as employed in (4.2.2d)) with clear interpretation to the NMPC formulation (4.2.1)–(4.2.3) for stabilizing system (4.1.1). They require that the terminal predicted position lies within the 3D rectangular box bounded in each direction by (X_x, X_y, X_z), respectively, and a small terminal predicted velocity along the three axes (bounded by (V_x, V_y, V_z), respectively). In the rest of this section, we will prove that the above formulation guarantees the stability of the NMPC controller (4.2.1)-(4.2.3) in a fashion similar to the standard ellipsoidal terminal sets (e.g., $\mathcal{S}(P,r)$ as in (4.2.18), or the *quasi-infinite horizon* NMPC design [Chen and Allgöwer, 1998] as summarized in Section 3.4.1) but reducing the complexity of the NMPC optimization problem (i.e., quadratic cost with linear constraints in comparison with quadratic cost with quadratic inequality constraints when employing the standard ellipsoidal invariant sets).
In the next section, we firstly present the characterization of a δ-invariant set and its usage within the NMPC design (4.2.1)–(4.2.3).

4.2.2.1 δ-invariant and safe sets characterization

Definition 4.2.14 (δ-invariant set)**.** *Consider the general nonlinear system:*

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}). \tag{4.2.72}$$

with $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{u} \in \mathbb{R}^m$ the state and input vectors[2]*.*
Given a real positive scalar δ and a controller $\boldsymbol{u}_{loc}(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}^m$, the compact set $\mathcal{R} \subseteq \mathbb{R}^n$ is called a δ-invariant set under $\boldsymbol{u}_{loc}(\boldsymbol{x})$ if the state trajectory of the system $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}_{loc}(\boldsymbol{x}))$ admits:

$$\text{if } \exists\ t_0 \geq 0 \text{ such that } \boldsymbol{x}(t_0) \in \mathcal{R} \text{ holds, then } \boldsymbol{x}(t_0 + \delta) \in \mathcal{R}. \tag{4.2.73}$$

[2]The general system (4.2.72) is defined similarly to (3.1.1) but without explicitly considering the system constraints as in (3.1.3). Also, the controller $\boldsymbol{u}_{\text{loc}}(\boldsymbol{x})$ is named according to its usage in the NMPC design principles presented in the following.

Furthermore, if the following also holds:

$$\boldsymbol{x}(t_0) \in \mathcal{R} \Rightarrow \boldsymbol{x}(t) \in \mathcal{B}_{\mathcal{R}},\ \forall t \geq t_0. \tag{4.2.74}$$

for the set $\mathcal{B}_{\mathcal{R}}$ with $\mathcal{R} \subseteq \mathcal{B}_{\mathcal{R}} \subseteq \mathbb{R}^n$, then, the set $\mathcal{B}_{\mathcal{R}}$ is called the safe set associated with the δ-invariant set $\mathcal{R}$.

Remark 4.2.15. Definition 4.2.14 is adapted from [Magni and Scattolini, 2004, Doban and Lazar, 2018, Lazar and Spinu, 2015] in which [Magni and Scattolini, 2004] does not name the set, [Doban and Lazar, 2018] provides the same definition for the δ-invariant set as in (4.2.73) while [Lazar and Spinu, 2015] similarly defines a *finite-step contractive set* in discrete time. For more details, [Doban and Lazar, 2018] uses the δ-invariant set for computing Lyapunov functions for nonlinear continuous-time differential equation via a Massera-type construction and [Lazar and Spinu, 2015] employs a finite number (undefined by the practitionners but resulted from the designing procedure) of finite-step contractive sets as a chain of terminal regions in an NMPC design. Also, [Wildhagen et al., 2019] uses "M-step invariant set" for stabilizing an NMPC with varying prediction horizon.
In comparison with [Magni and Scattolini, 2004, Lazar and Spinu, 2015], we exploit one polytopic (in a simple rectangle form) δ-invariant set $\mathcal{R}$ as the terminal region of a continuous-time NMPC controller while the associated safe set $\mathcal{B}_{\mathcal{R}}$ is required to be constraint admissible and ensures the existence of a local Lyapunov function within the set. □

Example on constructing a box-type δ-invariant set and the corresponding safe set for a double integrator system:
Let us consider an autonomous double integrator system:

$$\begin{bmatrix} \dot{\mathsf{x}}_1 \\ \dot{\mathsf{x}}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -s_1 s_2 & s_1 + s_2 \end{bmatrix} \begin{bmatrix} \mathsf{x}_1 \\ \mathsf{x}_2 \end{bmatrix}, \tag{4.2.75}$$

with $s_1 < s_2 < 0$, the system's real pole, defined in order to avoid the oscillator effect on the state response. Then, the state response of the system (4.2.75) is easily obtained for all $t \geq 0$ as follows:

$$\begin{bmatrix} \mathsf{x}_1(t) \\ \mathsf{x}_2(t) \end{bmatrix} = \underbrace{\begin{bmatrix} \lambda_1(s,t) & \lambda_2(s,t) \\ \lambda_3(s,t) & \lambda_4(s,t) \end{bmatrix}}_{\Lambda(s,t)} \begin{bmatrix} \mathsf{x}_1(0) \\ \mathsf{x}_2(0) \end{bmatrix}, \tag{4.2.76}$$

with $s = [s_1\ s_2]^\top$ and the auxiliary functions $\lambda_i(s,t)$, with $i \in \{1, \ldots, 4\}$ defined as follows:

$$\begin{aligned} \lambda_1(s,t) &= \frac{s_2 e^{s_1 t} - s_1 e^{s_2 t}}{s_2 - s_1}, & \lambda_2(s,t) &= \frac{e^{s_2 t} - e^{s_1 t}}{s_2 - s_1}, \\ \lambda_3(s,t) &= \frac{s_1 s_2 (e^{s_1 t} - e^{s_2 t})}{s_2 - s_1}, & \lambda_4(s,t) &= \frac{s_2 e^{s_2 t} - s_1 e^{s_1 t}}{s_2 - s_1}. \end{aligned} \tag{4.2.77}$$

Since we are interested in box-type δ-invariant set due to their simplicity (different shapes of the δ-invariant set do exist, e.g., ellipsoidal sets in [Magni and Scattolini, 2004] or general polytopic sets in [Lazar and Spinu, 2015]), we establish the following condition for the system (4.2.76):

$$\left| \begin{bmatrix} \mathsf{x}_1(\delta) \\ \mathsf{x}_2(\delta) \end{bmatrix} \right| = \left| \begin{bmatrix} \lambda_1(s,t) & \lambda_2(s,t) \\ \lambda_3(s,t) & \lambda_4(s,t) \end{bmatrix} \begin{bmatrix} \mathsf{x}_1(0) \\ \mathsf{x}_2(0) \end{bmatrix} \right| \leq \begin{bmatrix} X_{1_{\max}} \\ X_{2_{\max}} \end{bmatrix},\ \forall \left| \begin{bmatrix} \mathsf{x}_1(0) \\ \mathsf{x}_2(0) \end{bmatrix} \right| \leq \begin{bmatrix} X_{1_{\max}} \\ X_{2_{\max}} \end{bmatrix}, \tag{4.2.78}$$

with (X_1, X_2) positive state limits. The problem turns out to be finding s and (X_1, X_2) such that the condition (4.2.78) holds and this will be answered hereinafter.

Lemma 4.2.16. *For a real positive scalar δ, let us consider the set $\mathcal{P}_\delta$ bounding the possible pole values (s_1, s_2) of (4.2.76) which is defined as:*

$$\mathcal{P}_\delta = \left\{ s = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \in \mathbb{R}^2 | s_1 < s_2 < 0 \text{ and } \frac{\lambda_2(s,\delta)}{1-\lambda_1(s,\delta)} \leq \frac{1-|\lambda_4(s,\delta)|}{-\lambda_3(s,\delta)} \right\}, \tag{4.2.79}$$

with $\lambda_i(s,t)$ ($i \in \{1,\ldots,4\}$) defined as in (4.2.77). Choosing $s \in \mathcal{P}_\delta$ allows us to choose $X_1 > 0$ and $X_2 > 0$ such that:

$$\frac{\lambda_2(s,\delta)}{1-\lambda_1(s,\delta)} \leq \frac{X_1}{X_2} \leq \frac{1-|\lambda_4(s,\delta)|}{-\lambda_3(s,\delta)}. \tag{4.2.80}$$

Then, the sets $\mathcal{R}_{\mathsf{x}}$ and $\mathcal{B}_{\mathcal{R}_{\mathsf{x}}}$ defined as follows are the δ-invariant set and the corresponding safe set of the double integrator system (4.2.75):

$$\mathcal{R}_{\mathsf{x}} = \left\{ \mathsf{x} = \begin{bmatrix} \mathsf{x}_1 \\ \mathsf{x}_2 \end{bmatrix} \in \mathbb{R}^2 \middle| \ |\mathsf{x}_1| \leq X_1, |\mathsf{x}_2| \leq X_2 \right\}, \tag{4.2.81}$$

$$\mathcal{B}_{\mathcal{R}_{\mathsf{x}}} = \left\{ \mathsf{x} = \begin{bmatrix} \mathsf{x}_1 \\ \mathsf{x}_2 \end{bmatrix} \in \mathbb{R}^2 \middle| \ |\mathsf{x}_1| \leq \widetilde{X}_1, |\mathsf{x}_2| \leq \widetilde{X}_2 \right\}, \tag{4.2.82}$$

in which, $\widetilde{X}_1 = \chi(X_1, X_2, s, \delta)$ and $\widetilde{X}_2 = \nu(X_1, X_2, s, \delta)$ with the two functions $\chi(\cdot), \nu(\cdot)$ defined by the following optimization problems:

$$\chi(a,b,s,\delta) = \max_{t\in[0,\delta]} \left\{ a\lambda_1(s,t) + b\lambda_2(s,t) \right\}, \tag{4.2.83a}$$

$$\nu(a,b,s,\delta) = \max_{t\in[0,\delta]} \left\{ -a\lambda_3(s,t) + b|\lambda_4(s,t)| \right\}. \tag{4.2.83b}$$

Proof. The detailed proof is given in Appendix E and only the main idea is sketched here. The goals of choosing $s \in \mathcal{P}_\delta$ are twofold: i) we need to ensure the closed-loop stability of the linear system (4.2.75) by imposing $s_1 < s_2 < 0$, ii) the second constraint guarantees the consistency of (4.2.80), i.e., there always exists a value of X_1/X_2 satisfying (4.2.80). Then, by choosing $X_1 > 0$ and $X_2 > 0$ satisfying (4.2.80), we guarantee the maximum values of $|\mathsf{x}_1(\delta)|$ and $|\mathsf{x}_2(\delta)|$ obtained as in (4.2.78) are bounded by X_1 and X_2, respectively, as proved in (E.0.6). Thus, the condition (4.2.78) holds and the δ-invariant property (4.2.73) of the set $\mathcal{R}_{\mathsf{x}}$ (4.2.81) is established.
Furthermore, $\widetilde{X}_1$ as in (4.2.83a) and $\widetilde{X}_2$ as in (4.2.83b) are the maximum values of $|\mathsf{x}_1(t)|$ and $|\mathsf{x}_2(t)|$ as defined in (4.2.76) for all $t \in [0,\delta]$ and for any initial condition $[\mathsf{x}_1(0)\ \mathsf{x}_2(0)]^\top \in \mathcal{R}_{\mathsf{x}}$ (4.2.81). Thus, we have that $|\mathsf{x}_1(t)| \leq \widetilde{X}_1$ and $|\mathsf{x}_2(t)| \leq \widetilde{X}_2$ which are equivalent to $[\mathsf{x}_1(t)\ \mathsf{x}_2(t)]^\top \in \mathcal{B}_{\mathcal{R}_{\mathsf{x}}}$ (4.2.82) for all $t \in [0,\delta]$.
Then, as $[\mathsf{x}_1(\delta)\ \mathsf{x}_2(\delta)]^\top \in \mathcal{R}_{\mathsf{x}}$ (4.2.81) (obtained from the δ-invariant property of the set $\mathcal{R}_{\mathsf{x}}$), further recursive analysis provides $[\mathsf{x}_1(t)\ \mathsf{x}_2(t)]^\top \in \mathcal{B}_{\mathcal{R}_{\mathsf{x}}}$ for all $t \geq 0$, completing the proof. □

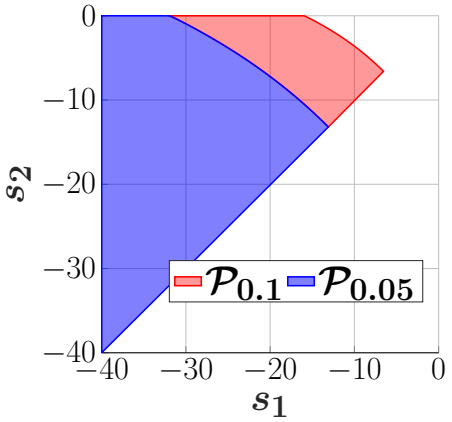
Example 4.2.17. *Let us illustrate Lemma 4.2.16 by considering two values of the time step $\delta_0 = 0.05$ seconds and $\delta = 0.1$ seconds. At first, the sets $\mathcal{P}_{\delta_0}$ and $\mathcal{P}_\delta$ from (4.2.79) are found numerically and illustrated in Figure 4.2.5a within the range of $-40 \leq s_1 < s_2 < 0$ (small blue region for $\mathcal{P}_{\delta_0}$ and red region for $\mathcal{P}_\delta \subset \mathcal{P}_{\delta_0}$). Note that, the figure clips the two sets $\mathcal{P}_{\delta_0}$ and $\mathcal{P}_\delta$ which are unbounded. Having unbounded sets means that an infinite choice of poles satisfying Lemma 4.2.16 is possible. It is worth pointing out that $\mathcal{P}_{\delta_0} \subset \mathcal{P}_\delta$ (as observed in Figure 4.2.5a). The reason is that from any pole vector $s \in \mathcal{P}_{\delta_0}$, we can design the associating δ_0-invariant set which always satisfies the δ-invariant property since $\delta = 2\delta_0$. Thus, any feasible pole vector $s \in \mathcal{P}_{\delta_0}$ is also a possible choice for the case of δ but not vice versa.*
As the time step $\delta = 0.1$ seconds will be used for controlling the system (4.1.1) later, we will

construct only the 0.1*-invariant set* $\mathcal{R}_x$ *and its associating safe set* $\mathcal{B}_{\mathcal{R}_x}$ *as defined in* (4.2.81)–(4.2.82). *Firstly, from Figure 4.2.5a,* $s = [-16 \;\; -0.5]^\top \in \mathcal{P}_{0.1}$ *(red region) is a feasible choice which provides us with* $X_1/X_2 = 2$ *satisfying* (4.2.80). *Then, by arbitrarily choosing* $X_1 = 2$ *and* $X_2 = 1$, $\widetilde{X}_1$ *and* $\widetilde{X}_2$ *as in* (4.2.82) *are obtained by solving the optimization problems given in* (4.2.83) *with* $s_1 = -16$ *and* $s_2 = -0.5$ *as follows:*

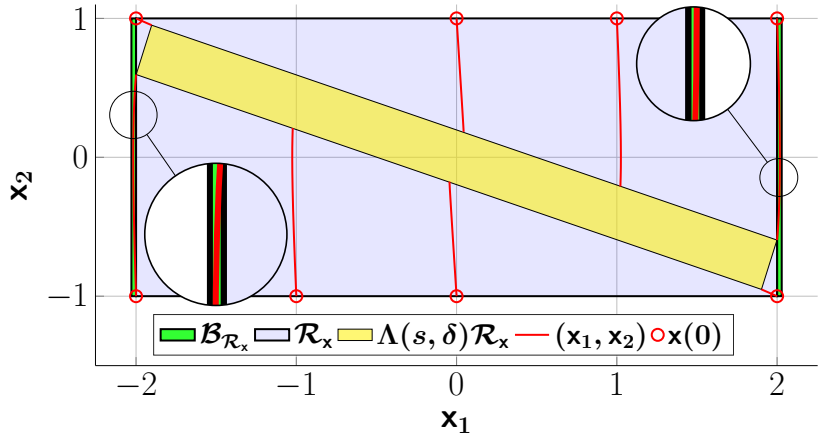
$$\widetilde{X}_1 = \max_{t\in[0,0.1]} \left\{ 2\frac{s_2 e^{s_1 t} - s_1 e^{s_2 t}}{s_2 - s_1} + \frac{e^{s_2 t} - e^{s_1 t}}{s_2 - s_1} \right\}, \tag{4.2.84}$$

$$\widetilde{X}_2 = \max_{t\in[0,0.1]} \left\{ -2\frac{s_1 s_2 (e^{s_1 t} - e^{s_2 t})}{s_2 - s_1} + \left| \frac{s_2 e^{s_2 t} - s_1 e^{s_1 t}}{s_2 - s_1} \right| \right\},$$

which are straightforward to obtain, $\widetilde{X}_1 = 2.0189$ *and* $\widetilde{X}_2 = X_2 = 1$. *The* 0.1*-invariant set* $\mathcal{R}_x$ *(blue rectangle) and its safe set* $\mathcal{B}_{\mathcal{R}_x}$ *(green rectangle covering wholly the blue one) are illustrated in Figure 4.2.5b. The yellow polytopic region* $\Lambda(s,\delta)\mathcal{R}_x$ *with* $\Lambda(\cdot)$ *as in* (4.2.76) *is the image of*



(a) Illustration of the two sets $\mathcal{P}_{0.05}$ and $\mathcal{P}_{0.1}$.



(b) The δ-invariant set $\mathcal{R}_x$ and the safe set $\mathcal{B}_{\mathcal{R}_x}$ (with δ=0.1 seconds, $s = [-16 \;\; -0.5]^\top$, $X_1 = 2$, $\widetilde{X}_1 = 2.0189$ and $X_2 = \widetilde{X}_2 = 1$).

Figure 4.2.5: Construction of δ-invariant set and safe set for the double integrator system (4.2.75).

the set $\mathcal{R}_x$ *under the map* (4.2.76), *i.e., its evolution in one step under the dynamics:*

$$\Lambda(s,\delta)\mathcal{R}_x = \{x \in \mathbb{R}^2 | x = \Lambda(s,\delta)x_0, \; x_0 \in \mathcal{R}_x\}, \tag{4.2.85}$$

which stays within the set $\mathcal{R}_x$. *This clearly shows the* 0.1*-invariant property of* $\mathcal{R}_x$. *For more details, there are also illustrations of eight state trajectories* (x_1, x_2) *(red lines) starting from eight different initial conditions (red circles) at* $t = 0$ *within the* 0.1*-invariant set* $\mathcal{R}_x$ *and ending at* $t = 0.1$ *seconds. There are two trajectories which first go out of* $\mathcal{R}_x$ *(can be seen from the right enlarging inset) but all the eight trajectories stay within the safe region* $\mathcal{B}_{\mathcal{R}_x}$ *and arrive inside the set* $\mathcal{R}_x$ *after* 0.1 *seconds. The trajectories are obtained by explicitly solving the linear system* (4.2.75), *and then, by plotting the continuous-time solutions with the time step of* 0.01 *seconds much smaller than* 0.1 *seconds.* □

4.2.2.2 δ-invariant set for the multicopter's translation system

This section introduces the design of the δ-invariant set associated to the FL linearization controller $u_{\text{FL}}(\cdot)$ as in (4.2.9). The set is constructed by using the resulted linearized system (4.2.7)

and hence, the construction makes use of the results (4.2.81)–(4.2.82) corresponding to the double integrator system (4.2.75) detailed in the previous section.
At first, let us consider the sampling time δ (4.2.3) of the NMPC controller (4.2.1)–(4.2.3) and choose the pole vector $s_q \in \mathbb{R}^2$ such that:[3]

$$s_q \triangleq [s_{1_q}\ s_{2_q}]^\top \in \mathcal{P}_\delta,\ q \in \{x, y, z\}, \tag{4.2.86}$$

with $\mathcal{P}_\delta$ as in (4.2.79). Next, the control gains (K_{p_q}, K_{d_q}) as employed in (4.2.8) are defined as follows:

$$K_{p_q} = -s_{1_q} s_{2_q}, \qquad K_{d_q} = s_{1_q} + s_{2_q}. \tag{4.2.87}$$

This allows to define the three pairs of position $X_q > 0$ and velocity $V_q > 0$ such that:

$$\frac{\lambda_2(s_q, \delta)}{1 - \lambda_1(s_q, \delta)} \leq \frac{X_q}{V_q} \leq \frac{1 - |\lambda_4(s_q, \delta)|}{-\lambda_3(s_q, \delta)}, \tag{4.2.88}$$

$$\left|K_{p_q}\right| \widetilde{X}_q + \left|K_{d_q}\right| \widetilde{V}_q \leq U_q, \tag{4.2.89}$$

with $\lambda_i(\cdot)$, $i \in \{1, \ldots, 4\}$ defined in (4.2.77) and U_q as in (4.1.6). The parameters $(\widetilde{X}_q, \widetilde{V}_q)$ are given by:

$$\widetilde{X}_q \triangleq \chi(X_q, V_q, s_q, \delta),\ \widetilde{V}_q \triangleq \nu(X_q, V_q, s_q, \delta), \tag{4.2.90}$$

with $\chi(\cdot)$, $\nu(\cdot)$ as in (4.2.83a)-(4.2.83b). Then, the δ-invariant set and its corresponding safe set of the system (4.1.1) are given in the following.

Lemma 4.2.18. *Let us define two sets $\mathcal{R}$ and $\mathcal{B}_\mathcal{R}$ as follows:*

$$\mathcal{R} = \left\{ \mathsf{p} \in \mathbb{R}^6 \middle| \begin{cases} |x| \leq X_x,\ |v_x| \leq V_x \\ |y| \leq X_y,\ |v_y| \leq V_y \\ |z| \leq X_z,\ |v_z| \leq V_z \end{cases} \right\}, \tag{4.2.91}$$

$$\mathcal{B}_\mathcal{R} = \left\{ \mathsf{p} \in \mathbb{R}^6 \middle| \begin{cases} |x| \leq \widetilde{X}_x,\ |v_x| \leq \widetilde{V}_x \\ |y| \leq \widetilde{X}_y,\ |v_y| \leq \widetilde{V}_y \\ |z| \leq \widetilde{X}_z,\ |v_x| \leq \widetilde{V}_z \end{cases} \right\}, \tag{4.2.92}$$

with $X_q, V_q, \widetilde{X}_q, \widetilde{V}_q$ as in (4.2.88)–(4.2.89). Then, for the system (4.1.1) controlled by the FL controller $u_{FL}(K\mathsf{p}, \psi)$ as in (4.2.9) with K as in (4.2.87), we have that:

(i) $\mathcal{B}_\mathcal{R}$ is input constraints admissible.

(ii) $\mathcal{R} \subset \mathcal{B}_\mathcal{R}$ is δ-invariant w.r.t. the safe set $\mathcal{B}_\mathcal{R}$ with δ the sampling time as in (4.2.3). □

Proof. By using $\mu_\xi = K\mathsf{p}$ with $\mu_\xi = [\mu_x\ \mu_y\ \mu_z]^\top$, the virtual input vector as in (4.1.4) and the gain matrix K from (4.2.8), we obtain the bounds on the virtual inputs μ_q ($q \in \{x, y, z\}$):

$$|\mu_q| \leq \left|K_{p_q} q\right| + \left|K_{d_q} v_q\right|. \tag{4.2.93}$$

Next, for all $\mathsf{p} \in \mathcal{B}_\mathcal{R}$ as in (4.2.92), we have that $|q| \leq \widetilde{X}_q$, $|v_q| \leq \widetilde{V}_q$) which further leads to:

$$|\mu_q| \leq \left|K_{p_q}\right| \widetilde{X}_q + \left|K_{d_q}\right| \widetilde{V}_q \leq U_q. \tag{4.2.94}$$

[3]Notation with subscript q, e.g., s_q, stands for three similar notations with subscript x, y, z, e.g., s_x, s_y, s_z.

in which the second inequality is due to (K_{p_q}, K_{d_q}) satisfying the condition (4.2.89). Then, $\mathcal{B}_\mathcal{R} \subset \mathcal{X}_{\text{FL}}$ with $\mathcal{X}_{\text{FL}}$ the input constraint admissible set as in (4.1.6), hence, (i) is validated.
At the next step, for all $\mathsf{p} \in \mathcal{B}_\mathcal{R}$, with $\mathcal{B}_\mathcal{R}$ the admissible set obtained from (i), the FL controller $u_{\text{FL}}(K\mathsf{p}, \psi)$ as in (4.2.9) linearizes the dynamics (4.1.1) into the linear stable system (4.2.7). Let us consider only the dynamics of x, v_x partly taken from (4.2.7) as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{v}_x \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ K_{p_x} & K_{d_x} \end{bmatrix} \begin{bmatrix} x \\ v_x \end{bmatrix}, \tag{4.2.95}$$

which is actually the double integrator system from (4.2.75). Then, with X_x, V_x satisfying the condition (4.2.88), $|x(0)| \leq X_x$ and $|v_x(0)| \leq V_x$ imply that $|x(\delta)| \leq X_x$, $|v_x(\delta)| \leq V_x$ and $|x(t)| \leq \widetilde{X}_x$, $|v_x(t)| \leq \widetilde{V}_x$ as proved in Lemma 4.2.16. Similar results are obtained for (y, v_y) and (z, v_z). Thus, (ii) is validated, completing the proof. □

Remark 4.2.19. The sizes of the box-type δ-invariant set $\mathcal{R}$ and the corresponding safe set as defined in (4.2.91)–(4.2.92) are easy to tune by applying the following procedure:

i) choose the poles s_q with $q \in \{x, y, z\}$ as in (4.2.86);

ii) define the ratio X_q/V_q satisfying condition (4.2.88);

iii) tune the values of (X_q, V_q) while maintaining their predefined ratio and checking the constraint (4.2.89).

Note that, the larger values of (X_q, V_q) the larger the δ-invariant set $\mathcal{R}$ (4.2.91) becomes. The set $\mathcal{R}$ can also be enlarged significantly but its limit is still under question (in comparison with the unlimited expandability of the invariant set $\mathcal{S}(P, r)$ as detailed in Proposition 4.2.7). The difficulties on analyzing this problem are twofold: i) the control gains are limited to the choice of the poles $s_q \in \mathcal{P}_\delta$ as in (4.2.86) and ii) the implicit relation of $(\widetilde{X}, \widetilde{V})$ and (X, V) as in (4.2.90). These problems are worthy of further consideration. □

4.2.2.3 NMPC design using terminal δ-invariant set

This section firstly presents the principles for an NMPC scheme employing a δ-invariant terminal constraint set with δ the sampling time of the NMPC controller for a general constrained nonlinear system. The design guarantees the recursive feasibility and asymptotic stability of the closed-loop controlled system in a manner similar to the design using a standard terminal invariant set proposed in Lemma 3.1.1. Then, we complete the NMPC design (4.2.1)–(4.2.3) for the multicopter's translation system (4.1.1) by using the terminal δ-invariant set as in (4.2.91).

Lemma 4.2.20. *Let us consider the NMPC setup* (3.1.4)-(3.1.7) *for stabilizing the general system* (3.1.1) *around the desired equilibrium point* $(\boldsymbol{x}_e, \boldsymbol{u}_e)$ *as in* (3.1.2). *Assume that there exists a local controller* $\boldsymbol{u}_{loc}(\boldsymbol{x})$ *under which, the system* (3.1.1), *i.e.,* $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}_{loc}(\boldsymbol{x}))$ *admits a pair of* δ*-invariant,* $\mathcal{R}$, *and safe,* $\mathcal{B}_\mathcal{R}$ *sets as in* (4.2.73)-(4.2.74). *Then, the recursive feasibility and asymptotic stability of the closed-loop system controlled by the NMPC controller* (4.2.3) *can be achieved if the following design conditions hold:*

C1*: [State constraints satisfied in $\mathcal{B}_\mathcal{R}$]. *The safe set* $\mathcal{B}_\mathcal{R}$ *satisfies:*

$$\mathcal{B}_\mathcal{R} \subseteq \mathcal{X}, \ \boldsymbol{x}_e \in \mathcal{B}_\mathcal{R}, \tag{4.2.96}$$

with $\mathcal{X}$ *the state constraint set as in* (3.1.3) *and* $\boldsymbol{x}_e$ *the equilibrium point as in* (3.1.2).

C2**: [Input constraints satisfied in $\mathcal{B}_\mathcal{R}$]. *The local controller* $\boldsymbol{u}_{loc}(\boldsymbol{x})$ *respects the input constraints within the safe set:*

$$\boldsymbol{u}_{loc}(\boldsymbol{x}) \in \mathcal{U}, \ \forall \boldsymbol{x} \in \mathcal{B}_\mathcal{R}, \tag{4.2.97}$$

with $\mathcal{U}$ *the input constraint set as in* (3.1.2).

C3**: [Terminal δ-invariant set]. *The δ-invariant set* $\mathcal{R}$ *serves as the terminal region* $\mathcal{X}_f$ *as in* (3.1.6d) *of the NMPC design.*

$$\mathcal{X}_f := \mathcal{R}. \tag{4.2.98}$$

C4**: [Local Lyapunov function existence]. *Starting from any* $\boldsymbol{x} \in \mathcal{R}$*, the trajectory of the system* (3.1.1) *under* $\boldsymbol{u}_{loc}(\boldsymbol{x})$ *satisfies:*

$$\frac{dF(\boldsymbol{x})}{dt} + \ell(\boldsymbol{x}, \boldsymbol{u}_{loc}(\boldsymbol{x})) \leq 0, \tag{4.2.99}$$

where $F(\boldsymbol{x})$ *and* $\ell(\boldsymbol{x}, \boldsymbol{u}_{loc})$ *are the terminal and stage costs from* (3.1.5)*, respectively.* □

Proof. The proof is explicitly given in Appendix F. It is constructed similarly to the one of Lemma 3.1.1 (i.e., desinging an NMPC with terminal invariant set) as given in [Mayne et al., 2000] but with the slight difference in the fact that the safe set $\mathcal{B}_\mathcal{R}$ is required to be constraint admissible as in (4.2.96)–(4.2.97) while the δ-invariant set $\mathcal{R}$ is employed as the terminal region (4.2.98). It is because all the state trajectories starting from the δ-invariant set $\mathcal{R}$ lie within the safe set and come back to the set $\mathcal{R}$ at the next sampling time. This ensures the recursive feasibility of the design while the fourth condition **C4*** (4.2.99) guarantees the asymptotic stability of the closed-loop system in the same manner as the condition **C4** given in (3.1.10). □

Lemma 4.2.20 opens the path on using the δ-invariant set as the terminal constraint set for guaranteeing the stability of an NMPC design. In the following, we will apply the method for desinging the NMPC controller for the multicopter's translation system (4.1.1).

Proposition 4.2.21. *Let us consider the NMPC design given in* (4.2.1)–(4.2.5) *for stabilizing the system* (4.1.1)*. By using the terminal δ-invariant set $\mathcal{R}$ as defined in* (4.2.91) *(with δ the NMPC sampling time as in* (4.2.3)*):*

$$\mathcal{X}_f := \mathcal{R}, \tag{4.2.100}$$

the closed-loop controlled system achieves (nominal) asymptotic stability.

Proof. The proof is constructed by guaranteeing the satisfaction of the four conditions **C1***–**C4*** as in (4.2.96)–(4.2.99). The three first conditions are obviously validated by using the δ-invariant set $\mathcal{R}$ and its corresponding safe set $\mathcal{B}_\mathcal{R}$ as defined in (4.2.91)–(4.2.92). Then, the fourth condition (4.2.99) is ensured by using the result of Proposition 4.2.1 which points out that (4.2.99) is validated for all $\mathsf{p} \in \mathcal{X}_{\text{FL}}$ with $\mathcal{X}_{\text{FL}}$ the input constriant admissible set as in (4.1.6) while any trajectories starting from the δ-invariant set $\mathcal{R}$ remains within the safe set $\mathcal{B}_\mathcal{R}$ as in (4.2.91)–(4.2.92) and $\mathcal{B}_\mathcal{R} \subset \mathcal{X}_{\text{FL}}$ as proved in (4.2.94), hence, completing the proof. □

Procedure 4.2.22 (NMPC design using terminal δ-invariant set)**.** *The design procedure of the NMPC controller* (4.2.1)–(4.2.3) *using the terminal δ-invariant terminal set (with δ the controller sampling time) for stabilizing the multicopter's translation dynamics* (4.1.1) *is summarized hereinafter. The steps to follow are:*

1. *Choose the pole vectors* $s_q \in \mathcal{P}_\delta$ *as in* (4.2.86) *and define the position and velocity limits* (X_q, V_q) *satisfying* (4.2.88)–(4.2.89) *(with* $q \in \{x, y, z\}$*). Then, establish the terminal δ-invariant set $\mathcal{R}$ as in* (4.2.91).

2. *Choose the symmetric matrices $Q \in \mathbb{R}^{6\times 6}$ (positive definite) and $R \in \mathbb{R}^{3\times 3}$ (positive semi-definite) to formulate the stage cost as in* (4.2.4).

3. *Define the symmetric matrix M satisfying* (4.2.10)*, then, solve the Lyapunov function* (4.2.6) *to obtain the weighting matrix P of the terminal cost as in* (4.2.5).

4. *Choose the prediction horizon T_p as in* (4.2.1).

Remark 4.2.23. Choosing T_p at step 4 of the Procedure 4.2.22 requires to take into account the computational constraints of the platform (e.g., the processing speed requirement) as well as to ensure that the first NMPC iteration is successful. Bear in mind that when the prediction horizon T_p is already large, one can increase the size of the δ-invariant set $\mathcal{R}$ defined (4.2.91) instead. The solution is similar to the enlarging approach introduced for the invariant set $\mathcal{S}(P, r)$ from (4.2.79) as detailed in Section 4.2.1.2 but using the different tuning method discussed in Remark 4.2.19. □

4.2.2.4 Simulation validation of NMPC design with δ-invariant set

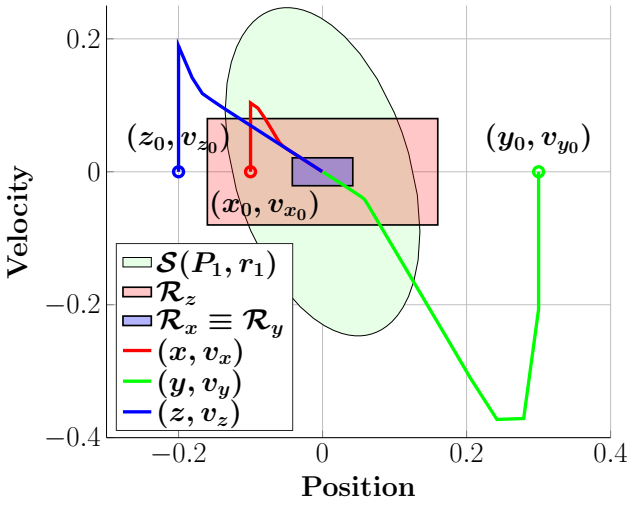
In this section, we present the simulation results for stabilizing the model of a Crazy flie 2.0 nano quadcopter platform characterized by the parameters given in (4.2.68) using the NMPC design with terminal δ-invariant set as detailed in Proposition 4.2.21. The simulation employs the same parameters as considered in Section 4.2.1.3, i.e., the NMPC sampling time and model discretization step $\delta = 0.1$ seconds, the initial state $\mathsf{p}_0 = [-0.1\ 0.3\ -0.2\ 0\ 0\ 0]^\top$, the weighting matrices $Q = \boldsymbol{I}_6$ and $R = 0.1\boldsymbol{I}_3$ as employed in (4.2.4). Also, we will make comparisons with the results obtained by using the standard invariant set $\mathcal{S}(P_1, r_1)$ (c.f. Figure 4.2.3a and P_1, r_1 detailed in Table 4.2.1) under Scenario 1 as given in Section 4.2.1.3 since they share various similarities, e.g., simulation scenario, sizes of terminal regions. Further parameters related to the δ-invariant set construction can be found in Table 4.2.3. We provide the illustration of

Parameters	Values
(U_x, U_y, U_z) as in (4.2.89)	$(0.7168, 0.7168, 2.597)$
(s_1, s_2) as in (4.2.86)	$(-16, -0.5)$
(K_{p_q}, K_{d_q}) as in (3.4.4)	$(-8, -16.5),\ \forall q \in \{x, y, z\}$
X_q/V_q as in (4.2.88)	$2,\ \forall q \in \{x, y, z\}$
(X_x, V_x) as in (4.2.91)	$(0.042, 0.021)$
(X_y, V_y) as in (4.2.91)	$(0.042, 0.021)$
(X_z, V_z) as in (4.2.91)	$(0.16, 0.08)$
M as in (4.2.6)	$20\boldsymbol{I}_6$ (c.f. Proposition 4.2.2)
P as in (4.2.6)	$\begin{bmatrix} 26.0795\boldsymbol{I}_3 & 1.25\boldsymbol{I}_3 \\ 1.25\boldsymbol{I}_3 & 0.6818\boldsymbol{I}_3 \end{bmatrix}$
T_p as in (3.1.5)	0.8 seconds (8 steps)

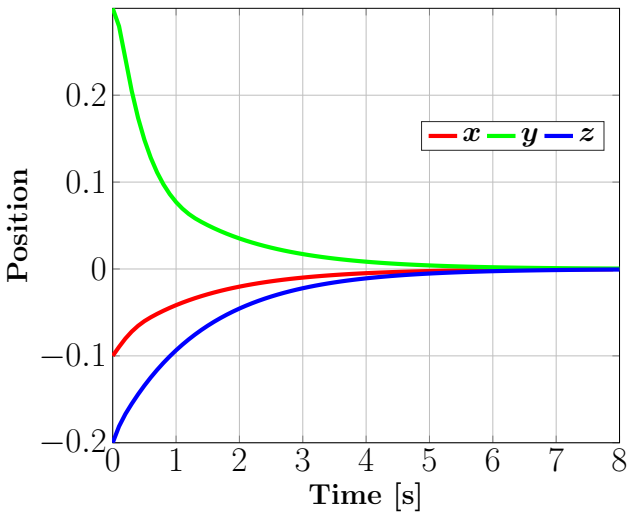
Table 4.2.3: Parameters of the NMPC deisng using terminal δ-invariant set.

the δ-invariant set $\mathcal{R}$ as defined in (4.2.91) in Figure 4.2.6a in which the set is split into three 2D sets $\mathcal{R}_q = \{|q| \leq X_q,\ |v_q| \leq V_q\}$ with $q \in \{x, y, z\}$. It can be observed that the size of the δ-invariant set is comparable to the size of the invariant set $\mathcal{S}(P_1, r_1)$ considered under

Scenario 1 of Section 4.2.1.3 (green ellipsoid given in Figure 4.2.6a) and hence, the prediction horizons of the two NMPC controllers are also similar with 8 steps for the δ-invariant approach as given in Table 4.2.3 w.r.t. 7 steps for the invariant set under Scenario 1 as given in Table 4.2.1. The important difference between them is the terminal box-type constraints resulted from the special construction of the δ-invariant set as in (4.2.91) in comparison with the quadratic inequality as in (4.2.79) of the standard invariant method which leads to a significant decreasing of the computational burden (average values of 46.5 ms for the δ-invariant approach as can be observed from Figure 4.2.7b w.r.t. 55 ms when using the invariant approach as given in Figure 4.2.4b). Beside that, the convergence speed is also improved. The δ-invariant approach enforces the states to converge within 3.2 seconds as can be seen from Figure 4.2.6 while it was 4.4 seconds when using the invariant set $\mathcal{S}(P_1, r_1)$ as being illustrated in Figure 4.2.3. However, bear in mind that when using a significantly larger invariant set and a clearly shorter prediction horizon length (e.g., $\mathcal{S}(P_2, r_2)$ in Figure 4.2.3a and the prediction horizon of 2 steps as considered under Scenario 2 of Section 4.2.1.3), we can further reduce the computing time down to 42 ms as shown in Figure 4.2.4b but the convergence speed is not better (i.e. up to 5 seconds, c.f. Figure 4.2.3). All the numerical results corresponding to the simulated NMPC controllers will be gathered into Table 4.4.1 for a complete comparison.



(a) Trajectories of (x, v_x), (y, v_y), (z, v_z) and the terminal δ-invariant set $\mathcal{R}$ in comparison with the invariant set $\mathcal{S}(P_1, r_1)$ from Figure 4.2.3.



(b) Convergences of x, y, z.

Figure 4.2.6: State convergences when using NMPC controllers with terminal δ-invariant set.

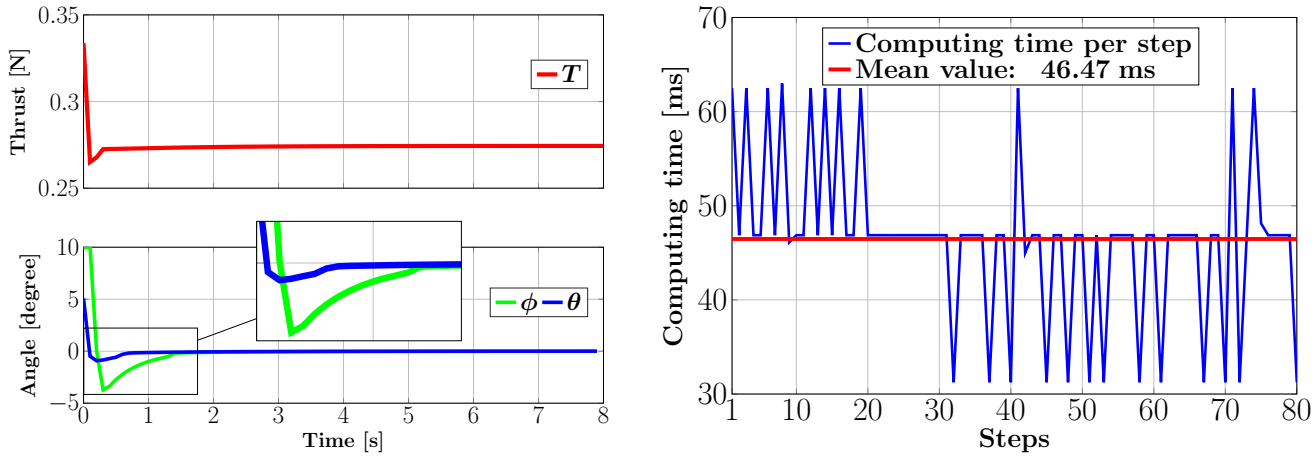
To conclude this contribution on the use of the δ-invariant set $\mathcal{R}$ (4.2.91) for stabilizing the NMPC controller (4.2.1)–(4.2.3), we would like to emphasize that using the proposed terminal box-type linear constraints provides many benefits in comparison with using a similar-size terminal invariant set. These can be enumerated as follows:

1. terminal constraints with clear interpretation as in (4.2.91) (w.r.t. the quadratic inequality of the standard ellipsoid invariant set (4.2.79)).

2. lower computing time (c.f. Figures 4.2.7b and 4.2.4b).

3. faster convergence speed (c.f. Figures 4.2.6 and 4.2.3).

We strongly prove the effectiveness of exploiting the FL controller (4.1.4) on stabilizing the NMPC designs (4.2.1)–(4.2.3) by using various terminal constraints which are as diverse in their



(a) Results of thrust T and pitch angle θ.

(b) Computing time per step.

Figure 4.2.7: Inputs and computing time when using NMPC controllers with terminal δ-invariant set.

formulations (e.g. quadratic and box-type linear inequalities) as they are in their sizes. All the designs show significant improvements when comparing with the classic quasi-infinite horizon NMPC design using a local linear controller [Chen and Allgöwer, 1998] in the sense that the design process and the tuning become easier and more efficient while the results including the convergence and the computing time are significantly better.
In the next section, we show another use of the FL controller (4.1.4), this time for stabilizing the NMPC design without terminal constraints. The design requires only a "long-enough" prediction horizon to guarantee its stability. The obtained results are also better than those of the existing approach using a standard linear controller [Kohler et al., 2018].

4.3 NMPC design without terminal constraint for position control

In the literature, it is well-known that the stability of an MPC controller can be achieved by either adding terminal stabilizing constraints (as presented in Section 4.2) or simply by enlarging the prediction horizon [Grüne, 2012]. However, how large the prediction horizon should be is always a difficult question as also discussed in Remarks 4.2.13 and 4.2.19. Therefore, in this section, we address an NMPC design for the multicopter's translation system (4.1.1) with its stability induced by using a "long-enough" prediction horizon length corresponding to a specific domain of attraction. The design, again, exploits the FL controller $u_{\text{FL}}(K\mathsf{p}, \psi)$ as in (4.1.4) in order to benefit from the simple linear dynamics (4.2.7). Especially, since the prediction horizon becomes the only important element, it will be easy to understand and put the design into practice if the implementation form is addressed from the beginning. Therefore, we consider the discrete domain for designing the NMPC scheme within this section. We employ the simplest NMPC implementation form where the dynamics (4.1.1) are discretized by using the same NMPC sampling time δ and the NMPC input is taken as the first step among the resulted optimal sequence. More precisely, the optimization problem at time step k is given by:

$$V_{N_p}(\mathsf{p}_k) = \min_{\bar{u}_k(\cdot)} J_{N_p}(\mathsf{p}_k, \bar{u}_k(\cdot)), \tag{4.3.1}$$

subject to

$$\bar{\mathsf{p}}_k(i+1) = \mathcal{F}\left(\bar{\mathsf{p}}_k(i), \bar{u}_k(i), \psi\right), \tag{4.3.2a}$$
$$\bar{u}_k(i) \in \mathcal{U}_\mathrm{p},\ i \in \{0, \ldots, N_p - 1\}, \tag{4.3.2b}$$
$$\bar{\mathsf{p}}_k(0) = \mathsf{p}_k, \tag{4.3.2c}$$

with $V_{N_p}(\mathsf{p}(k))$ the optimal value function, $(\bar{\mathsf{p}}_k(i), \bar{u}_k(i))$ the predicted state and input at step i employed within the problem at step k, $\mathcal{F}(\cdot)$ the discrete model of the system (4.1.1) explicitly defined later, $\mathcal{U}_\mathrm{p}$ the input constraint set as in (4.1.2). Furthermore, p_k denotes the state feedback at time k and ψ represents the yaw angle value assumed to be constant. The cost function $J_{N_p}(\mathsf{p}(k), \bar{u}_k(\cdot))$ is defined in terms of the stage cost $\ell(\cdot)$ given in (4.2.4), as follows:

$$J_{N_p}(\mathsf{p}(k), \bar{u}_k(\cdot)) = \sum_{i=0}^{N_p-1} \ell(\bar{\mathsf{p}}_k(i), \bar{u}_k(i)). \tag{4.3.3}$$

Then, the NMPC control action and the nominal closed-loop system at time k are given by:

$$u_\mathrm{MPC}(\mathsf{p}_k) = \bar{u}_k^*(0), \tag{4.3.4}$$
$$\mathsf{p}_{k+1} = \mathcal{F}(\mathsf{p}_k, u_\mathrm{MPC}(\mathsf{p}_k)) = \bar{\mathsf{p}}_k^*(1), \tag{4.3.5}$$

with $(\bar{\mathsf{p}}_k^*(i), \bar{u}_k^*(i))$ the optimal state and input at the predicted step i which is resulted from the optimization problem (4.3.1) at the time step k.
The discrete model employed in (4.3.2a) is obtained by discretizing the continuous system (4.1.1) with the Runge-Kutta fourth-order method [Hager, 2000]:

$$\mathsf{p}_{k+1} = \mathcal{F}\left(\mathsf{p}_k, u_k, \psi\right), \tag{4.3.6}$$

with (p_k, u_k) the state and input at time step k and ψ the yaw angle assumed to be constant. The function $\mathcal{F}(\cdot)$ is explicitly given by:

$$\mathcal{F}\left(\mathsf{p}_k, u_k, \psi\right) = \begin{bmatrix} \boldsymbol{I}_3 & \delta \boldsymbol{I}_3 \\ 0_{3\times 3} & \boldsymbol{I}_3 \end{bmatrix} \mathsf{p}_k + \frac{1}{m} \begin{bmatrix} \frac{\delta^2}{2} \boldsymbol{I}_3 \\ \delta \boldsymbol{I}_3 \end{bmatrix} \begin{bmatrix} T_k(\cos\phi_k \sin\theta_k \cos\psi + \sin\phi_k \sin\psi) \\ T_k(\cos\phi_k \sin\theta_k \sin\psi - \sin\phi_k \cos\psi) \\ -g + T_k \cos\phi_k \cos\theta_k \end{bmatrix}, \tag{4.3.7}$$

in which, the input vector $u_k \triangleq [T_k\ \phi_k\ \theta_k]^\top \in \mathbb{R}^3$ and δ is the discretization step which is also the NMPC sampling time. The discrete system (4.3.6) is linear in the state p_k since the original continuous system (4.1.1) already possesses this property.
The stability of system (4.3.5) is well studied and presented in [Grüne and Pannek, 2011, Kohler et al., 2018] and summarized hereinafter.

4.3.1 NMPC design with stability induced by a "long-enough" prediction horizon

Assumption 4.3.1. *Regarding the NMPC optimization problem* (4.3.1)*, there exist constants* $(\gamma, c) > 0$ *such that for any* $N_p \geq 2$ *and for all the initial state* p_0 *satisfying* $\|\mathsf{p}_0 - \mathsf{p}_e\|_Q^2 \leq c$*, we have:*

$$V_{N_p}(\mathsf{p}_0) \leq \gamma \|\mathsf{p}_0 - \mathsf{p}_e\|_Q^2, \tag{4.3.8}$$

with $V_{N_p}(\mathsf{p}_0)$ *the optimal value function as in* (4.3.1) *at step* 0*,* p_e *the desired equilibrium and* $Q \in \mathbb{R}^{6\times 6}$ *the weighting matrix as in* (4.2.4)*.*

Theorem 4.3.2. *Let Assumption 4.3.1 hold. Then, there exists an $N_0 \in \mathbb{N}$, such that for all the prediction horizon length $N_p \geq N_0$, the equilibrium p_e is uniformly exponentially stable under the nominal closed-loop dynamics* (4.3.5) *for any initial state p_0 satisfying $V_{N_p}(\mathsf{p}_0) \leq c\gamma$.* □

Proof. See Theorem 1 in [Kohler et al., 2018] and Theorem 3.6 in [Grüne, 2012]. At first, in [Kohler et al., 2018], the authors show that $V_{N_p}(\mathsf{p}_k) \leq c\gamma$ implies that $V_{N_p}(\mathsf{p}_k) \leq \gamma \|\mathsf{p}_k - \mathsf{p}_e\|_Q^2$ with a case dictinction based on whether $\|\mathsf{p}_k - \mathsf{p}_e\|_Q^2 \leq c$ or not [Kohler et al., 2018].
Secondly, whenever $V_{N_p}(\mathsf{p}_k) \leq \gamma \|\mathsf{p}_k - \mathsf{p}_e\|_Q^2$ holds, in [Grüne, 2012], the authors show that $V_{N_p}(\mathsf{p}_k)$ decreases for all $N_p \geq N_0$ with N_0 given by (see Variant 3 in [Grüne, 2012] for more details):

$$N_0 = 2 + \frac{\ln(\gamma - 1)}{\ln \gamma - \ln(\gamma - 1)}. \tag{4.3.9}$$

Lastly, the recursive feasibility and exponential stability are obtained with the initial condition $V_{N_p}(\mathsf{p}_0) \leq c\gamma$. □

In the literature, the standard approach for valdating Assumption 4.3.1 and Theorem 4.3.2 is to employ a linear controller and to estimate the parameters (γ, c) as in (4.3.8) within its associated invariant set (of ellipsoidal form as in [Chen and Allgöwer, 1998, Kohler et al., 2018] or polyhedral form as in [Cannon et al., 2003]). However, employing a linear controller for the nonlinear system (4.3.6) obviously restricts the corresponding invariant set, hence, arguably leading to an impractically large prediction horizon N_p from (4.3.9) of the NMPC controller (4.3.1)-(4.3.4).
Thus, it is worthwhile to ask whether, for the particular dynamics (as those shown in (4.3.6)), we may dispense with the linearized dynamics/linear controller construction and, instead, check Assumptions 4.3.1 by applying the FL controller $u_{\text{FL}}(\mu_\xi, \psi)$ defined in (4.1.4).
In order to do that, we emphasize that the feedback linearization property of the control law $u_{\text{FL}}(\mu_\xi, \psi)$ as in (4.1.4) remains validated for the discrete system (4.3.6) but requires the virtual input vector $\mu_\xi = [\mu_x \ \mu_y \ \mu_z]^\top$ as in (4.1.4) to admit the discrete PD formulation:

$$\mu_q(\mathsf{p}_k) = K_{1_q} q_k + K_{2_q} v_{q_k}, \ \forall q \in \{x, y, z\}, \tag{4.3.10}$$

in which, the control gains (K_{1_q}, K_{2_q}) is chosen differently in comparison with the continuous gains (K_{p_q}, K_{q_q}) (only required to be negative) as in (4.2.8):

$$-\frac{2}{\delta} < K_{2_q} < \frac{\delta}{2} K_{1_q} < 0. \tag{4.3.11}$$

The condition (4.3.11) is to guarantee the stability of the following linear dynamics obtained from introducing the FL controller $u_{\text{FL}}(\mu_\xi(\mathsf{p}_k), \psi)$ to the nonlinear system (4.3.6):

$$\mathsf{p}_{k+1} = \underbrace{\begin{bmatrix} \boldsymbol{I}_3 + \frac{\delta^2}{2} K_1 & \delta \boldsymbol{I}_3 + \frac{\delta^2}{2} K_2 \\ \delta K_1 & \boldsymbol{I}_3 + \delta K_2 \end{bmatrix}}_{A_{K_d}} \mathsf{p}_k, \tag{4.3.12}$$

in which, the gain matrix $K_d \in \mathbb{R}^{3\times 6}$ gathers all the control gains (K_{1_q}, K_{2_q}) as in (4.3.11):

$$K_d = \begin{bmatrix} K_1 & K_2 \end{bmatrix}, \tag{4.3.13}$$

with $K_1 = \text{diag}\{K_{1_x}, K_{1_y}, K_{1_z}\}$ and $K_2 = \text{diag}\{K_{2_x}, K_{2_y}, K_{2_z}\}$.
Next, we will address the discrete formulation of the corresponding ellipsoid invariant set (i.e.

the continuous form is given in Proposition 4.2.4) and the convergence rate of the state within the set which will be useful to check Assumption 4.3.1.
By choosing a symmetric positive definite matrix $M_d \in \mathbb{R}^{6\times 6}$, we obtain a symmetric positive definite matrix $P_d \in \mathbb{R}^{6\times 6}$ as the unique solution of the following discrete Lyapunov equation:

$$A_{K_d}^\top P_d A_{K_d} = P_d - M_d. \tag{4.3.14}$$

with the discrete stable matrix A_{cl} from (4.3.12). Note that, the symmetric matrix M_d employed for the discrete Lyapunov equation (4.3.14) does not require to be chosen as in (4.2.10) but only to be positive definite which is due to the fact that the terminal cost as in (4.2.5) is not used in the NMPC design (4.3.1)–(4.3.4). The matrix P_d obtained from (4.3.14) allows us to construct an ellipsoid invariant set for the discrete system (4.3.6) as follows:

$$\mathcal{S}(P_d, r_d) = \{\mathsf{p}_k \in \mathbb{R}^6 |\ \mathsf{p}_k^\top P_d\, \mathsf{p}_k \leq \min(\mathrm{eig}(P_d)) r_d^2\}, \tag{4.3.15}$$

with $r_d \in \mathbb{R}_+$ chosen similarly to r in (4.2.19) but with the new discrete control gains (K_{1_q}, K_{2_q}) as in (4.3.11):

$$r_d = \min_{q\in\{x,y,z\}} \left\{ \frac{U_q}{\sqrt{K_{1_q}^2 + K_{2_q}^2}} \right\}, \tag{4.3.16}$$

with (U_x, U_y, U_z) the positive constants as in (4.1.6).

Proposition 4.3.3 ([Nguyen et al., 2020c]). *The set $\mathcal{S}(P_d, r_d)$ defined in (4.3.15) is input constraint admissible and positive invariant for the system (4.3.6) under the FL controller $u_{FL}(K_d\mathsf{p}_k, \psi)$ as in (4.1.4) with K_d from (4.3.13). Furthermore, within the set, the state convergence speed is bounded by a scalar ρ as follows:*

$$\|\mathsf{p}_{k+1}\|_{P_d}^2 \leq \underbrace{\left(1 - \frac{\min(\mathrm{eig}(M))}{\max(\mathrm{eig}(P_d))}\right)}_{\rho} \|\mathsf{p}_k\|_{P_d}^2, \tag{4.3.17}$$

with M and P_d symmetric positive definite matrices as in (4.3.14).

Proof. The input constraint admissible property of the set $\mathcal{S}(P_d, r_d)$ as in (4.3.15) is constructed identically to the one of Proposition 4.2.4. Then, within the set, the FL controller $u_{\mathrm{FL}}(K_d\mathsf{p}_k, \psi)$, given as in (4.1.4) with K_d from (4.3.13) provides:

$$\|\mathsf{p}_{k+1}\|_{P_d}^2 = \mathsf{p}^\top A_{K_d}^\top P_d A_{K_d} \mathsf{p} = \|\mathsf{p}_k\|_{P_d}^2 - \|\mathsf{p}_k\|_M^2, \tag{4.3.18}$$

with M and P_d symmetric positive definite matrices as in (4.3.14). This firstly proves the invariant property of the set $\mathcal{S}(P_d, r_d)$ from (4.3.15) and further leads to the convergence rate as in (4.3.17) due to:

$$\|\mathsf{p}_k\|_M^2 \geq \min(\mathrm{eig}(M))\|\mathsf{p}_k\|^2 \geq \frac{\min(\mathrm{eig}(M))}{\max(\mathrm{eig}(P_d))}\|\mathsf{p}_k\|_{P_d}^2. \tag{4.3.19}$$

This also completes the proof. □

Hereinafter, we will employ the results of Proposition 4.3.3 to validate the Assumption 4.3.1 and Theorem 4.3.2. We firstly define the positive constants c, γ from (4.3.8) as follows:

$$c = r_d^2 \min(\mathrm{eig}(Q)) \frac{\min(\mathrm{eig}(P_d))}{\max(\mathrm{eig}(P_d))},\ \gamma = \frac{\max(\mathrm{eig}(Q^*))}{\min(\mathrm{eig}(Q))} \frac{\max(\mathrm{eig}(P_d))}{\min(\mathrm{eig}(P_d))(1-\rho)}, \tag{4.3.20}$$

with Q the weighting matrix as in (4.2.4), r_d from (4.3.16), P_d obtained from solving the Lyapunov equation (4.3.14) and ρ as defined in (4.3.17). The matrix Q^* is given as follows:

$$Q^* = Q + \mathsf{L} \max(\text{eig}(R)) K_d^\top K_d, \tag{4.3.21}$$

with R the weighting matrix as in (4.2.4), L a positive constant as in (4.1.10) and K_d the control gain matrix from (4.3.13).

Proposition 4.3.4 ([Nguyen et al., 2020c]). *Assumption 4.3.1 is satisfied with (c, γ) as in* (4.3.20). *Furthermore, the closed-loop dynamics* (4.3.5) *(controlled by the NMPC controller* (4.3.1)–(4.3.4)*) are uniformly exponentially stable for all the initial states p_0 satisfying $V_{N_p}(\mathsf{p}_0) \leq c\gamma$ with $V_{N_p}(\mathsf{p}_0)$ the optimal value function from* (4.3.1). □

Proof. The proof is detailed in Appendix G. □

By Proposition 4.3.4, the stability of the NMPC controller (4.3.1)–(4.3.4) is established by using the parameters (γ, c) as defined in (4.3.20) which are exploited from the FL controller $u_{\text{FL}}(K_d\mathsf{p}, \psi)$ as in (4.1.4), (4.3.13). In the following, the design procedure is summarized.

Procedure 4.3.5 (NMPC design without terminal stabilizing constraints)**.**

1. *Choose the symmetric matrices $Q \in \mathbb{R}^{6\times 6}$ (positive definite) and $R \in \mathbb{R}^{3\times 3}$ (positive semi-definite) to formulate the stage cost as in* (4.2.4).
2. *Choose the control gains (K_{1_q}, K_{2_q}) $(q \in \{x, y, z\})$ as in* (4.3.13).
3. *Choose the symmetric positive definite matrix M in* (4.3.14) *and solve the Lyapunov equation* (4.3.14) *for P_d.*
4. *Find ρ as in* (4.3.17)*, c and γ as in* (4.3.20) *in order to find N_0 given in* (4.3.9).
5. *Define the prediction horizon $N_p \geq N_0$ as in* (4.3.9).

4.3.2 Tuning the prediction horizon

It is well-known in the literature that an NMPC controller without terminal stabilizing constraints as in (4.3.1)–(4.3.4) requires a sophisticated tuning procedure in order to obtain the reasonable values of the required minimum prediction horizon N_0 and also of the region of attraction which guarantees the stability [Kohler et al., 2018, Grüne, 2012]. However, to the best of our knowledge, the tuning problems of the NMPC without terminal stabilizing constraints have been underestimated in various relating works [Limón et al., 2006, Grüne, 2012, Reble and Allgöwer, 2012, Kohler et al., 2018]. That is to say, people concentrate mostly on the stability proofs of their NMPC designs (e.g. as our contribution in Proposition 4.3.4), then, provide one illustrative example with specific parameters [Kohler et al., 2018, Boccia et al., 2014]. Theses examples actually aim to show how the results are obtained (e.g. in order for the readers to validate again the calculation process by themselves) but do not give the insight into the actual tuning process. For our particular NMPC design (4.3.1)-(4.3.4), the most influential parameters are the control gains K_{1_q}, K_{2_q} with $q \in \{x, y, z\}$ satisfying the condition (4.3.11), the symmetric positive definite matrices M from (4.3.14) and Q, R from (4.2.4) which are not easy to tune due to a large amount of decision variables. We will show that an arbitrary choice of these parameters can easily lead to a dramatically large and hence, impractical prediction horizon length.

Example 4.3.6 (Effects of tuning parameters on prediction horizon length)**.** *The values of the required prediction horizon length N_0 as in* (4.3.9) *and the corresponding parameter c as in* (4.3.20) *w.r.t. to different tuning scenarios are given in Table 4.3.1. It can be observed that appropriate changes in the tuning parameters allow to reduce the required prediction horizon N_0 and to increase the value of c which represents the inner part of the domain of attraction, i.e., $\|\mathsf{p}\|_Q^2 \leq c$ as in* (4.3.8)*, hence, also enlarging the domain. The tuning steps are listed following Procedure 4.3.4. The values of (U_x, U_y, U_z) as in* (4.3.16) *are taken from Table 2.5.1.*

Description	Q	R	K_1	K_2	M	N_0	c
First choice	$\begin{bmatrix} \boldsymbol{I}_3 & 0_{3\times3} \\ 0_{3\times3} & 0.1\boldsymbol{I}_3 \end{bmatrix}$	$0.1\boldsymbol{I}_3$	$-\boldsymbol{I}_3$	$-\boldsymbol{I}_3$	$\boldsymbol{I}_6$	2853	0.01
Tuning R	$\begin{bmatrix} \boldsymbol{I}_3 & 0_{3\times3} \\ 0_{3\times3} & 0.1\boldsymbol{I}_3 \end{bmatrix}$	$0.01\boldsymbol{I}_3$	$-\boldsymbol{I}_3$	$-\boldsymbol{I}_3$	$\boldsymbol{I}_6$	2762	0.01
Tuning Q	$\boldsymbol{I}_6$	$0.01\boldsymbol{I}_3$	$-\boldsymbol{I}_3$	$-\boldsymbol{I}_3$	$\boldsymbol{I}_6$	173	0.104
Tuning K_1	$\boldsymbol{I}_6$	$0.01\boldsymbol{I}_3$	$-1.2\boldsymbol{I}_3$	$-\boldsymbol{I}_3$	$\boldsymbol{I}_6$	156	0.09
Tuning K_2	$\boldsymbol{I}_6$	$0.01\boldsymbol{I}_3$	$-1.2\boldsymbol{I}_3$	$-0.9\boldsymbol{I}_3$	$\boldsymbol{I}_6$	152	0.106
Tuning M	$\boldsymbol{I}_6$	$0.01\boldsymbol{I}_3$	$-1.2\boldsymbol{I}_3$	$-0.9\boldsymbol{I}_3$	$\begin{bmatrix} \boldsymbol{I}_3 & 0.1\boldsymbol{I}_3 \\ 0.1\boldsymbol{I}_3 & 1.3\boldsymbol{I}_3 \end{bmatrix}$	150	0.118

Table 4.3.1: Prediction horizon length w.r.t different tuning parameters [Nguyen et al., 2020c].

By Example 4.3.6, we have shown that appropriately tuning the variables can increase the performance of the NMPC controller (4.3.1)–(4.3.4), however, it also raises a question on:

How to obtain the appropriate parameters for the NMPC design (4.3.1)–(4.3.4) ?

In order to answer this question, we consider the important ingredient γ as calculated in (4.3.20) which affects the required prediction horizon N_0 as in (4.3.9). By introducing ρ from (4.3.17) into γ as in (4.3.20), we can express γ as the multiplication of γ_1 and γ_2 as follows:

$$\gamma = \underbrace{\frac{\max(\text{eig}(Q^*))}{\min(\text{eig}(Q))}}_{\gamma_1} \underbrace{\frac{\left(\max(\text{eig}(P_d))\right)^2}{\min(\text{eig}(P_d))\min(\text{eig}(M))}}_{\gamma_2}, \tag{4.3.22}$$

with Q as in (4.2.4), P_d, M as employed in the Lyapunov equation (4.3.14) and Q^* as in (4.3.21), all symmetric positive definite matrices. At first, from the formulation of N_0 given in (4.3.9), decreasing the value of γ as in (4.3.22) also reduces the prediction horizon N_0. Therefore, it is in our interest to minimize the values of γ_1 and γ_2 as defined in (4.3.22). In order to reduce the decision variables, we follow some reasonable assumptions:

- We give equal importance to the motions along the three axes (as similar to (4.2.49)–(4.2.50)), i.e.:

$$Q = \text{diag}\{q_1, q_1, q_1, q_2, q_2, q_2\}, \tag{4.3.23}$$
$$K_{1_x} = K_{1_y} = K_{1_z} = k_1, \tag{4.3.24}$$
$$K_{2_x} = K_{2_y} = K_{2_z} = k_2. \tag{4.3.25}$$

in which, q_1, q_2 are positive scalars and $k_1, k_2 < 0$ have to satisfy (4.3.11). As a result, the matrix A_{K_d} in (4.3.12) becomes:

$$A_{K_d} = \begin{bmatrix} \left(1 + \frac{\delta^2}{2}k_1\right)\boldsymbol{I}_3 & \delta\left(1 + \frac{\delta}{2}k_2\right)\boldsymbol{I}_3 \\ \delta k_1 \boldsymbol{I}_3 & (1 + \delta k_2)\boldsymbol{I}_3 \end{bmatrix}, \tag{4.3.26}$$

- The symmetric matrix $M \in \mathbb{R}^{6\times 6}$ in (4.3.14) is parametrized as follows:

$$M = \begin{bmatrix} m_1\boldsymbol{I}_3 & m_3\boldsymbol{I}_3 \\ m_3\boldsymbol{I}_3 & m_2\boldsymbol{I}_3 \end{bmatrix}, \tag{4.3.27}$$

with $m_1, m_2, m_3 \in \mathbb{R}$ satisfying the following conditions:

$$m_1 m_2 > m_3^2. \tag{4.3.28}$$

in which the role of (4.3.28) is to guarantee the positive definite property of M. Note that, with $m_1 = 1$, (4.3.28) is simplified to $m_2 > m_3^2$.

Remark 4.3.7. We fix $m_1 = 1$ as employed in (4.3.27) since any scaling on M results on the same scaling on P due to the Lyapunov equation (4.3.14). Then, all the eigenvalues of M and P are scaled similarly, hence, γ_2 in (4.3.22) does not change. Therefore, fixing $m_1 = 1$ does not cause any loss of generality within the analysis (but reduces the numerical issues). □

4.3.2.1 Tuning γ_1 as in (4.3.22)

By introducing the parametrizations of Q and (K_{1_q}, K_{2_q}) (with $q \in \{x, y, z\}$) (4.3.23)–(4.3.25) to Q^* as in (4.3.21), γ_1 as defined in (4.3.22) is explicitly given by:

$$\gamma_1 = \frac{\max(\text{eig}(Q^*))}{\min(\text{eig}(Q))} = \frac{q_1 + q_2 + \tilde{r}(k_1^2 + k_2^2) + \sqrt{\left(q_1 - q_2 + \tilde{r}(k_1^2 - k_2^2)\right)^2 + 4\tilde{r}^2 k_1^2 k_2^2}}{2\min(q_1, q_2)}, \tag{4.3.29}$$

in which, $\tilde{r}$ is directly proportional with the maximum eigenvalue of the positive semi-definite matrix R:

$$\tilde{r} = \mathsf{L}\max(\text{eig}(R)), \tag{4.3.30}$$

with L a positive constant as in (4.1.10). From (4.3.29), we obtain that:

$$\frac{q_1 + q_2 + \tilde{r}(k_1 + k_2)^2}{2\min(q_1, q_2)} \leq \gamma_1 \leq \frac{q_1 + q_2 + \tilde{r}(k_1^2 + k_2^2) + \sqrt{2\left(q_1 - q_2\right)^2 + 2\tilde{r}^2(k_1^4 + k_2^4)}}{2\min(q_1, q_2)}, \tag{4.3.31}$$

in which, the first inequality is due to $\left(q_1 - q_2 + \tilde{r}(k_1^2 - k_2^2)\right)^2 \geq 0$ and the latter one is by using $\left(q_1 - q_2 + \tilde{r}(k_1^2 - k_2^2)\right)^2 \leq 2\left(q_1 - q_2\right)^2 + 2\tilde{r}^2(k_1^2 - k_2^2)^2$. From (4.3.31), it can be observed that reducing the values of $|q_1 - q_2|$, $\max(\text{eig}(R))$, k_1^2 and k_2^2 probably provide a smaller value of γ_1 (we use "probably" since the reductions of the aforementioned parameters actually make both the upper and lower bounds as in (4.3.31) smaller). Especially, in case of $q_1 = q_2 = a\tilde{r}$ with a positive scalar a, γ_1 from (4.3.29) becomes:

$$\gamma_1 = 1 + \frac{k_1^2 + k_2^2}{a}, \tag{4.3.32}$$

which actually allows us to obtain a specific value of γ_1 by tuning only two weighting matrices Q and R as employed in (4.2.4) regardless the predefined values of (k_1, k_2).
Therefore, for tuning γ_1 as in (4.3.29), we propose several general directions:

- decrease the ratio $\max(\text{eig}(Q))/\min(\text{eig}(Q))$ as much as possible.
- decrease the value of $\max(\text{eig}(R))$ based on the employed values of (k_1, k_2) but bear in mind that a small value of R causes large input consumption.
- decrease the values of k_1^2 and k_2^2. However, this is not encouraged since it can cause an unexpected increase in the value of γ_2 as in (4.3.22) and finally result in a larger prediction horizon N_0.

Remark 4.3.8. Formulation (4.3.32) explains the effects of the first two tuning steps given in Table 4.3.1: i) reducing $\max(\text{eig}(R))$ and ii) eliminating the term $|q_1 - q_2|$. However, it does not explain for the rest of Table 4.3.1 which requires us to analyze the parameter γ_2 as detailed in the following. □

4.3.2.2 Tuning γ_2 as in (4.3.22)

As similar to the analysis on how to tune γ_1 as in (4.3.29), the first step is to find the explicit formulation of γ_2 (defined in (4.3.22)) in terms of the tuning variables (4.3.23)–(4.2.13). The calculation is identically to the one proving the semi-global stability property as in Proposition 4.2.7 and is too cumbersome to present here. Therefore, we give the proof in Appendix H and present only the steps to follow hereinafter:

1. Solve the discrete Lyapunov equation (4.3.14) with A_{K_d} as in (4.3.26) and M as in (4.2.13), very similar to the approach detailed in Proposition 4.2.8. The matrix solution P_d is explicitly given in (H.0.1).
2. Obtain the explicit formulations of the eigenvalues of P_d and M (given in (H.0.4)–(H.0.6)) similarly to the results of Corollary 4.2.9.
3. Introduce the eigenvalues of P_d and M as given in (H.0.4)–(H.0.6) to γ_2 defined in (4.3.22).

Then, we obtain the explicit formulation of γ_2 (4.3.22) in terms of the parameters (k_1, k_2, m_2, m_3) (4.3.24)–(4.2.13) which is defined as follows:

$$\gamma_2 \triangleq \gamma_2(k_1, k_2, m_2, m_3), \tag{4.3.33}$$

with the function $\gamma_2(\cdot)$ illustrated in Figure 4.3.1 for some specific values of (k_1, k_2).

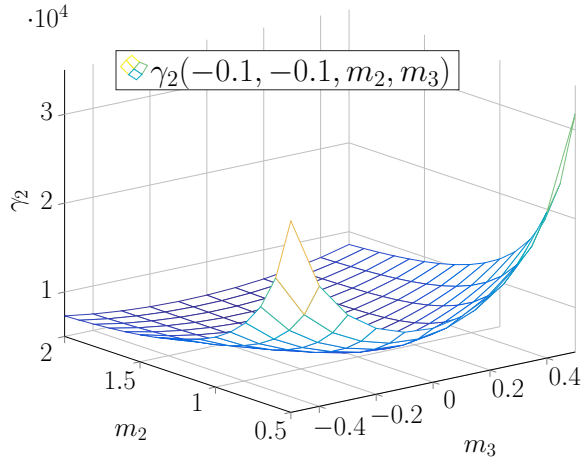
It will be in our interest to analyze and to find the (local) minimums of the function $\gamma_2(\cdot)$ defined in (4.3.33). Since the function is strongly nonlinear, non-convex and contains up to four variables, we have to divide the task into two steps:

1. Find the optimal values of (m_2, m_3) which provide the local minimum value of γ_2 corresponding to a specific values of (k_1, k_2):

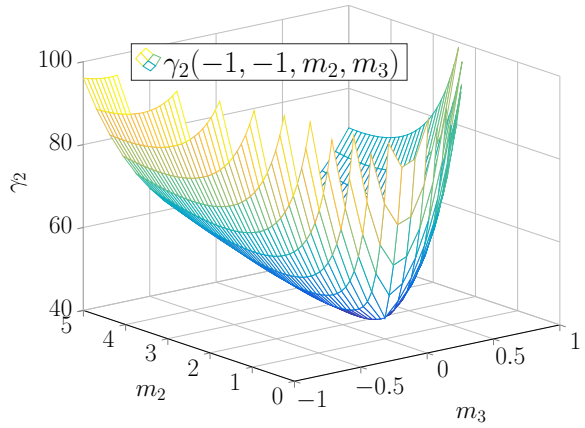
$$\begin{aligned} \gamma_2^*(k_1, k_2) = \min_{m_2, m_3} \ & \gamma_2(k_1, k_2, m_2, m_3). \\ \text{subject to } & m_2 > m_3^2 \text{ as required in (4.3.28)}, \\ & (m_2, m_3) \in [m_{2_{\min}}, m_{2_{\max}}] \times [-m_{3_{\max}}, m_{3_{\max}}], \end{aligned} \tag{4.3.34}$$

   with $(m_{2_{\min}}, m_{2_{\max}}, m_{3_{\max}})$ positive scalars specifying the ranges of m_2 and m_3 as in (4.2.13), defined by user (c.f. Figure 4.3.1).

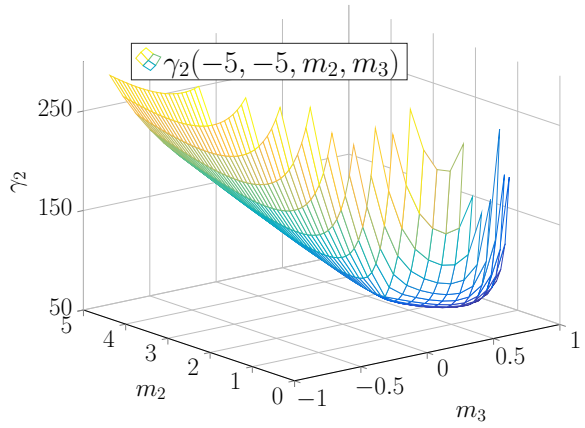
2. Apply Step 1 for different choices of (k_1, k_2), then, compare the obtained minimum values γ_2^* as in (4.3.34) in order to provide the complete analysis.



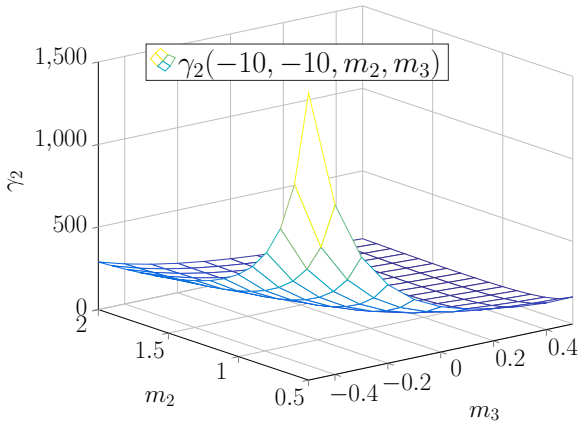
(a) $(k_1, k_2) = (-0.1, -0.1)$



(b) $(k_1, k_2) = (-1, -1)$

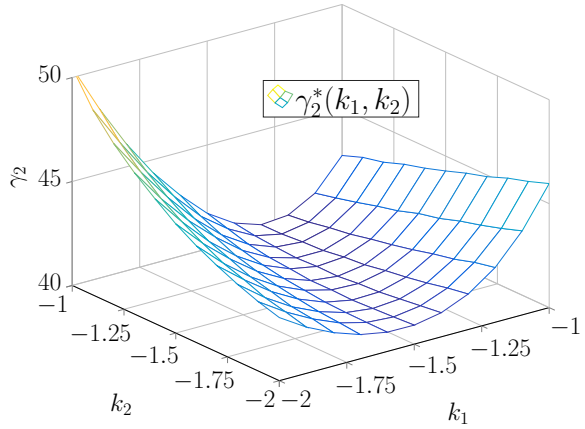


(c) $(k_1, k_2) = (-5, -5)$

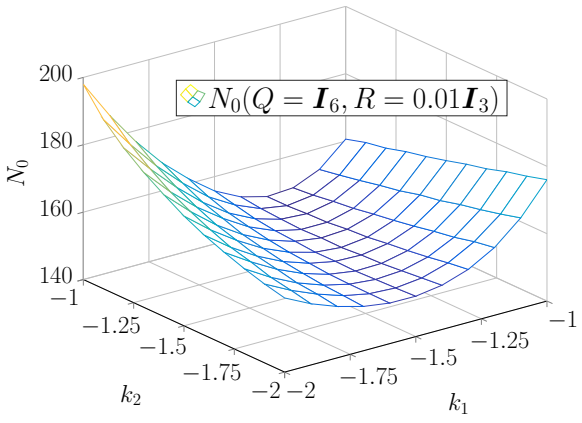


(d) $(k_1, k_2) = (-10, -10)$

Figure 4.3.1: Illustration of $\gamma_2(k_1, k_2, m_2, m_3)$ as in (4.3.33) with different values of (k_1, k_2).



(a) Illustration of the optimal function $\gamma_2^*(k_1, k_2)$ defined in (4.3.34)



(b) Minimum values of N_0 obtained at the optimal choices of (m_2, m_3)

Figure 4.3.2: Illustration of the analysis on N_0 [Nguyen et al., 2020c]

Remark 4.3.9. The solution of the optimization problem in (4.3.34) is obtained by checking a mesh grid of the variables (m_2, m_3) within the specific range $[m_{2_{\min}}, m_{2_{\max}}] \times [-m_{3_{\max}}, m_{3_{\max}}]$, hence, the solution's accuracy depends on the resolution of the mesh grid. However, the accuracy problem is not critical due to the fact that we always have to choose the prediction horizon

$N_p \geq N_0$ and $N_p \in \mathbb{N}$ (e.g., it does not matter if we obtain $N_0 = 149.4$ instead of the precise value assumed to be 149.36 since the minimum prediction horizon needs to be an integer, which leads in both case to $N_p = 150$). Furthermore, using the mesh grid method provides very fast computing times. E.g., it takes only 0.06 seconds to construct the whole data points for Figure 4.3.2 which contains 51×41 points of (m_2, m_3) and 11×11 points of (k_1, k_2). □

k_1 (4.3.24)	k_2 (4.3.25)	m_3 (4.3.27)	m_2 (4.3.27)	N_0 (4.3.9)	c (4.3.20)
-1.3	-1.1	0.2	1.6	150	0.089
-1.3	-1	0.1	1.3	150	0.093
-1.2	-0.9	0.1	1.3	150	0.118
-1.2	-0.8	0	1	150	0.125
-0.1	-0.1	**Method in [Kohler et al., 2018]**		10^6	10^{-4}
-1	-1	Using linear controller $u = K_d\mathsf{p} + u_e$		5124	10^{-7}
-2	-2	and the linearized dynamics		1112	10^{-7}

Table 4.3.2: Optimal values of (k_1, k_2, m_2, m_3) which provide the smallest N_0 in comparison with the method in [Kohler et al., 2018] (using $Q = \boldsymbol{I}_6$ and $R = 0.01\boldsymbol{I}_3$ as in (4.2.4)).

We enumerate in Table 4.3.2 all the scenarios in which, we obtain the smallest value of the required prediction horizon $N_0 = 150$ by using the weighting matrices $Q = \boldsymbol{I}_6$ and $R = 0.01\boldsymbol{I}_3$. Table 4.3.2 also shows the corresponding values of c from (4.3.20) which helps identifying all the feasible initial states p_0, i.e., $V_{N_p}(\mathsf{p}_0) \leq c\gamma$ as stated in Theorem 4.3.2. All the choices of (k_1, k_2, m_2, m_3) gathered in Table 4.3.2 require the minimum prediction horizon of 150 but they provide a large range of c from 0.089 to 0.125. Obviously, with the same prediction horizon of 150 (i.e., similar γ), the larger value of c results in larger domain of attraction. Thus, the control gains $(k_1, k_2) = (-1.2, -0.8)$ with $N_0 = 150$ and $c = 0.125$ obtained with $M = \boldsymbol{I}_6$ appears to be the best choice in our analysis.

For comparison, we use the method proposed in [Kohler et al., 2018] which employs a linear controller $u = K_d\mathsf{p} + u_e$ with K_d as in (4.2.50) and u_e as in (4.1.3) as well as the linearized model of the dynamics (4.3.6). The results are given in the last three lines in which both the value of c and the prediction horizon N_0 are much more conservative than the results of our proposed method. This is due to the restriction of using the linear controller for the strongly nonlinear system (4.3.6). This strongly confirms the effectiveness of our NMPC design approach using the local FL controller $u_{\text{FL}}(K_d\mathsf{p}, \psi)$ as in (4.1.4).

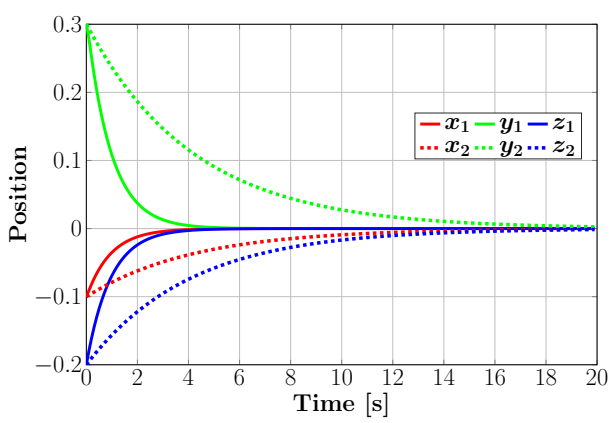
Furthermore, we also notice that even the shortest prediction horizon in our analysis $N_0 = 150$ steps is still extremely large for real implementation. However, bear in mind that the obtained results (c.f. Table 4.3.2) are still more promising than employing the linear controller as considered in [Kohler et al., 2018]. This also indicates a big gap still existing between the theory on NMPC design and their practical formulations since through various simulations and experimental tests, the NMPC controller (4.3.4)–(4.3.4) requires a prediction horizon of only 10 steps to stabilize the system (4.3.6). This is due to the fact that the NMPC controller can fully exploit the inputs of the system while a standard controller (e.g. FL controller $u_{\text{FL}}(K_d\mathsf{p}, \psi)$ as in (4.1.4) or the linear controller as employed in [Kohler et al., 2018]) enforces the inputs to follow their explicit formulations, hence, the convergence speed of the standard controller (e.g. ρ as in (4.3.17)) can not be fast enough to obtain this short prediction horizon of 10 steps.

To conclude, we have completely exploited the usage of the FL controller $u_{\text{FL}}(K_d\mathsf{p}, \psi)$ for designing the NMPC controller (4.3.4)–(4.3.4) through the stability proof as in Proposition 4.3.4

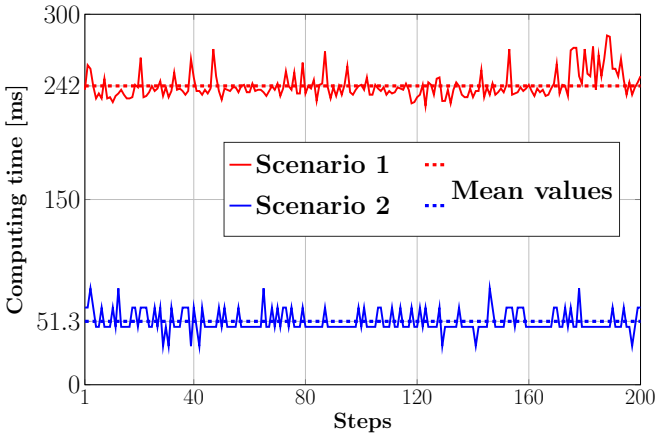
and also the tuning procedure as detailed in Section 4.3.2 and positively believe that our complete analysis can be generalized and similarly applied for various nonlinear controllers (e.g., passivity-based control [Ha et al., 2014], $\mathcal{H}_\infty$ control [Kerma et al., 2012], etc) which may result in more realistic prediction horizons.

4.3.3 Simulation validation of NMPC controller without terminal stabilizing constraints

This section provides the simulation results for stabilizing the model of a Crazy flie 2.0 nano quadcopter platform characterized by the parameters given in (4.2.68) using the NMPC design without terminal constraints (4.3.1)–(4.3.4). The simulation employs the same parameters as considered in Sections 4.2.1.3 and 4.2.2.4, i.e., the NMPC sampling time and model discretization step $\delta = 0.1$ seconds and the initial state $\mathsf{p}_0 = [-0.1\ 0.3\ -0.2\ 0\ 0\ 0]^\top$. We consider two simulation scenarios:
Scenario 1: employing the prediction horizon of $N_p = 150$ steps which guarantees the closed-loop stability according to Table 4.3.1.
Scenario 2: employing the prediction horizon of $N_p = 10$ steps.
Both scenarios use the weighting matrices $Q = \boldsymbol{I}_6$ and $R = 0.01\boldsymbol{I}_3$ as considered in Table 4.3.1. We provide the state convergence results in Figure 4.3.3a in which, we can see that the NMPC



(a) Position convergences under two scenarios.



(b) Computing time per step.

Figure 4.3.3: Position results and computing time when using NMPC controllers without terminal stabilizing constraints.

controller under Scenario 1 (using the 150-step prediction horizon) obtains the transition time of 3.2 seconds while employing the prediction horizon of only 10 steps makes the states converging in longer time, i.e., 14 seconds. It is also trivial that the computation burden under Scenario 1 is much heavier than the one under Scenario 2 as can be seen from Figure 4.3.3b. The average computing time of the NMPC controller with the 150-step prediction horizon is 242 milliseconds while it is only 51.3 second for the second scenario.
In the next section, all the simulation results of the presented NMPC designs in Sections 4.2–4.3 will be gathered together in order to conduct a thorough and complete comparison between them.

4.4 Comparisons between different NMPC designs

This section gathers all results of the numerical simulations carried out for the different NMPC designs introduced in this Chapter. The analysis is given in Table 4.4.1 which includes:

1. the proposed NMPC design with terminal invariant set constructed under the FL controller (4.1.4). The design achieves the semi-global stability property. Two scenarios are considered:

   i) the initial state does not stay within the terminal set, hence, the prediction horizon length needs to be increased in order to guarantee the feasibility of the NMPC problem (Scenario 1 in Section 4.2.1.3);

   ii) the terminal set is enlarged until covering the initial state. The shortest prediction horizon of 2 steps is employed (Scenario 2 in Section 4.2.1.3).

2. the quasi-infinite horizon NMPC (denoted by qMPC in the analysis) design [Chen and Allgöwer, 1998] (Scenario 3 in Section 4.2.1.3).

3. the proposed NMPC design with terminal δ-invariant set under the the FL controller (4.1.4) (Section 4.2.2.4).

4. the NMPC design without terminal stabilizing constraints which is constructed by using the FL controller (4.1.4) as the feasible guess and is able to guarantee stability for a prediction horizon of 150 steps (Scenario 1 in Section 4.3.3).

5. the NMPC design without terminal stabilizing constraints using the prediction horizon of 10 steps which, in simulation proves capable of ensuring a bounded behavior but lacks theoretical guarantees – it has no stability proof (Scenario 2 in Section 4.3.3).

6. the NMPC design without terminal stabilizing constraints but with its stability guaranteed by using the method introduced in [Kohler et al., 2018] with the prediction horizon of 1152 steps (c.f. Table 4.3.2). No simulation is conducted for this scenario due to its impractically long prediction horizon.

The data given in Table 4.4.1 clearly shows that exploiting the FL controller (4.1.4) to design an NMPC controller provides better results than using a standard linear controller (as used in qMPC design [Chen and Allgöwer, 1998] and in [Kohler et al., 2018]) for both cases of using and not using the terminal stabilizing constraints. At first, regarding the first four scenarios, all making use of the terminal stabilizing constraints, the qMPC design provides the smallest terminal region (c.f. Ω_α in Figure 4.2.3). Hence, it requires the longest prediction horizon of 12 steps. These conservative elements lead to a significantly larger computing time (with the mean value of 77 milliseconds and the maximum value reaching 109 milliseconds) in comparison with the three other cases using the proposed methods. Next, the first two scenarios (i.e., using the proposed terminal invariant set $\mathcal{S}(P, d)$ as in (4.2.79)) indicates the ease to tune the corresponding NMPC design while still guaranteeing a low computation burden and good control performance. We can either use the "medium-size" terminal set $\mathcal{S}(P_1, d_1)$ with 7-step prediction horizon or employ the enlarging approach detailed in Section 4.1 in order to obtain the terminal set $\mathcal{S}(P_2, d_2)$ ultimately containing the initial state (c.f. Figure 4.2.3a) and hence, to possibly employ a short prediction horizon of 2 steps which still guarantees the closed-loop stability. The second design (i.e., using the large-size terminal region $\mathcal{S}(P_2, d_2)$) requires the least computing time with only 42 milliseconds per step on average but provides inferior control performance

with its convergence time of 5.2 seconds - the second slowest in the analysis. However, if the users want to increase the performance, they can easily reduce the size of the terminal invariant set and the convergence time can be reduced accordingly, e.g., down to 4.4 seconds when using the smaller invariant set $\mathcal{S}(P_1, d_1)$.
Furthermore, an interesting point is that we are able not only to increase the control performance but also to reduce the computing time by using a relaxed invariant set, called δ-invariant set $\mathcal{R}$ as in (4.2.91) with its linear construction as the terminal region of the NMPC design with the closed-loop stability guaranteed by Proposition 4.2.21. The data given in the fourth scenarios shows that using these linear box-type constraints (4.2.91) helps reducing the computing time down to 46.5 milliseconds per step on average (i.e., the second lowest among the considered scenarios) while its "medium" size (c.f. the set $\mathcal{R}$ in Figure 4.2.6a in comparison with the set $\mathcal{P}(P_1, r_1)$ in Figure 4.2.3a) provides a good convergence speed of 3.2 seconds.
Next, regarding the last three scenarios in which we consider the NMPC design without terminal stabilizing constraints (denoted by uMPC), the proposed calculation process based on the FL controller (4.1.4) significantly reduces the minimum required prediction horizon (150 steps) in comparison with the method using the standard linear controller introduced in [Kohler et al., 2018] which requires 1152 steps to stabilize the system. However, both the results are not ready for real implementation since the computing time under the case of 150 steps is already 242 milliseconds on average. When the practical prediction horizon of 10 steps is employed, the uMPC controller requires only 51.3 milliseconds for one iteration on average, however, the convergence time is very long which is up to 12.5 seconds - the highest value in the analysis.

Controller's information	Terminal region	Prediction horizon	Convergence time (95%)	Computing time [ms]		
				Mean	Min	Max
Ter. invariant set Sce. 1, Sec. 4.2.1.3	$\mathcal{S}(P_1, r_1)$ in (4.2.79) $\mathsf{p}_0 \notin \mathcal{S}(P_1, r_1)$	7	4.4	55	46.3	62.5
Ter. invariant set Sce. 2, Sec. 4.2.1.3	$\mathcal{S}(P_2, r_2)$ in (4.2.79) $\mathsf{p}_0 \in \mathcal{S}(P_2, r_2)$	2	5.2	42	31.2	62.5
qMPC design Sce. 3, Sec. 4.2.1.3	Ω_α in (3.4.6) $\mathsf{p}_0 \notin \Omega_\alpha$	12	2.6	77	62.5	109
Ter. δ-invariant set Sec. 4.2.2.4	$\mathcal{R}$ in (4.2.91) $\mathsf{p}_0 \notin \mathcal{R}$	8	3.2	46.5	31.2	63
uMPC design Sce. 1, Sec. 4.3.3	not applicable	150	2.6	242	226	283
uMPC design Sce. 2, Sec. 4.3.3	not applicable	10*	12.5	51.3	31.2	78.1
uMPC design [Kohler et al., 2018]	not applicable	1152	Not simulated			

Notations: qMPC: quasi-infinite horizon NMPC [Chen and Allgöwer, 1998], uMPC: NMPC design without terminal stabilizing constraints (also referred as *unconstrained NMPC* [Grüne, 2009]) and *: without stability proof.

Table 4.4.1: Comparison between different NMPC designs under simulation.

In the next section, the two NMPC controllers using the terminal invariant set $\mathcal{S}(P_1, r_1)$ as considered under Scenario 1 in Section 4.2.1.3 and the terminal δ-invariant set $\mathcal{R}$ as considered in Section 4.2.2.4 will be employed for conducting real experimental tests over the Crazyflie 2.0 nano-quadcopter platform (c.f. Section 2.5.1).

4.5 Experimental validation

This section introduces the experimental validation of the two proposed NMPC controllers: using the terminal invariant set $\mathcal{S}(P, r)$ as detailed in Proposition 4.2.5 and using the terminal δ-invariant set $\mathcal{R}$ as detailed in Proposition 4.2.21 over the laboratory Crazyflie 2.0 (CF) nano-quadcopter. The controllers are calculated by using a station computer and only the set-point consisting of the thrust and the three angles is sent to the Crazyflie 2.0 quadcopter [Nguyen et al., 2018b]. The station computer is capable of solving the NMPC optimization problem (4.2.1)–(4.2.3) as its original formulation within the chosen sampling time $\delta_t = 0.1$ seconds by using Pyomo [Hart et al., 2011] and the IPOPT solver [Wächter and Biegler, 2006] in Python 3.0. In the following, we describe the limits of the platform and the mismatches between the theoretically nominal NMPC application and real implementation, then, illustrates the obtained results of the two proposed NMPC controllers.

4.5.1 Experimental validation limits and how to overcome them

Firstly, the yaw angle ψ as in (4.1.1) is assumed to be a known constant. However, maintaining a constant direction angle for an aerial vehicle is obviously impossible for long running times. Thus, we try to stabilize the yaw angle of the CF around zero and update the actual yaw value to the NMPC optimization problem at each sampling time. This alternative approach still guarantees the nominal stability of the closed-loop scheme with a less strict assumption, i.e., requiring the yaw angle value to be constant only along the prediction horizon [4].

Secondly, the execution time is always significant when considering an NMPC controller, especially for controlling the Crazyflie 2.0 quadcopter system with the required sampling time of 0.1 seconds. Thus, even with a perfect state feedback at time instant t, we cannot obtain the MPC control action immediately at the same time instant t as assumed in (4.2.3). Hence, we have to relax this assumption by introducing the MPC input computed from information received at time t to the CF system at time $t + \delta$ (i.e., we assume the controller requires a computation time less than δ).

The experimental platform consists of a Crazyflie 2.0 nano-quadcopter, a Loco Positioning system (LPS) and a Z-ranger deck [Nguyen et al., 2018b,Giernacki et al., 2017]. LPS provides the x and y positions of the CF system but with low accuracy (around 10 cm) (green and blue lines in Figure 4.5.1 within the first 2 seconds) while the Z-ranger deck measures only the distance between the CF and the ground (state z) with much higher precision (around few millimeters) (red lines in Figure 4.5.1). Therefore, in order to mitigate the issues raised by the imprecise estimation of the positions x and y, we fix the initial state at $\mathsf{p}_0 = [0\ 0.3\ -0.2\ 0\ 0\ 0]^\top$ so that the motion along the x-axis is reduced and hence, we observe clearly the motions along the y and z axes. We consider two scenarios over which we conduct the experimental tests:

Scenario 1: use the NMPC controller (4.2.1)–(4.2.3) with the terminal invariant set $\mathcal{S}(P_1, r_1)$ as in (4.2.79) with P_1, r_1 the parameters given in Table 4.2.1 (i.e., Scenario 1 in Section 4.2.1.3).

Scenario 2: use the NMPC controller (4.2.1)–(4.2.3) with the terminal δ-invariant set $\mathcal{R}$ as in (4.2.91) and with the tuning parameters given in Table 4.2.3 in Section 4.2.2.4.

Furthermore, the velocity of the CF is estimated by using a Kalman filter with the required noise information measured before taking off (during the gray time interval given in Figure 4.5.1).

[4] The stability is still guaranteed since all the required ingredients detailed in Propositions 4.2.5 and 4.2.21 hold with a general constant yaw angle $\psi \in [-\pi, \pi]$.

Remark 4.5.1. If the reader is interested in using embedded NMPC with low-power hardware, a discussion on how to re-formulate the NMPC problem (4.2.1) into its approximated quadratic formulations, and thereafter, solve them by using a modified interior-point solver is detailed in [Zanelli et al., 2018]. Another discussion on solving a linear MPC problem with a 8-bit microcontroller by using a convex lifting method is presented in [Gulan et al., 2017]. □

4.5.2 Experimental results

Figure 4.5.1 shows the experimental results along the x (green), y (blue) and z (red) under two scenarios which shows that both the proposed NMPC controllers succeed in stabilizing the CF quadcopter. It can be seen again that the terminal δ-invariant set $\mathcal{R}$ under Scenario 2 provides faster convergence speed than the invariant set $\mathcal{S}(P_1, r_2)$ under Scenario 1 (3.6 seconds vs. 4.2 seconds, as illustrated by the red plot in Figure 4.5.1). The NMPC controller with the terminal δ-invariant set under Scenario 2 also results in more input consumption as clearly observed from Figure 4.5.2a (dashed lines) in which the maximum values of the thrust and the pitch angle under Scenario 2 (dashed lines) are significantly higher than thoses of the NMPC controller using terminal invariant set $\mathcal{S}(P_1, r_1)$ (solid lines). All the inputs respect their constraints. The computing times of both NMPC controllers under experiment are given in Figure 4.5.2b are higher when compared to the simulation results given in Table 4.4.1 but they still share the same trend, i.e., the NMPC controller using the terminal δ-invariant set $\mathcal{R}$ (i.e., using terminal box-type linear constraints) under Scenario 1 (red line in Figure 4.5.2b) provides less computational burden than the controller using the terminal invariant set $\mathcal{S}(P_1, r_1)$ (i.e., terminal quadratic constraints) (blue line in Figure 4.5.2b) with their average computing time of 51.8 milliseconds and 61.4 milliseconds, respectively. For the whole simulation time, the computing times of both the two controllers are smaller than the sampling time $\delta_t = 0.1$ seconds. The obtained results are summarized in Table 4.5.1.

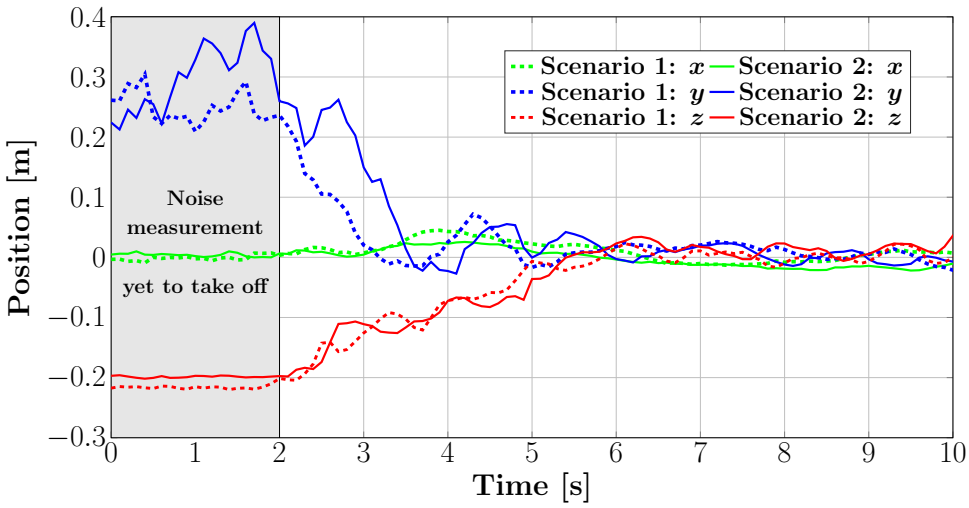
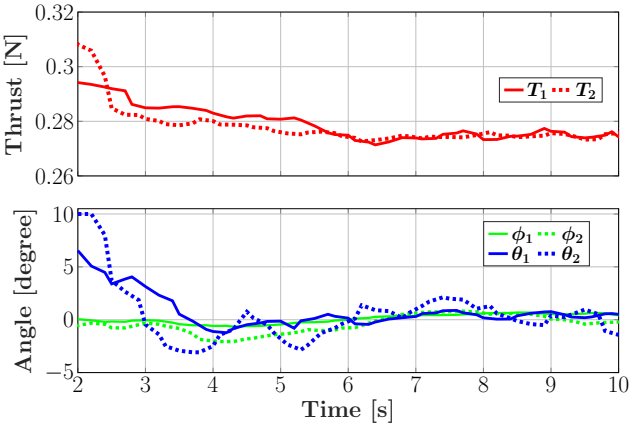
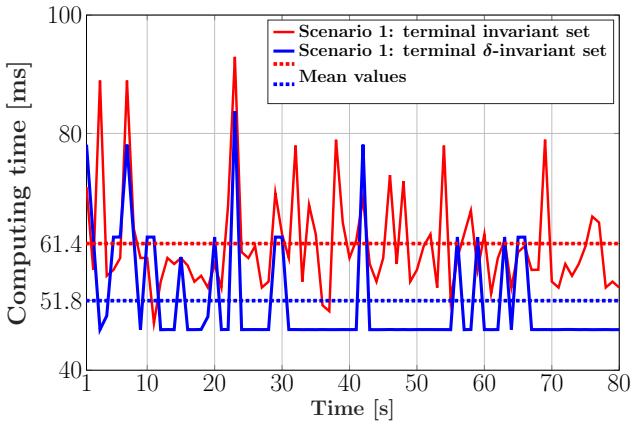


Figure 4.5.1: Position convergences under two experimental tests.

The experimental results clearly confirm the effectiveness of the proposed approaches, i.e., employing the feedback linearization controller (2.4.1) to design the terminal regions which guarantee the stability of the corresponding NMPC controllers. The terminal regions can be either the invariant set $\mathcal{S}(P, r)$ as in (4.2.79) or the δ-invariant set $\mathcal{R}$ as in (4.2.91). The two



(a) Inputs under two experimental tests.



(b) Computing time under two experimental tests.

Figure 4.5.2: Input results and computing time of the two NMPC controllers under two experimental tests.

Controller's information	Terminal region	Prediction horizon	Convergence time (95%)	Computing time		
				Mean	Min	Max
Ter. invariant set Sce. 1, Sec. 4.2.1.3	$\mathcal{S}(P_1, r_1)$ in (4.2.79) $\mathsf{p}_0 \notin \mathcal{S}(P_1, r_1)$	7	4.2	61.4	48	92.9
Ter. δ-invariant set Sec. 4.2.2.4	$\mathcal{R}$ in (4.2.91) $\mathsf{p}_0 \notin \mathcal{R}$	8	3.6	51.8	46.8	83.7

Table 4.5.1: Comparison between two NMPC designs using the invariant and δ-invariant sets as their terminal constraint sets under experiment.

proposed sets are the original contributions of the thesis which provide several unique and useful advantages, i.e.: the ease of tuning (i.e., enlarging and vice versa) for invariant set $\mathcal{S}(P, r)$ as in (4.2.79) and the clear interpretation, simple construction for the δ-invariant set $\mathcal{R}$ as in (4.2.91) which allows to reduce much of the complexity encountered in a typical NMPC problem.

4.6 Concluding remarks and open questions

In this chapter, we have addressed the problem of exploiting the feedback linearization (FL) controller $u_{\text{FL}}(K\mathsf{p}, \psi)$ as defined in (4.1.4) for designing several NMPC schemes with guaranteed stability. More precisely, the original contributions include:

1. an NMPC design using terminal invariant set with semi-global asymptotic stability guarantee:
   The design employs a terminal invariant set and a terminal cost for ensuring the recursive feasibility and asymptotic stability following the existing design principles in the literature [Mayne et al., 2000]. The originalities lie in the construction of the invariant set under the aforementioned FL controller and the corresponding *relatively* easy tuning process (in comparison with employing a standard linear controller as in the quasi-infinite horizon NMPC design [Chen and Allgöwer, 1998]). We prove that the set can be enlarged infinitely in the absence of state constraints, hence, the domain of attraction can also be expanded unlimitedly. Therefore, the proposed NMPC design achieve the semi-global stability property (c.f. Definition 4.2.3). The controller is validated under both simulation

and experiment. The controller provides good computing times (i.e., down to 42 milliseconds per step when using the terminal invariant set which contains the initial state, hence, possibly employing the prediction horizon of 2 steps as detailed in Table 4.4.1) and also good control performance (i.e., convergence time of 4.4 seconds under simulation and 4.2 seconds under experiment with a "medium-size" terminal invariant set $\mathcal{S}(P_2, r_2)$ as in Tables 4.4.1 and 4.5.1).

2. an NMPC design using terminal δ-invariant set with asymptotic stability guarantee:
The design is constructed similarly to the previous one but using a δ-invariant set (with δ the sampling time of the controller, c.f. Definition 4.2.14) as its terminal region. The set allows the state to escape from itself but guarantees that it will return inside at predefined periodic time instants and always stay within the admissible set. Due to this relaxation, the terminal region can be obtained through simple box-type constraints as in (4.2.91). This can reduce the complexity of the NMPC optimization problem while still guaranteeing the closed-loop stability. The proposed controller is also validated under both simulation and experiment with good results on the computing times (i.e., 46.5 milliseconds per step under simulation and 51.8 milliseconds under experiment in comparison with 55 and 61.4 milliseconds under similar scenarios when using the terminal invariant set with similar size as detailed in Tables 4.4.1 and 4.5.1).

3. a new process to calculate the required minimum prediction horizon for guaranteeing the stability of an NMPC design without terminal stabilizing constraints and the corresponding detailed analysis on the tuning parameters:
This *unconstrained* NMPC design ensures the closed-loop stability by defining an appropriate prediction horizon length according to the existing Theorem 4.3.2 [Grüne, 2012, Kohler et al., 2018]. We firstly prove that the FL controller (4.1.4) satisfies Assumption 4.3.1 - the prerequisite of Theorem 4.3.2, then, derive the formulation of the required prediction horizon from the FL controller (4.1.4). Next, a thorough analysis on how to tune the parameters is conducted which shows the shortest required prediction horizon length to be 150 steps (c.f. Table 4.3.2), in comparison with the thousands steps obtained when using the approach proposed in [Kohler et al., 2018] with a standard linear controller.

In this chapter, we have shown that the use of the nonlinear FL controller (4.1.4) significantly improves the performance and also the design procedure of the NMPC schemes with guaranteed stability (i.e., the form of the terminal constraint sets, the computing time, the length of the prediction horizon and the ease for tuning them). Even though the proposed designs are particularized for the translation dynamics of the multicopter system (4.1.1), they are actually the continuations of our contribution on the use of the computed-torque control on stabilizing the NMPC scheme detailed in the previous Chapter 3. Therefore, the generalization for similar feedback linearizable systems are possible and promising. Beside that, there are still some open questions which should be regarded in order to improve the contributions of this Chapter:

- Can we avoid the assumption on the constant yaw angle ψ as in (4.1.1) when designing the proposed NMPC controllers? This actually increases much the complexity of the works since we have to take into account the mismatch on the predicted yaw angle and the actual yaw angle at the next step which is impossible to know before hand. This most probably leads to a problem of robust control design.

- Can we estimate the domain of attraction of the proposed NMPC design with terminal invariant set $\mathcal{S}(P, r)$ as in (4.2.79) with more accuracy without solving the first iteration? The aim of this question is to improve the proof of the semi-global stability given in

Section 4.2.1.2 where we enlarge the terminal invariant set $\mathcal{S}(P, r)$ as in (4.2.79) until the predefined compact set $\mathcal{X}_0$ is inside the set $\mathcal{S}(P, r)$. This actually results in an oversized domain of attraction - significantly larger than the necessary size which, hence, reduces the control performance (i.e., low convergence speed as can be seen from Table 4.4.1, second scenario).

- Can the NMPC design using the terminal δ-invariant set $\mathcal{R}$ as in (4.2.91) achieve the semi-global stability property, similarly to the one using the terminal invariant set $\mathcal{S}(P, r)$ detailed in Section 4.2.1?

- Regarding the NMPC design without terminal stabilizing constraints as detailed in Section 4.3, can we obtained a shorter prediction horizon than 150 steps as given in Table 4.3.2 by using a more complex paramterization of the tuning parameters ? I.e., for now, we are constraining the weighting matrix Q, the control gain matrix K and the symmetric matrix M as in (4.3.23)–(4.3.27) which allows only 7 decision variables from the total of 48 which are in fact possible and hence, probably restricts the obtained results.

Chapter 5

Reliable control of a quadcopter under stuck actuator fault

In the three previous chapters, we have presented the hierarchical control architecture with different control strategies for a multicopter system. Even though the control designs have been validated through various simulations and experiments, the proposed contributions still lack an anlysis on their reliability under faulty events since they may cause performance degradation and instability. In the literature, possible faults occurring in the multicopters are usually related to their rotors, such as loss of effectiveness of the rotors [Chamseddine et al., 2012, Avram et al., 2017, Hasan and Johansen, 2018], or complete loss of one, two or even three rotors [Freddi et al., 2011, Mueller and D'Andrea, 2014, Başak and Prempain, 2015, Sun et al., 2018]. One particularly intereting scenario among them is the *stuck rotor* type of fault [Yang and Lum, 2003, Jiang et al., 2016, Shao et al., 2018]. Once stuck, the faulty actuators keep rotating at a constant speed regardless of the actual control inputs. Thus, under a single stuck rotor fault, the multicopter system not only loses one degree of freedom in its control ability but also suffers from persistent disturbances [Chen and Jiang, 2005].
Henceforth, we consider the tracking of the position as the main objective (to the detriment of, e.g., tracking the yaw angle). Then, the consequence of this fault's effect on the system is based on the number of rotors: i.e., a tricopter (having two symmetrical rotors and one tail rotor tilted by another servo [Prach and Kayacan, 2018]) instantly crashes, a quadcopter (having four symmetrical rotors) loses its yaw control capability [Freddi et al., 2011, Nguyen et al., 2017b] and multicopters with more than four rotors (e.g., hexacopter with six rotors, octocopter with eight rotors) are able to compensate for the loss of one stuck rotor by using the remaining healthy rotors, hence, fully maintaining their capabilities [Saied et al., 2015]. Even if the fault can be accommodated through hardware redundancy, there is still a need to design a FTC (Fault Tolerant Control) scheme which detects and isolates the fault and, consequently, provides control reconfiguration strategies. In what follows, we find our interest in investigating the quadcopter case where the unavoidable yaw spinning motion is a significant control challenge while the three remaining healthy rotors compensate for the forth within the limit of their saturation constraints. Part of these issues are addressed in our previous paper [Nguyen et al., 2017b] where an FTC scheme for quadcopter control was developed by using feedback linearization but without providing a fault diagnosis module and considering the actuator saturation. This shortcoming was due to the complexity of the whole controlled system under input constraints but can, arguably, lead to bad behavior such as loss of stability and decrease in performance. Then, in [Nguyen et al., 2020d], we improve the works in [Nguyen et al., 2017b] by designing a fault diagnosis module particularized for the stuck rotor fault which consists of three standard elements: fault detection, fault isolation and fault estimation [Hasan and Johansen, 2018]. The residual vector is

calculated as the differences between the estimated rotor speeds and their references, hence, being able to detect the faulty rotor's misbehavior. Furthermore, the saturation constraints on the rotor speeds are fulfilled by employing the NMPC (Nonlinear Model Predictive Control) design presented in Chapter 3 with some modifications (e.g., reformulating the rotor speeds constraints into torques constraints). The control module is also reconfigured by no longer attempting to control the yaw motion. As an extended version of [Nguyen et al., 2020d], this chapter provides several contributions which, to the best of our knowledge, are new to state of the art:

- Provides a hierarchical control scheme for the trajectory tracking of a quadcopter system under actuator saturation and a stuck fault occurrence. The high level employs a feedback linearization controller while the low level switches between two different NMPC schemes according to the system's functioning states (healthy or under fault). The two NMPC designs guarantees that the rotor speeds do not exceed their limit;
- Designs a fault diagnosis module for detecting the faults of a unique rotor being stuck at varying speeds. The residue is constructed based on the differences between the estimated and reference normalized torques (given in the unit of the rotor speed).
- Proposes a model of the uncontrolled yaw motion under fault. This allows the prediction dynamics employed within the NMPC controller to be more realistic than keeping the yaw value as a constant feedback within the whole prediction horizon would have been.

The remainder of this chapter is organized as follows. Section 5.1 presents the rotor configuration of a standard quadcopter system and the modeling of the stuck actuator fault. Next, Section 5.2 introduces the hierarchical FTC scheme and the fault diagnosis module. The simulation results are given and discussed in Section 5.3. Finally, Section 5.4 draws the conclusions and presents the future directions.

5.1 Quadcopter rotor configuration under stuck fault

Recall the dynamical model of a general multicopter system $\dot{\mathsf{x}} = \mathsf{f}(\mathsf{x}, \mathsf{u})$ given in (2.1.10). The state $\mathsf{x} \triangleq [\xi\ v\ \eta\ \omega]^\top \in \mathbb{R}^{12}$ gathers the 3D position ξ, translation velocity v, Euler angles η and the body angle rates ω, all in the vector space $\mathbb{R}^3$. The input vector $\mathsf{u} \triangleq [T\ \tau_\phi\ \tau_\theta\ \tau_\psi]^\top \in \mathbb{R}^4$ represents the thrust force and the three torques acting on the three body axes. These inputs are created by the rotation of the rotors and are calculated based on the specific rotor architecture of the considered multicopter system. Particularly, we consider a quadcopter system with its four rotors being placed symmetrically along the body coordinate axes as shown in Figure 5.1.1. This configuration is the so-called "plus" type as opposed to the "cross" type illustrated in Figure 2.1.1 [Prabha et al., 2016]. The reason for choosing this "plus" configuration is that it provides separate control mechanisms for angle control, i.e., the second and fourth rotors are in charge of the roll angle while the others are responsible for pitch motion. Thus, it results in a clearer notation and simpler calculation when considering the rotor faults [Nguyen et al., 2020d, Nguyen et al., 2017b, Hasan and Johansen, 2018]. Note that, changing from "plus" to "cross" types simply requires us to rotate the body frame $\mathcal{B}$ around its ${}^{\mathcal{B}}z$ axis by 45°, hence, the contributions proposed hereinafter can be applied for all applications on the quadcopter system regardless its original configuration type.

In Figure 5.1.1, the rotating directions of the propellers are defined by the four green curved arrows with Ω_i ($i \in \{1, \ldots, 4\}$) representing their speeds. Note that, $\Omega_i \in \mathbb{R}_+$ since the four rotors can only rotate in their predefined directions. Whenever the i^{th} propeller is rotating, it creates the upward force T_i (shown by red arrows) and the torque τ_i acting in the inverse

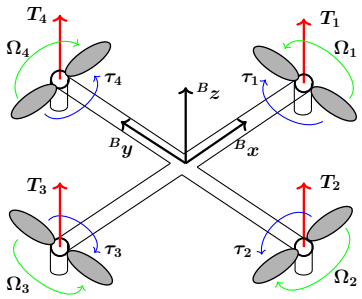


Figure 5.1.1: Quadcopter and its "plus" rotor configuration.

direction of Ω_i (shown by green curved arrows) due to the conservation law of momentum [Craig, 2005]:

$$T_i = K_T \Omega_i^2, \tag{5.1.1}$$
$$\tau_i = (-1)^i b \Omega_i^2, \tag{5.1.2}$$

for all $i \in \{1, \ldots, 4\}$ and with $K_T, b \in \mathbb{R}_+$ aerodynamics scalars. Therefore, based on the schematic shown in Figure 5.1.1, the input vector $\mathsf{u} \triangleq [T \ \tau_\phi \ \tau_\theta \ \tau_\psi]^\top$ from (2.1.10) is calculated in terms of the rotor speeds Ω_i as follows:

$$\begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \underbrace{\begin{bmatrix} K_T & K_T & K_T & K_T \\ 0 & -LK_T & 0 & LK_T \\ -LK_T & 0 & LK_T & 0 \\ -b & b & -b & b \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix}}_{\boldsymbol{\Omega}^2}, \tag{5.1.3}$$

with $L > 0$ the arm length of the quadcopter. Note that, (5.1.3) is shortened as $\mathsf{u} = M\boldsymbol{\Omega}^2$. Also, in the rest of this chapter, the square of a vector gives a vector consisting of the squares of all its elements, e.g. $\Omega^2 \triangleq= [\Omega_1^2 \ \Omega_2^2 \ \Omega_3^2 \ \Omega_4^2]^\top$ with $\Omega = [\Omega_1 \ \Omega_2 \ \Omega_3 \ \Omega_4]^\top$. The four rotors are

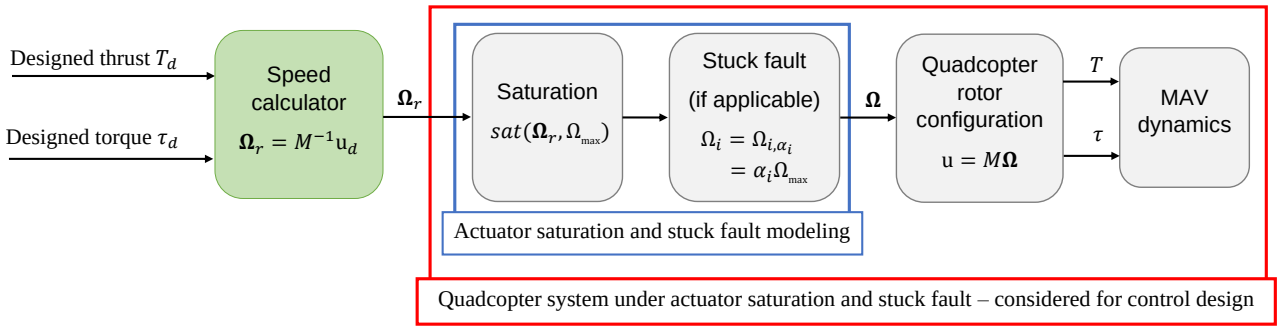


Figure 5.1.2: Further details of Figure 2.3.1: transformation from the designed inputs to the real inputs within the quacopter system under actuator saturation and stuck fault.

assumed to have their own speed controller running at a significantly higher frequency. Therefore, we consider that the real rotor speeds Ω_i ($i \in \{1, \ldots, 4\}$) can instantly track their references, denoted by $\Omega_{i,r}$, but are still limited by their maximum rotating speed denoted by $\Omega_{\max} \in \mathbb{R}_+$. Furthermore, the rotors can be affected by *stuck actuator* fault [Chen and Jiang, 2005, Yang

and Lum, 2003, Shao et al., 2018] which forces the rotating speed Ω_i of the i^{th} faulty rotor to remain stuck at a constant value $\Omega_{i,\alpha}$, regardless of the actual reference value $\Omega_{i,r}$. Hence, under both the saturation and fault effects as illustrated within the blue rectangle in Figure 5.1.2, the rotating speed of the i^{th} rotor is modeled through:

$$\Omega_i = \begin{cases} \text{sat}(\Omega_{i,r}, \Omega_{\max}), & \text{under nominal condition,} \\ \Omega_{i,\alpha}, & \text{under stuck fault,} \end{cases} \tag{5.1.4}$$

with the saturation function $\text{sat}(\cdot)$ defined in (2.3.4). The constant stuck value $\Omega_{i,\alpha}$ is expressed in terms of $0 \leq \alpha \leq 1$ as a fraction of the maximum rotor speed, i.e.:

$$\Omega_{i,\alpha} = \alpha \Omega_{\max}. \tag{5.1.5}$$

Within this paper, we consider at most one rotor being stuck at a time, so that the quadcopter can still track a 3D reference trajectory since having one stuck rotor means that one degree of freedom is lost in the control of the quadcopter system (in the sense of tracking the position component of the trajectory while losing command over the yaw component). Thus, if the i^{th} rotor is stuck at $\Omega_{i,\alpha}$ from (5.1.5), the rotor speed vector is denoted as follows:

$$\boldsymbol{\Omega}_{i,\alpha} \triangleq [\Omega_1 \dots \Omega_{i,\alpha} \dots \Omega_4]^\top, \tag{5.1.6}$$

which further leads to the distinction between the nominal functioning and faulty case when considering the rotor-to-input relation as in (5.1.3):

$$\mathsf{u} = \begin{cases} M\boldsymbol{\Omega}^2, & \text{under nominal condition,} \\ M\boldsymbol{\Omega}_{i,\alpha}^2, & \text{if the } i^{th} \text{ rotor under fault (stuck),} \end{cases} \tag{5.1.7}$$

with the nominal rotor speed vector $\boldsymbol{\Omega}$ defined in (5.1.3). By this, we also complete the quadcopter modeling under actuator saturation and stuck fault consideration which is illustrated by the red rectangle block shown in Figure 5.1.2. In the followings, we introduce some new notations for easily describing the system when under the fault of the i^{th} stuck rotor ($i \in \{1 \dots 4\}$):

- $M_i \in \mathbb{R}^4$ defines the i^{th} column while $M_{!i}$ gathers all the other columns except the i^{th} column of the matrix M from (5.1.3).
- $\widehat{\Xi}$ and $\breve{\Xi}$ for a matrix $\Xi \in \mathbb{R}^{n \times m}$ gather the first and the last $(n-1)$ rows of Ξ. The definitions are also applied for the vector case where $m = 1$.
  Below are some examples employed throughout the chapter. From the input vector u as in (5.1.3), we have:
  $$\widehat{\mathsf{u}} = [T \ \tau_\phi \ \tau_\theta]^\top. \tag{5.1.8}$$
  From the matrix M as in (5.1.3), we have:
  $$\widehat{M_1} = [K_T \ 0 \ -LK_T]^\top, \ \breve{M_4} = [LK_T \ 0 \ b]^\top. \tag{5.1.9}$$
  $$\widehat{M_{!2}} = \begin{bmatrix} K_T & \cancel{K_T} & K_T & K_T \\ 0 & -LK_T & 0 & LK_T \\ -LK_T & \cancel{0} & LK_T & 0 \end{bmatrix} = \begin{bmatrix} K_T & K_T & K_T \\ 0 & 0 & LK_T \\ -LK_T & LK_T & 0 \end{bmatrix}. \tag{5.1.10}$$

- The references of the four rotor speeds from (5.1.4) are gathered into:

$$\mathbf{\Omega}_r \triangleq [\Omega_{1,r} \dots \Omega_{4,r}]^\top, \tag{5.1.11}$$

while $\mathbf{\Omega}_{r,!i} \in \mathbb{R}^3$ represents the three references except the one of the i^{th} stuck rotor. E.g. for $i = 2$:

$$\mathbf{\Omega}_{r,!2} \triangleq [\Omega_{1,r}\ \Omega_{3,r}\ \Omega_{4,r}]^\top. \tag{5.1.12}$$

The next section will address a hierarchical FTC (Fault Tolerant Control) scheme whose goal is to counteract the stuck fault through the feedback linearization position controller presented in Section 2.4 and the NMPC attitude controller presented in Section 3.5.

5.2 Fault Tolerant Control design

Figure 5.2.1 presents the control process proposed in this chapter to accommodate the fault corresponding to a single stuck rotor [Nguyen et al., 2020d]. The design is built upon the standard hierarchical two-layer control scheme as given in Figure 2.3.1. Under nominal functioning, the position controller and the attitude controller operate similarly to the standard design as introduced in Section 2.3 while the speed calculator block converts the desired inputs $\mathsf{u}_d \triangleq [T_d\ \mathcal{T}_{\phi_d}\ \mathcal{T}_{\theta_d}\ \mathcal{T}_{\psi_d}]^\top$ into four rotor speed references $\mathbf{\Omega}_r$ (5.1.11) by using the inversion of the speed-to-rotor relation (5.1.3). Then, the references are sent to the quadcopter system which actually stands for the whole process subject to actuator saturation and stuck fault as given inside the red rectangle in Figure 5.1.2.

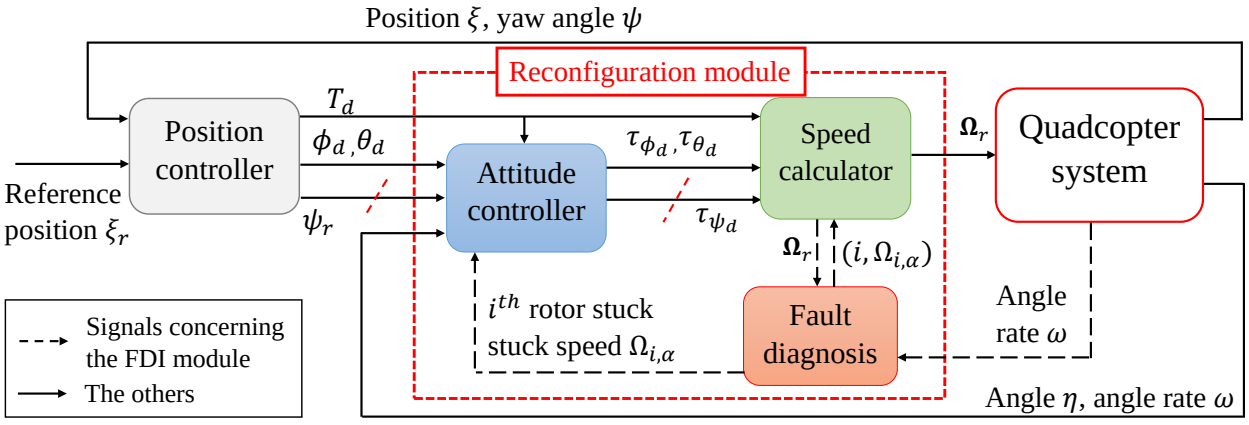


Figure 5.2.1: Hierarchical Fault Tolerant Control scheme for the quadcopter system.

Besides that, the speed references $\mathbf{\Omega}_r$ are also sent to the fault diagnosis module (red block in Figure 5.2.1) in which, they are compared with the estimated rotor speeds (represented by *normalized* torques in the chapter) in order to detect the stuck fault, identify the stuck rotor and estimate the speed at which is stuck. Under fault, as we lose the control of one rotor, we reconfigure the controller structure to no longer control the yaw motion. The two signals with dashed slash in Figure 5.2.1 (i.e., yaw angle reference, ψ_r, and desired yaw torque, $\mathcal{T}_{\psi_d}$) will be discarded. Furthermore, based on the information of the fault provided by the fault diagnosis block, the attitude controller (blue block in Figure 5.2.1) reconfigurates its control law and the speed calculator block modifies the speed transformation (e.g., the nominal functioning given in (5.1.3)) to provide appropriate rotor speed references in order to counteract the detected fault. Note that, the reconfiguration block at the low level is required to run at a higher frequency than the position controller in order to achieve the stability of the whole scheme [Scattolini, 2009].

5.2.1 Hierarchical FTC control scheme

In this section, we present the designs of the two control layers employed within the hierarchical FTC scheme shown in Figure 5.2.1. At first, the role of the position controller does not change while under stuck fault since we choose to only renounce controlling the yaw motion. Therefore, in order to simplify the whole control process, we make use of the feedback linearization controller from Section 2.4 which provides the desired values of the thrust $T_d = T_{\text{FL}}$ as given in (2.4.1a) and the roll, pitch angles $\phi_d = \phi_{\text{FL}}$, $\theta_d = \phi_{\text{FL}}$ as given in (2.4.1b)–(2.4.1c).
The main contributions of the section will be the designs of the attitude controller and the speed calculator shown in Figure 5.2.1. The FTC law alternates between the healthy and under fault modes. Each of these correspond to a particular combination of the attitude controller and the speed calculator. In order to avoid the rotor saturation effect as in (5.1.4) and to respect the state constraints from (2.1.12), the NMPC strategy introduced in Section 3.1 appears again as a good control candidate. The main problem is that the saturation constraints from (5.1.4) actually imposes the time-varying polytopic constraints on the desired torques τ_d and furthermore, the yaw motion becomes uncontrollable under fault. These prevent us from directly applying the attitude control design given in Section 3.5 even under nominal functioning. In the following sections, the solutions will be addressed for the nominal and faulty cases, respectively.

5.2.1.1 Nominal functioning

Since we are now considering the quadcopter system as introduced in Section 5.1, the healthy NMPC attitude controller requires some important changes in comparison with the NMPC design proposed in Section 3.5. The input constraints are not taken as those given in (3.5.5), i.e., $|\tau_d| \leq \tau_{\max}$ but have to take into account the desired thrust T_d obtained from the position controller (c.f. Figure 5.2.1) and the rotor speed constraints [Nguyen et al., 2020d]:

$$\tau_d \in \mathcal{S}(T_d) = \left\{ \tau_d \in \mathbb{R}^3 \middle| 0 \leq M^{-1}\mathsf{u}_d \leq \Omega_{\max}^2 \right\}, \tag{5.2.1}$$

with $\mathsf{u}_d \triangleq [T_d \ \tau_d]^\top$, M in (5.1.3) and $\Omega_{\max}$ the maximum rotor speed in (5.1.4). Furthermore, the terminal constraint set $\mathcal{B}^*(\eta_d, \varepsilon)$ from (3.5.24) is constructed to be constraint admissible w.r.t the torque constraint $|\tau_d| \leq \tau_{\max}$ from (3.5.5) and not for the set $\mathcal{S}(T_d)$. Therefore, the next step is to define the parameter $\tau_{\max} \in \mathbb{R}_+^3$ such that the condition (5.2.1) holds for all the feasible values of the desired thrust T_d which is facilitated by the following proposition.

Proposition 5.2.1 ([Nguyen et al., 2020d]). *There always exists $\tau_{max} \in \mathbb{R}_+^3$ such that the following holds:*

$$[-\tau_{max}, \tau_{max}] \subseteq \mathcal{S}(T_d), \ \forall \ T_d \in [T_{d_{min}}, T_{d_{max}}], \tag{5.2.2}$$

where $[T_{d_{min}}, T_{d_{max}}]$ represents the range of the desired thrust received from the position controller and satisfies $0 < T_{d_{min}} < T_{d_{max}} < 4K_T\Omega_{max}^2$, with Ω_{max} the maximum rotor speed.

Proof. From the definition of $\mathcal{S}(T_d)$ in (5.2.1), the polytopic constraints on the desired torques τ_d is explicitly given by:

$$-\frac{T_d}{4K_T}\begin{bmatrix}1\\1\\1\\1\end{bmatrix} \leq \begin{bmatrix} 0 & -\frac{1}{2LK_T} & -\frac{1}{4b} \\ -\frac{1}{2LK_T} & 0 & \frac{1}{4b} \\ 0 & \frac{1}{2LK_T} & -\frac{1}{4b} \\ \frac{1}{2LK_T} & 0 & \frac{1}{4b} \end{bmatrix} \begin{bmatrix} \tau_{\phi_d} \\ \tau_{\theta_d} \\ \tau_{\psi_d} \end{bmatrix} \leq \left(\Omega_{\max}^2 - \frac{T_d}{4K_T}\right)\begin{bmatrix}1\\1\\1\\1\end{bmatrix}, \tag{5.2.3}$$

with K_T, b, L from (5.1.3) the constant parameters of the quadcopter system. Next, the constraints (5.2.3) are split into two separate systems of linear inequalities as follows:

$$-\frac{T_d}{4K_T}\begin{bmatrix}1\\1\end{bmatrix} \leq \begin{bmatrix} -\frac{1}{2LK_T} & \frac{1}{4b} \\ \frac{1}{2LK_T} & \frac{1}{4b}\end{bmatrix}\begin{bmatrix}\tau_{\phi_d}\\ \tau_{\psi_d}\end{bmatrix} \leq \left(\Omega_{\max}^2 - \frac{T_d}{4K_T}\right)\begin{bmatrix}1\\1\end{bmatrix}, \tag{5.2.4}$$

$$-\frac{T_d}{4K_T}\begin{bmatrix}1\\1\end{bmatrix} \leq \begin{bmatrix} -\frac{1}{2LK_T} & -\frac{1}{4b} \\ \frac{1}{2LK_T} & -\frac{1}{4b}\end{bmatrix}\begin{bmatrix}\tau_{\theta_d}\\ \tau_{\psi_d}\end{bmatrix} \leq \left(\Omega_{\max}^2 - \frac{T_d}{4K_T}\right)\begin{bmatrix}1\\1\end{bmatrix}. \tag{5.2.5}$$

The aforementioned constraints imply that $(\tau_{\phi_d}, \tau_{\psi_d})$ and $(\tau_{\theta_d}, \tau_{\psi_d})$ are required to stay inside the polytopes $\mathbb{P}_1$ and $\mathbb{P}_2$ defined as follows:

$$(\tau_{\phi_d}, \tau_{\psi_d}) \in \mathbb{P}_1 = \text{Conv}(V_1), \tag{5.2.6}$$

$$(\tau_{\phi_d}, \tau_{\psi_d}) \in \mathbb{P}_1 = \text{Conv}(V_2), \tag{5.2.7}$$

with the vertices sets V_1 and V_2 obtained from (5.2.4)–(5.2.5):

$$\begin{aligned} V_1 = \Bigg\{ &\left(0, \left(4\Omega_{\max}^2 - \frac{T_d}{K_T}\right)b\right), \left(-LK_T\Omega_{\max}^2, \left(2\Omega_{\max}^2 - \frac{T_d}{K_T}\right)b\right) \\ &\left(0, -\frac{T_d}{K_T}b\right), \left(LK_T\Omega_{\max}^2, \left(2\Omega_{\max}^2 - \frac{T_d}{K_T}\right)b\right)\Bigg\}, \end{aligned} \tag{5.2.8}$$

$$\begin{aligned} V_2 = \Bigg\{ &\left(0, -\left(4\Omega_{\max}^2 - \frac{T_d}{K_T}\right)b\right), \left(-LK_T\Omega_{\max}^2, -\left(2\Omega_{\max}^2 - \frac{T_d}{K_T}\right)b\right) \\ &\left(0, \frac{T_d}{K_T}b\right), \left(LK_T\Omega_{\max}^2, -\left(2\Omega_{\max}^2 - \frac{T_d}{K_T}\right)b\right)\Bigg\}. \end{aligned} \tag{5.2.9}$$

Furthermore, the polytopes $\mathbb{P}_1$ and $\mathbb{P}_2$ from (5.2.4)–(5.2.5) always overlap with each other due to $T_d < 4K_T\Omega_{\max}^2$ as can be seen from their illustrations given in Figure 5.2.2. The intersection is given by:

$$\mathbb{P}_1 \cap \mathbb{P}_2 = \begin{cases} \text{Conv}\left\{\left(0, \frac{T_d}{K_T}b\right), \left(\frac{T_dL}{2}, 0\right), \left(0, -\frac{T_d}{K_T}b\right), \left(-\frac{T_dL}{2}, 0\right)\right\}, & \text{if } T_d \leq 2K_T\Omega_{\max}^2, \\ \text{Conv}\left\{\begin{array}{l} \left(0, \left(4\Omega_{\max}^2 - \frac{T_d}{K_T}\right)b\right), \left(2LK_T\Omega_{\max}^2 - \frac{T_dL}{2}, 0\right), \\ \left(0, -\left(4\Omega_{\max}^2 - \frac{T_d}{K_T}\right)b\right), \left(-2LK_T\Omega_{\max}^2 + \frac{T_dL}{2}, 0\right) \end{array}\right\}, & \text{if } T_d > 2K_T\Omega_{\max}^2, \end{cases} \tag{5.2.10}$$

and especially, $\mathbb{P}_1 \equiv \mathbb{P}_2$ when $T_d = 2K_T\Omega_{\max}^2$. The overlap region facilitates a simple choice of the box constraints on the desired torques τ_d given as follows (c.f. Figure 5.2.2):

$$\langle|\tau_{\phi_d}|, |\tau_{\theta_d}|\rangle \leq \frac{T_d b}{2K_T} \text{ and } |\tau_{\psi_d}| \leq \frac{T_dL}{4}, \text{ if } T_d \leq 2K_T\Omega_{\max}^2, \tag{5.2.11}$$

$$\langle|\tau_{\phi_d}|, |\tau_{\theta_d}|\rangle \leq \left(2\Omega_{\max}^2 - \frac{T_d}{2K_T}\right)b \text{ and } |\tau_{\psi_d}| \leq LK_T\Omega_{\max}^2 - \frac{T_dL}{4}, \text{ if } T_d > 2K_T\Omega_{\max}^2, \tag{5.2.12}$$

which, if satisfied, guarantees that $\tau_d \in \mathcal{S}(T_d)$ as required in (5.2.1). Therefore, the maximum desired torques $\tau_{\max} = [\tau_{\phi_{\max}}\ \tau_{\theta_{\max}}\ \tau_{\psi_{\max}}]^\top$ as required in (5.2.2) can be found by constraining

$\tau_{\phi_{\max}} = \tau_{\theta_{\max}}$ and then, finding the minimum values of each limits for all $T_d \in [T_{d_{\min}}, T_{d_{\max}}]$. In the following, we provide an example for the case of $T_{d_{\min}} < 2K_T\Omega^2_{\max} < T_{d_{\max}}$:

$$\begin{aligned}\tau_{\phi_{\max}} &= \min\left\{ \min_{T_d\in[T_{d_{\min}},2K_T\Omega^2_{\max}]} \frac{T_d}{2K_T}b, \min_{T_d\in[2K_T\Omega^2_{\max},T_{d_{\max}}]} \left(2\Omega^2_{\max} - \frac{T_d}{2K_T}\right)b \right\} \qquad (5.2.13)\\ &= \min\left\{ \frac{T_{d_{\min}}}{2K_T}b, \left(2\Omega^2_{\max} - \frac{T_{d_{\max}}}{2K_T}\right)b \right\},\\ \tau_{\theta_{\max}} &= \tau_{\phi_{\max}}, \qquad (5.2.14)\\ \tau_{\psi_{\max}} &= \min\left\{ \min_{T_d\in[T_{d_{\min}},2K_T\Omega^2_{\max}]} \frac{T_dL}{4}, \min_{T_d\in[2K_T\Omega^2_{\max},T_{d_{\max}}]} LK_T\Omega^2_{\max} - \frac{T_dL}{4} \right\} \qquad (5.2.15)\\ &= \min\left\{ \frac{T_{d_{\min}}L}{4}, LK_T\Omega^2_{\max} - \frac{T_{d_{\max}}L}{4} \right\}.\end{aligned}$$

The algorithms for the two other cases when $2K_T\Omega^2_{\max} < T_{d_{\min}} < T_{d_{\max}}$ and $T_{d_{\min}} < T_{d_{\max}} < 2K_T\Omega^2_{\max}$ are obtained similarly. This completes the proof. □

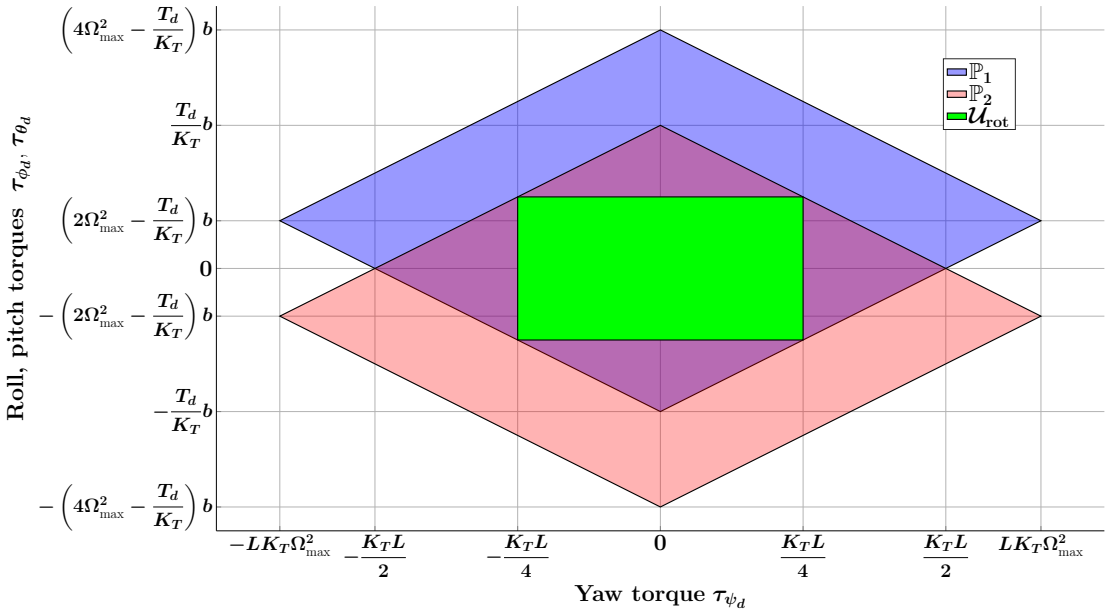


Figure 5.2.2: Illustration of the polytopes $\mathbb{P}_1$ and $\mathbb{P}_2$ from (5.2.4)–(5.2.5) in comparison with a feasible choice of the constraint set $\mathcal{U}_{\text{rot}}$ from (3.5.5) when $T_d \le 2K_T\Omega^2_{\max}$.

Then, by using the feedback linearization law $T_d = T_{\text{FL}}$ from (2.4.1a), we can find the range of the desired thrust T_d as required in (5.2.2):

$$m|U_z - g| \le T_{\text{FL}} \le m\sqrt{U_x^2 + U_y^2 + (U_z + g)^2}, \qquad (5.2.16)$$

with U_x, U_y, U_z the positive scalars defined in Proposition 2.4.2 and m the system mass. Thus, by applying Proposition 5.2.1, we obtain the value of $\tau_{\max}$ such that the following holds:

$$[-\tau_{\max}, \tau_{\max}] \subseteq \mathcal{S}(T_d),\ \forall T_d \in \left[m|U_z - g|, m\sqrt{U_x^2 + U_y^2 + (U_z + g)^2}\ \right], \qquad (5.2.17)$$

with $\mathcal{S}(T_d)$ from (5.2.1). By this, we achieve all the required elements for constructing the NMPC scheme proposed in Section 3.5.

Remark 5.2.2. Bear in mind that we need to ensure $0 < T_{d_{\min}} < T_{d_{\max}} < 4K_T\Omega_{\max}^2$, with $\Omega_{\max}$ the maximum rotor speed, as required by Proposition 5.2.1. Hence, this transforms into the conditions on (U_x, U_y, U_z) from (5.2.16) as follows:

$$0 < m|U_z - g| < m\sqrt{U_x^2 + U_y^2 + (U_z + g)^2} < 4K_T\Omega_{\max}^2. \tag{5.2.18}$$

To guarantee these, we only need to define $T_{\max}$ from (2.1.13) such that $T_{\max} < 4K_T\Omega_{\max}^2$. It is due to the fact that $U_z < g$ is already established in (2.4.5) and $m\sqrt{U_x^2 + U_y^2 + (U_z + g)^2} < T_{\max}$ is imposed in (2.4.7). □

Then, the desired torques τ_d as in (5.2.1) are combined with the thrust T_d obtained from the position controller at the high level to achieve the reference rotor speeds $\mathbf{\Omega}_r$ inside the speed calculator block (c.f. Figure 5.2.1):

$$\mathbf{\Omega}_r^2 = M^{-1}\mathsf{u}_d, \tag{5.2.19}$$

with $\mathbf{\Omega}_r^2$ gathering the squares of the speed references, M the rotor-to-input configuration matrix as in (5.1.3) and u_d as in (5.2.1).
Since the condition $\tau_d \in \mathcal{S}(T_d)$ from (5.2.1) is fulfilled by the NMPC attitude controller, the constraint $0 \leq M^{-1}\mathsf{u}_d \leq \Omega_{\max}^2$ as in (5.2.1) is guaranteed. This also ensures that the reference rotor speeds $\mathbf{\Omega}_r$ calculated in (5.2.19) do not exceed their maximum value $\Omega_{\max}$ as in (5.1.4), hence, do not trigger the saturation effect. Therefore, under nominal functioning, the proposed control scheme ensures:

$$\mathbf{\Omega} = \mathbf{\Omega}_r, \ \mathsf{u} = \mathsf{u}_d, \tag{5.2.20}$$

with $\mathbf{\Omega}$ the real rotor speeds obtained from (5.1.4), u the real input vector obtained from (5.1.7) and u_d the desired input as in (5.2.19). As the result, the closed-loop controlled system is stable according to the hierarchical control design theory detailed in Section 2.3 since the feedback linearization position controller at high level is globally asymptotically stable from (2.4.16) and the NMPC attitude controller is also exponentially stable from Lemma 3.1.1 (i.e., under nominal consideration). Hereinafter, we show the implementation formulation of the nominal NMPC attitude controller presented in this section. The controller is given in the standard discrete NMPC form with the prediction model discretized at the same NMPC rate (c.f. the NMPC sampling time δ and the model discretization step Δ as in (3.1.12)–(3.1.13)).

NMPC design for the nominal attitude controller:
The NMPC attitude controller runs at the sampling time δ_{att} with its open-loop optimization problem at time step k given in discrete form as follows:

$$\begin{aligned}\min_{\bar{\tau}(\cdot)} \sum_{s=0}^{N_p-1} \left(\begin{bmatrix} \bar{\eta}(s) - \eta_d(k) \\ \dot{\bar{\eta}}(s) \end{bmatrix}^\top Q \begin{bmatrix} \bar{\eta}(s) - \eta_d(k) \\ \dot{\bar{\eta}}(s) \end{bmatrix} + \bar{\tau}^\top(s) R\, \bar{\tau}(s) \right) \\ + \begin{bmatrix} \bar{\eta}(N_p) - \eta_d(k) \\ \dot{\bar{\eta}}(N_p) \end{bmatrix}^\top P \begin{bmatrix} \bar{\eta}(N_p) - \eta_d(k) \\ \dot{\bar{\eta}}(N_p) \end{bmatrix},\end{aligned} \tag{5.2.21}$$

$$\text{subject to}\ \begin{cases} \text{discrete model with discretization step } \delta_{\text{att}} \\ \text{obtained from the following continuous model:} \\ \quad \dot{\bar{\eta}} = W^{-1}\bar{\omega},\ \dot{\bar{\omega}} = J^{-1}(-\bar{\omega}\times(J\bar{\omega}) + \bar{\tau}), \\ \text{state constraints as in (3.5.4):} \\ \quad \langle|\bar{\phi}(s)|, |\bar{\theta}(s)|\rangle \leq \epsilon_{\max},\ \langle|\bar{\omega}_x(s)|, |\bar{\omega}_y(s)|\rangle \leq \omega_{\max}, \\ \text{torque constraints as in (5.2.1)} \\ \quad \bar{\tau}(s) \in \mathcal{S}(T_d(k)), \\ \text{terminal stabilizing constraints as in (3.5.24)} \\ \quad \begin{bmatrix} \bar{\eta}(N_p) - \eta_d(k) \\ \dot{\bar{\eta}}(N_p) \end{bmatrix} \in \mathcal{B}^*_\varepsilon(\eta_d(k)), \\ \text{initial conditions:} \\ \quad \bar{\eta}(0) = \eta(k),\ \bar{\Omega}(0) = \Omega(k), \end{cases}$$

along the prediction horizon, i.e., for all $s \in \{0, \ldots, N_p\}$,

with $Q \in \mathbb{R}^{6\times6}, R \in \mathbb{R}^{3\times3}$ the positive definite weighting matrices as in (3.3.3) and $P \in \mathbb{R}^{6\times6}$ the terminal weighting matrix resulted from the Lyapunov equation (3.3.5). The notations with a "bar" above denote the variables employed within the prediction model as similar to their usages in (3.1.4). I.e, $\bar{\tau} = \begin{bmatrix}\bar{\tau}_\phi & \bar{\tau}_\theta & \bar{\tau}_\psi\end{bmatrix}^\top$ are the predicted torques acting on the prediction model. Furthermore, $\eta_d(k) \in \mathbb{R}^3$ represents the desired values of the three Euler angles and $T_d(k)$ is the desired thrust, all obtained from the position controller at the high level at time step k. The terminal constraint set $\mathcal{B}^*_\varepsilon(\eta_d(k))$ is constructed as in (3.5.24) by using the parameters $\tau_{\max}$ given in (5.2.2).
Solving the problem (5.2.21) provides an optimal sequence of torques: $[\bar{\tau}^*(0), \ldots, \bar{\tau}^*(N_p-1)]$ and according to the standard MPC strategy, the desired torques $\tau_d(k)$ are taken as the first element from the optimal sequence:

$$\tau_d(k) = \bar{\tau}^*(0). \tag{5.2.22}$$

5.2.1.2 Under fault functioning (with i^{th} rotor stuck)

Once the fault is detected and the information of the faulty rotor (considered in this section to be the i^{th} one) as well as its stuck speed (denoted by $\Omega_{i,f}$ as in (5.1.5)) is available, the attitude controller reconfigures to no longer attempt to control the yaw angle (recall also Figure 5.2.1). It provides only the desired values of the roll, τ_{ϕ_d}, and pitch, τ_{θ_d}, torques to track the roll, pitch angle references, ϕ_d and θ_d as in (2.4.1b)–(2.4.1c) provided by the position controller at the high level (i.e., cut off the two signals with dashed red slashes from the control scheme in Figure 5.2.1). Then, the speed calculator block provides the three reference speeds $\mathbf{\Omega}_{r,!i}$ (as defined in (5.1.12)) for the three remaining healthy rotors (those of indices $\{1,2,3,4\} \setminus \{i\}$) as follows:

$$\mathbf{\Omega}^2_{r,!i} = \widehat{M}^{-1}_{!i}\widehat{\mathfrak{u}}_d - \widehat{M}^{-1}_{!i}\widehat{M}_i\Omega^2_{i,\alpha}, \tag{5.2.23}$$

with $\widehat{\mathfrak{u}}_d \triangleq [T_d\ \tau_{\phi_d}\ \tau_{\theta_d}]^\top$ as in (5.1.8), $\widehat{M}_i \in \mathbb{R}^3$ from (5.1.9), $\widehat{M}_{!i} \in \mathbb{R}^{3\times3}$ from (5.1.10) and $\Omega_{i,\alpha}$ as in (5.1.5) the stuck speed of the faulty i^{th} rotor. The formulation (5.2.23) is obtained through the calculating steps: i) introducing the stuck speed $\Omega_{i,\alpha}$ into the nominal rotor-to-input relation (5.1.3), ii) disregarding the last row corresponding to τ_{ψ_d}, and iii) solving the resulted full-rank system of linear equations for $\mathbf{\Omega}_{r,!i}$. The calculation is illustrated in the following for the case of

the 1^{st} rotor being stuck ($i = 1$):

$$\underbrace{\begin{bmatrix} T_d \\ \tau_{\phi_d} \\ \tau_{\theta_d} \\ \cancel{\tau_{\psi_d}} \end{bmatrix}}_{\widehat{\mathfrak{u}}_d} = \underbrace{\begin{bmatrix} K_T & K_T & K_T \\ -LK_T & 0 & LK_T \\ 0 & LK_T & 0 \\ \cancel{b} & \cancel{b} & \cancel{b} \end{bmatrix}}_{\widehat{M}_{!1}} \underbrace{\begin{bmatrix} \Omega^2_{2_r} \\ \Omega^2_{3_r} \\ \Omega^2_{4_r} \end{bmatrix}}_{\mathbf{\Omega}^2_{r,!1}} + \underbrace{\begin{bmatrix} K_T \\ 0 \\ -LK_T \\ \cancel{b} \end{bmatrix}}_{\widehat{M}_1} \Omega^2_{1,\alpha}. \tag{5.2.24}$$

In order to complete the references for the four rotor speeds $\mathbf{\Omega}_r$ as in (5.1.11) (c.f. Figure 5.2.1), we simply use the stuck value $\Omega_{i,\alpha}$ from (5.1.5) as the reference for the faulty i^{th} rotor. Thus, under stuck fault, the speed calculator block provides $\mathbf{\Omega}_r$ as follows:

$$\mathbf{\Omega}^2_r = \boldsymbol{I}_{4,:i}\Omega^2_{i,\alpha} + \boldsymbol{I}_{4,!i}\mathbf{\Omega}^2_{r,!i}, \tag{5.2.25}$$

with $\boldsymbol{I}_{n,:i} \in \mathbb{R}^n$ the i^{th} column of $\boldsymbol{I}_n$ and $\boldsymbol{I}_{n,!i} \in \mathbb{R}^{n\times(n-1)}$ gathering all the other columns except the i^{th} column of the identity matrix $\boldsymbol{I}_n \in \mathbb{R}^{n\times n}$. Let us illustrate the result of the speed calculator block under stuck fault (5.2.25) for the case of the 4^{th} rotor being stuck ($i = 4$):

$$\mathbf{\Omega}^2_r = [\Omega^2_{1,r}\ \Omega^2_{2,r}\ \Omega^2_{3,r}\ \Omega^2_{4,\alpha}]^\top, \tag{5.2.26}$$

with $\Omega_{j,r}$ calculated by (5.2.23), the reference of the healthy j^{th} rotor and Ω_{4,α_4} as in (5.1.5) the stuck speed of the 4^{th} rotor.
Furthermore, by simply constraining the three remaining healthy rotor speeds references $\mathbf{\Omega}_{r,!i}$ as in (5.2.23) to stay within their magnitude bounds $[0, \Omega_{\max}]$, we arrive to the constraints on the two desired torques $(\tau_{\phi_d}, \tau_{\theta_d})$ similarly to the constraints in (5.2.1):

$$\begin{bmatrix} \tau_{\phi_d} \\ \tau_{\theta_d} \end{bmatrix} \in \mathcal{S}_i(T_d) = \left\{ \begin{bmatrix} \tau_{\phi_d} \\ \tau_{\theta_d} \end{bmatrix} \in \mathbb{R}^2 \middle| 0 \le \widehat{M}^{-1}_{!i} \begin{bmatrix} T_d \\ \tau_{\phi_d} \\ \tau_{\theta_d} \end{bmatrix} - \widehat{M}^{-1}_{!i}\widehat{M}_i\Omega^2_{i,\alpha} \le \Omega^2_{\max} \right\}, \tag{5.2.27}$$

with all the employed notations taken from (5.2.23). Even though the formulation (5.2.27) looks convoluted at the first glance, the set $\mathcal{S}_i(T_d)$ from (5.2.27) is as simple as a normal 2-dimensional polytopic region which will be illustrated through an example considering the case of the 1^{st} rotor being stuck.

Example 5.2.3 (Construction of the torque constraint set $\mathcal{S}_1(T_d)$ as in (5.2.27))**.** *When the 1^{st} rotor being stuck at the speed of $\Omega_{1,\alpha}$ as in* (5.1.5)*, the torque constraints within the set $\mathcal{S}_1(T_d)$ from* (5.2.27) *with T_d the desired thrust received from the position controller at the high level are explicitly given by:*

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \le \underbrace{\begin{bmatrix} \frac{1}{2K_T} & -\frac{1}{2LK_T} & -\frac{1}{2LK_T} \\ 0 & 0 & \frac{1}{LK_T} \\ \frac{1}{2K_T} & \frac{1}{2LK_T} & -\frac{1}{2LK_T} \end{bmatrix}}_{\widehat{M}^{-1}_{!1}} \begin{bmatrix} T_d \\ \tau_{\phi_d} \\ \tau_{\theta_d} \end{bmatrix} - \underbrace{\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}}_{\widehat{M}^{-1}_{!1}\widehat{M}_1} \Omega^2_{1,\alpha} \le \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \Omega^2_{max}, \tag{5.2.28}$$

with L, K_T as in (5.1.3) *the physical parameters and Ω_{max} the maximum rotor speed of the quadcopter system. The constraints are then clearly transformed into a 2-dimensional polytopic*

representation as follows:

$$\begin{bmatrix} \Omega_{1,\alpha}^2 - \dfrac{T_d}{2K_T} \\ -\Omega_{1,\alpha}^2 \\ \Omega_{1,\alpha}^2 - \dfrac{T_d}{2K_T} \end{bmatrix} \leq \begin{bmatrix} -\dfrac{1}{2LK_T} & -\dfrac{1}{2LK_T} \\ 0 & \dfrac{1}{LK_T} \\ \dfrac{1}{2LK_T} & -\dfrac{1}{2LK_T} \end{bmatrix} \begin{bmatrix} \tau_{\phi_d} \\ \tau_{\theta_d} \end{bmatrix} \leq \begin{bmatrix} \Omega_{max}^2 + \Omega_{1,\alpha}^2 - \dfrac{T_d}{2K_T} \\ \Omega_{max}^2 - \Omega_{1,\alpha}^2 \\ \Omega_{max}^2 + \Omega_{1,\alpha}^2 - \dfrac{T_d}{2K_T} \end{bmatrix}, \tag{5.2.29}$$

which is illustrated in Figure 5.2.3 by the blue polytopic region. □

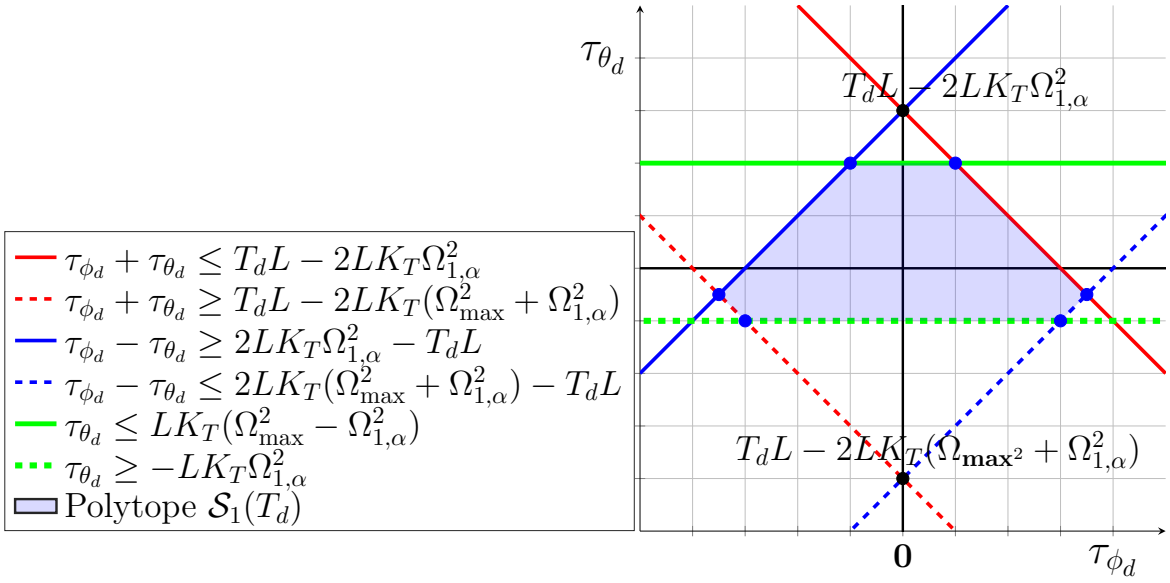


Figure 5.2.3: Illustration of the polytope $\mathcal{S}_1(T_d)$ defined in (5.2.27).

Remark 5.2.4. The polytopic region $\mathcal{S}_1(T_d)$ illustrated in Figure 5.2.3 implies a possibility to construct an admissible set of the torques $(\tau_{\phi_d}, \tau_{\theta_d})$ which remains the same under different stuck rotors (i.e., the intersection of the four polytopes $\mathcal{S}_1(T_d), \ldots, \mathcal{S}_4(T_d)$ as defined in (5.2.27)). However, bear in mind that the resulted admissible set still depends on the value of T_d and furthermore, being smaller than the original set $\mathcal{S}_i(T_d)$ (with i numbering the faulty rotor) due to the intersect operator. Hence, this direction requires more convoluted computation without any qualitative gains and even leads to more conservative results. □

After constructing the torque constraint set $\mathcal{S}_i(T_d)$ as in (5.2.27), the following proposition will summarize the saturation-avoiding results of the set and provide the predicted model of the yaw torque which is not provided by the reconfigured controller under fault, but by the open-loop behavior (5.1.3) of the quadcopter.

Proposition 5.2.5 ([Nguyen et al., 2020d]). *Let us consider the i^{th} rotor being stuck at the speed of $\Omega_{i,\alpha}$ ($i \in \{1, \ldots, 4\}$). Recalling the FTC control scheme in Figure 5.2.1, the reconfigured attitude controller constrains the desired torque values τ_{ϕ_d} and τ_{θ_d} to be within the set $\mathcal{S}_i(T_d)$ from (5.2.27) with T_d the desired thrust obtained from the position controller. The speed calculator block provides the speed references $\mathbf{\Omega}_r$ from (5.2.25). Then, the following hold:*

i) the actual values of the four rotor speeds $\mathbf{\Omega}_{i,\alpha} \in \mathbb{R}^4$ (i.e., gathering three healthy rotor speed and the stuck speed of the i^{th} rotor) as in (5.1.7) and of the thrust, roll and pitch torques

gathered in $\widehat{\mathsf{u}} \in \mathbb{R}^3$ *defined as in* (5.1.8) *(all calculated from the rotor-to-input relation* (5.1.7)*) equal their references* $(\boldsymbol{\Omega}_r, \widehat{\mathsf{u}}_d)$ *as in* (5.2.23)*:*

$$\boldsymbol{\Omega}_{i,\alpha} = \boldsymbol{\Omega}_r, \tag{5.2.30}$$

$$\widehat{\mathsf{u}} = \widehat{\mathsf{u}}_d. \tag{5.2.31}$$

ii) the uncontrolled yaw torque τ_ψ *as in* (5.1.3) *is given by the following formulation:*

$$\tau_\psi = 4(-b)^i \Omega_{i,\alpha}^2 + M_{4:} \boldsymbol{I}_{4,!i} \widehat{M}_{!i}^{-1} \widehat{\mathsf{u}}_d, \tag{5.2.32}$$

with b *the physical parameter from* (5.1.3)*,* $M_{4:}$ *the* 4^{th} *row of* M*,* $\widehat{M}_{!i}$ *from* (5.1.10)*,* $\widehat{\mathsf{u}}_d$ *from* (5.2.23) *and* $\Omega_{i,\alpha}$ *the stuck speed of the faulty* i^{th} *rotor.*

Proof. The constraints (5.2.27) on the desired torques τ_{ϕ_d} and τ_{θ_d} enforce the rotor speeds references under fault $\boldsymbol{\Omega}_r$, as in (5.1.12) (provided by the speed calculator block) to stay within their limitation $[0, \Omega_{\max}]$ from (5.1.4), hence, not being affected by the saturation. This ensures that the real speeds of the four rotors being equal to their references as stated in (5.2.30) of point i) which also includes the case of the i^{th} stuck rotor since its reference is the actual stuck speed $\Omega_{i,\alpha}$ as defined in (5.2.25). As the consequence, the real thrust and roll, pitch torques $\widehat{\mathsf{u}} = [T \; \tau_\phi \; \tau_\theta]$ partly calculated from (5.1.3) equals their desired values $\widehat{\mathsf{u}}_d$ due to the calculation in (5.2.23), completing point i) of the Proposition.
Next, the uncontrolled yaw torque τ_ψ still admits the rotor configuration from (5.1.3) of the quadcopter, hence, by applying the rotor-to-input relation (5.1.7), it is calculated by:

$$\tau_\psi = M_{4:} \boldsymbol{\Omega}_{i,\alpha}^2, \tag{5.2.33}$$

with $M_{4:}$ the 4^{th} row of the matrix M as in (5.1.3). Next, by using $\boldsymbol{\Omega}_{i,\alpha}^2 = \boldsymbol{\Omega}_r^2$ from (5.2.30) at point i), then, introducing the reference speeds $\boldsymbol{\Omega}_r^2$ from (5.2.25) to (5.2.33), we have that:

$$\begin{aligned}
\tau_\psi &= M_{4:} \left(\boldsymbol{I}_{4,:i} \Omega_{i,\alpha}^2 + \boldsymbol{I}_{4,!i} \boldsymbol{\Omega}_{r,!i}^2 \right), \\
&= M_{4:} \left(\boldsymbol{I}_{4,:i} \Omega_{i,\alpha}^2 + \boldsymbol{I}_{4,!i} \left(\widehat{M}_{!i}^{-1} \widehat{\mathsf{u}}_d - \widehat{M}_{!i}^{-1} \widehat{M}_i \Omega_{i,\alpha}^2 \right) \right), \text{(by using (5.2.23))} \\
&= M_{4:} \left(\boldsymbol{I}_{4,:i} - \boldsymbol{I}_{4,!i} \widehat{M}_{!i}^{-1} \widehat{M}_i \right) \Omega_{i,\alpha}^2 + M_{4:} \boldsymbol{I}_{4,!i} \widehat{M}_{!i}^{-1} \widehat{\mathsf{u}}_d, \\
&= 4(-b)^i \Omega_{i,\alpha}^2 + M_{4:} \boldsymbol{I}_{4,!i} \widehat{M}_{!i}^{-1} \widehat{\mathsf{u}}_d,
\end{aligned} \tag{5.2.34}$$

in which $M_{4:} \left(\boldsymbol{I}_{4,:i} - \boldsymbol{I}_{4,!i} \widehat{M}_{!i}^{-1} \widehat{M}_i \right) = 4(-b)^i$ due to the construction of M in (5.1.3). This also completes the proof. □

The representation of the yaw torque τ_ψ given in (5.2.32) may causes some implementation difficulties since the notation changes depending on the number of the stuck rotor. Thus, we show their explicit formulations in the following:

$$\begin{bmatrix} \tau_\psi \text{ under fault of } 1^{st} \text{ rotor} \\ \tau_\psi \text{ under fault of } 2^{nd} \text{ rotor} \\ \tau_\psi \text{ under fault of } 3^{rd} \text{ rotor} \\ \tau_\psi \text{ under fault of } 4^{th} \text{ rotor} \end{bmatrix} = \begin{bmatrix} -b\Omega_{1,\alpha}^2 \\ b\Omega_{2,\alpha}^2 \\ -b\Omega_{3,\alpha}^2 \\ b\Omega_{4,\alpha}^2 \end{bmatrix} + \begin{bmatrix} \frac{b}{K_T} & 0 & -\frac{2b}{LK_T} \\ -\frac{b}{K_T} & \frac{2b}{LK_T} & 0 \\ \frac{b}{K_T} & 0 & \frac{2b}{LK_T} \\ -\frac{b}{K_T} & -\frac{2b}{LK_T} & 0 \end{bmatrix} \begin{bmatrix} T_d \\ \tau_{\phi_d} \\ \tau_{\theta_d} \end{bmatrix}, \tag{5.2.35}$$

with b, K_T, L the parameters from (5.1.3), $\Omega_{i,\alpha}$ the stuck speed of the faulty i^{th} rotor (i.e., the notations do not constrain the rotors to remain stuck at the same speed) and $(T_d, \tau_{\phi_d}, \tau_{\theta_d})$ as in (5.2.32) the desired inputs.
Next, we present the design of the reconfigured NMPC attitude controller. Since we control only the two torques (τ_ϕ, τ_θ) as in (5.1.3) (i.e., equal their desired values as shown in (5.2.31)), the yaw angle ψ and its derivatives $\dot{\psi}$ is out of control. Thus, employing the terminal constraint set $\mathcal{B}^*(\eta_d, \varepsilon)$ as in (3.5.24) which requires the control of the two aforementioned yaw variables becomes infeasible. Therefore, in order to reduce the complexity of the design problem, we decide not to use the terminal stabilizing constraints within the reconfigured NMPC attitude controller but to employ a practical "long-enough" prediction horizon instead. The value is found by conducting various simulation tests.

NMPC design for the reconfigured attitude controller under fault of the i^{th} stuck rotor:
The NMPC optimization problem employed within the reconfigured attitude controller running at the sampling time δ_{att} is given in its discrete form at time step k as follows:

$$\min_{\widehat{\bar{\tau}}(\cdot)} \sum_{s=0}^{N_{p_f}} \Big(\big(\widehat{\bar{\eta}}(s) - \widehat{\eta}_d(k)\big)^\top Q_{\widehat{\eta}} \ \big(\widehat{\bar{\eta}}(s) - \widehat{\eta}_d(k)\big) + \widehat{\bar{\omega}}^\top(s) Q_{\widehat{\omega}} \ \widehat{\bar{\omega}}(s) + \widehat{\bar{\tau}}^\top(s) R_{\widehat{\tau}} \ \widehat{\bar{\tau}}(s) \Big), \tag{5.2.36}$$

$$\text{subject to} \begin{cases} \text{discrete model with discretization step } \delta_{\text{att}} \\ \text{obtained from the following continuous model:} \\ \quad \dot{\bar{\eta}} = W^{-1}\bar{\omega}, \ \dot{\bar{\omega}} = J^{-1}(-\bar{\omega} \times (J\bar{\omega}) + \bar{\tau}), \\ \text{yaw torque model as in (5.2.32):} \\ \quad \bar{\tau}_\psi(s) = 4(-b)^i \Omega_{i,\alpha}^2 + M_{4:}\boldsymbol{I}_{4,!i}\widehat{M}_{!i}^{-1} \Big[T_d(k) \ \bar{\tau}_\phi(s) \ \bar{\tau}_\theta(s)\Big]^\top, \\ \text{state constraints as in (3.5.4):} \\ \quad \langle |\bar{\phi}(s)|, |\bar{\theta}(s)| \rangle \leq \epsilon_{\max}, \ \langle |\bar{\omega}_x(s)|, |\bar{\omega}_y(s)| \rangle \leq \omega_{\max}, \\ \text{torque constraints as in (5.2.27):} \\ \quad \widehat{\bar{\tau}}(s) \in \mathcal{S}_i(T_d(k)), \\ \text{initial conditions:} \\ \quad \bar{\eta}(0) = \eta(k), \ \bar{\Omega}(0) = \Omega(k), \end{cases}$$

along the prediction horizon, i.e., for all $s \in \{0, \ldots, N_{p_f}\}$,

with $Q_{\widehat{\eta}}, Q_{\widehat{\omega}}, R_{\widehat{\tau}} \in \mathbb{R}^{2\times 2}$ the positive definite weighting matrices. The notation with a "hat" above is defined as in (5.1.8), e.g., $\widehat{\bar{\eta}} = \big[\bar{\phi} \ \bar{\theta}\big]^\top$, $\widehat{\bar{\omega}} = \big[\bar{\omega}_x \ \bar{\omega}_y\big]^\top$, $\widehat{\bar{\tau}} = \big[\bar{\tau}_\phi \ \bar{\tau}_\theta\big]^\top$ while the notations with a "bar" above denote the variables employed within the prediction model as in (5.2.21). The prediction horizon N_{p_f} is also chosen to be longer than N_p as in (5.2.19) under the nominal functioning.
The reconfigured attitude controller provides the desired torques $\widehat{\tau}_d = \big[\tau_{\phi_d}, \tau_{\theta_d}\big]^\top$ taken as the first element among the optimal solution sequence $[\widehat{\bar{\tau}}^*(0), \ldots, \widehat{\bar{\tau}}^*(N_p)]$:

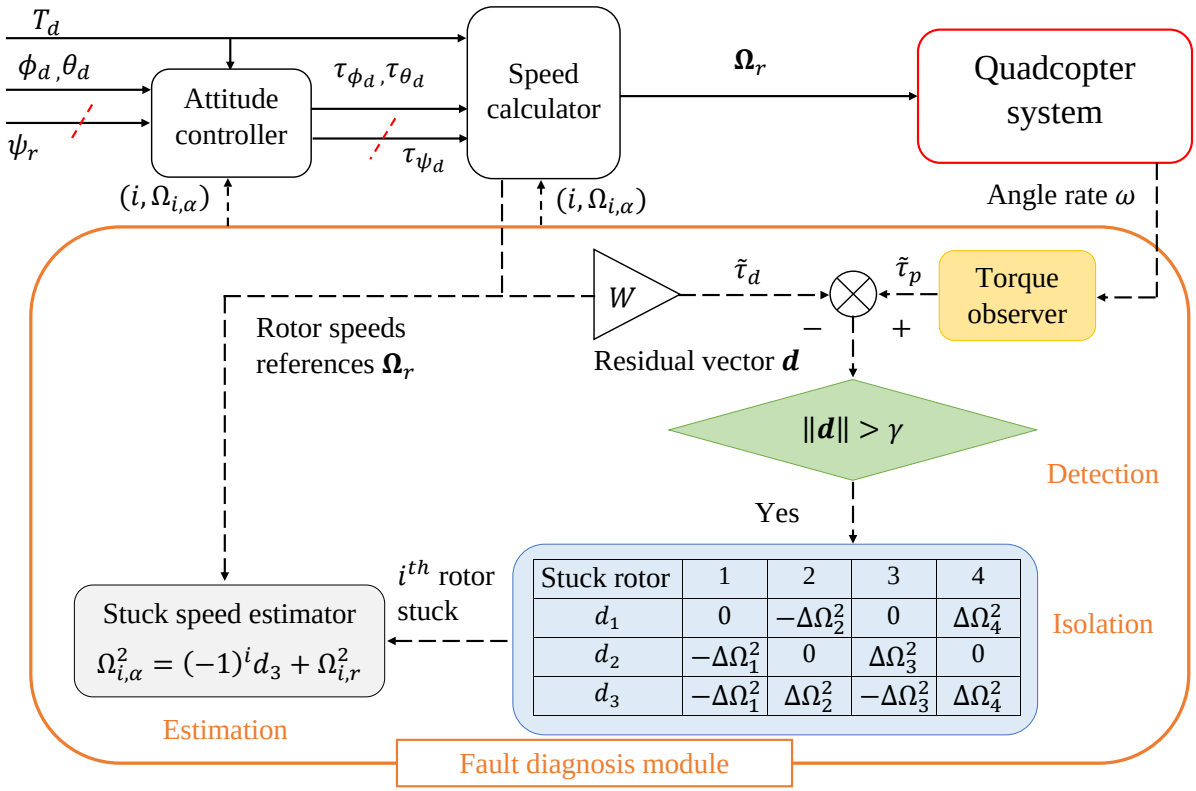
$$\widehat{\tau}_d = \widehat{\bar{\tau}}^*(0). \tag{5.2.37}$$

Note that, by constraining the predicted torques $\widehat{\bar{\tau}}(s)$ along the prediction horizon N_p to stay within the set $\mathcal{S}_i(T_d(k))$, Proposition 5.2.5 is validated. Thus, the predicted yaw torque $\bar{\tau}_\psi(s)$

employed within the NMPC design (5.2.36) can make use of the formulation (5.2.32). Furthermore, Proposition 5.2.5 also provides that the actual values of the thrust and the roll, pitch torques $\widehat{\mathrm{u}} = \begin{bmatrix} T & \tau_\phi & \tau_\theta \end{bmatrix}$ as in (5.2.31) equal with their desired references $\widehat{\mathrm{u}}_d = \begin{bmatrix} T_d & \tau_{\phi_d} & \tau_{\theta_d} \end{bmatrix}$ with T_d provided by the position controller and $(\tau_{\phi_d}, \tau_{\theta_d})$ obtained from the reconfigured attitude controller (5.2.36). The actual roll, pitch angles (ϕ, θ) as in (3.5.2) asymptotically track their references (ϕ_d, θ_d) provided by the position controller as in (2.4.1b)–(2.4.1c). These ensure the stability of the whole hierarchical control scheme.
Both the nominal and reconfigured attitude controllers given in (5.2.21) and (5.2.36) aim to enforce the actual rotor speeds to equal their references even under a single stuck rotor. Making use of these properties, in the next section, we introduce a design of the fault diagnosis module operating at the low control level as shown in Figure 5.2.4.

5.2.2 Fault diagnosis module

The proposed fault diagnosis module calculates a residual vector d which describes the differences between the actual normalized torque $\widetilde{\tau}$ from (5.1.3) (i.e., estimated by an input observer based on the measurement of the angle rate ω) and its desired value $\widetilde{\tau}_d$ - obtained by using the four desired rotor speeds $\mathbf{\Omega}_r$ as in (5.2.19). By analyzing the residual result, the module can detect the stuck fault, identify the i^{th} stuck rotor and estimate its stuck speed $\Omega_{i,\alpha}$.



Stuck rotor	1	2	3	4
d_1	0	$-\Delta\Omega_2^2$	0	$\Delta\Omega_4^2$
d_2	$-\Delta\Omega_1^2$	0	$\Delta\Omega_3^2$	0
d_3	$-\Delta\Omega_1^2$	$\Delta\Omega_2^2$	$-\Delta\Omega_3^2$	$\Delta\Omega_4^2$

Figure 5.2.4: Fault diagnosis module for detecting the stuck fault of the quadcopter system.

Torque normalization:

Regarding the rotor-to-input relation as in (5.1.3), the torques $\tau = [\tau_\phi \ \tau_\theta \ \tau_\psi]^\top$ are calculated as the differences between the rotor speeds multiplying with some constant parameters assumed

known. In order to simplify the analysis presented in the followings, we define the normalized torques which are actually the torques τ divided by the parameters and depend only on the actual rotor speeds:

$$\widetilde{\tau} = \begin{cases} W\mathbf{\Omega}^2, & \text{under nominal condition,} \\ W\mathbf{\Omega}_{i,\alpha}^2, & \text{if the } i^{th} \text{ rotor stuck,} \end{cases} \tag{5.2.38}$$

with $\mathbf{\Omega}$, $\mathbf{\Omega}_{i,\alpha}$ from (5.1.7) and the matrix $W \in \mathbb{R}^{3\times 4}$ defined as:

$$W = \begin{bmatrix} 0 & -1 & 0 & 1 \\ -1 & 0 & 1 & 0 \\ -1 & 1 & -1 & 1 \end{bmatrix}. \tag{5.2.39}$$

Note that the explicit relation between $\widetilde{\tau}$ and τ is given as follows:

$$\widetilde{\tau} = \text{diag}\left\{\frac{1}{LK_T}, \frac{1}{LK_T}, \frac{1}{b}\right\}\tau, \tag{5.2.40}$$

with L, K_T, b the physical parameters from (5.1.3).
Next, by a similar way, we also construct the normalized desired torques $\widetilde{\tau}_d$ as follows:

$$\widetilde{\tau}_d = W\mathbf{\Omega}_r^2, \tag{5.2.41}$$

with $\mathbf{\Omega}_r$ the rotor speeds references obtained by (5.2.19) under nominal functioning and by (5.2.25) under fault of the i^{th} stuck rotor. Then, it is straightforward to see that if the residual vector is taken as the difference between $\widetilde{\tau}$ and $\widetilde{\tau}_d$ as in (5.2.38)–(5.2.41), it obviously shows the differences between the real rotor speeds $\mathbf{\Omega}$ and their references $\mathbf{\Omega}_r$ and hence, any mismatches on their speed tracking due to the stuck fault can be easily detected. However, we do not have any direct measurement on the rotor speeds, hence, an estimation of $\widetilde{\tau}$ is essential to obtain. This is done through the rotation dynamics (3.5.2b) and the available information of the angle rate ω.

Torque observer (yellow block in Figure 5.2.4):

The observer makes use of the Euler discretization of the rotation dynamics $J\dot{\omega}+\omega\times(J\omega) = \tau$ as in (3.5.2b). By applying the backward Euler method to the rotation dynamics given in (3.5.2b), we have that:

$$\tau_p(k) = J\left(\frac{\omega(k) - \omega(k-1)}{\delta_{\text{FDI}}}\right) + \omega(k) \times (J\omega(k)), \tag{5.2.42}$$

with $\tau_p(k)$ the estimated torque at time step k, $\omega(k)$ the angle rate, δ_{FDI} the sampling time of the FDI module (can be chosen smaller than the sampling time of the attitude controller δ_{att} as in (5.2.36) to enhance the accuracy, but being limited by the feedback rate of ω). It has been shown in [El Assoudi et al., 2002] that the construction in (5.2.42) yields a stable observer in the case where the errors between the actual continuous states and the discrete ones can be neglected. Then, the normalized estimated torque $\widetilde{\tau}_p$ is calculated by dividing each element of τ_p from (5.2.42) by their corresponding constant parameters as similar to (5.2.40):

$$\widetilde{\tau}_p(k) = \text{diag}\left\{\frac{1}{LK_T}, \frac{1}{LK_T}, \frac{1}{b}\right\}\tau_p(k), \tag{5.2.43}$$

with L, K_T, b the physical parameters from (5.1.3).

Remark 5.2.6. The torque observer as in (5.2.42)–(5.2.43) requires a high-accuracy-and-frequency feedback of the angle rate ω and a good modeling of the quadcopter system. The first requirement mostly depends on the quality of the gyroscope installed within the platform. According to the latest updates in [Passaro et al., 2017], the recent gyroscope techonologies can achieve a very high precision, e.g., $0.08°/\sqrt{\text{hour}}$ (degree per root hour) Angular Random Walk with the frequency of up to 2000 Hz [IMU, 2019]. Furthermore, the angle rates values can be re-filtered before introducing them into (5.2.42) as detailed in [Kownacki, 2011, Jiang et al., 2012].
Next, a good modeling is possible to obtain but requires a sophisticated system identification process [Doniselli et al., 2002, Bottasso et al., 2009, Previati et al., 2009, Doniselli et al., 2002],. For example, the work in [Bottasso et al., 2009] considers the same rotation dynamics as in (3.5.2b) and proposes a method estimating the values of J with $\pm 5\%$ of accuracy. Also, the aerodynamics parameters K_T, b as in (5.2.43) can be measured following the procedure given in [Förster, 2015, Luis and Ny, 2016]. □

Residual vector and its functioning:

Recalling the FDI module scheme in Figure 5.2.1, the residual vector $\boldsymbol{d} \triangleq \begin{bmatrix} d_1 & d_2 & d_3 \end{bmatrix}^\top$ is simply taken as follows:

$$\boldsymbol{d} = \widetilde{\tau}_p - \widetilde{\tau}_d, \tag{5.2.44}$$

with $\widetilde{\tau}_p$, the estimated normalized torque as in (5.2.43) and $\widetilde{\tau}_d$, the normalized desired torque as in (5.2.41), both taken at the same time instants. Note that, formulation (5.2.44) should be given in the discrete domain due to the usage of $\widetilde{\tau}_p(k)$ in (5.2.43), i.e., $\boldsymbol{d}(k) = \widetilde{\tau}_p(k) - \widetilde{\tau}_d(k)$. However, we hide the time step k from (5.2.44) since taking two signals at the same time instant is clear from the context and furthermore, it simplifies the formulations presented in the following.

Proposition 5.2.7 (Fault detection [Nguyen et al., 2020d])**.** *Let us consider the residual vector* $\boldsymbol{d} \triangleq \begin{bmatrix} d_1 & d_2 & d_3 \end{bmatrix}^\top$ *calculated as in* (5.2.44) *and the quadcopter system controlled by the control scheme detailed in Section 5.2.1. The followings hold:*

1) If the attitude controller given in Section 5.2.1 is functioning in the appropriate mode, i.e., nominal mode (c.f. Section 5.2.1.1) under nominal case and under-fault mode (c.f. Section 5.2.1.2) corresponding to the right scenario (i^{th} rotor being stuck at $\Omega_{i,\alpha}$), then the norm of the residual vector $\|\boldsymbol{d}\|$ varies around zero:

$$\|\boldsymbol{d}\| \approx 0. \tag{5.2.45}$$

2) If the real rotor speeds are not tracking their references, then, the norm of the residual vector $\|\boldsymbol{d}\|$ varies around a non-zero value. More precisely, we consider two following scenarios:

2a) If the attitude controller given in Section 5.2.1 is functioning in the nominal mode (c.f. Section 5.2.1.1) and the i^{th} rotor is stuck at the speed of $\Omega_{i,\alpha}$, then we have that:

$$\|\boldsymbol{d}\| \approx \sqrt{2}|\Omega_{i,\alpha}^2 - \Omega_{i,r}^2|, \tag{5.2.46}$$

with $\Omega_{i,r}$ the speed reference of the i^{th} rotor.

2b) If the attitude controller given in Section 5.2.1 is functioning in under-fault mode (c.f. Section 5.2.1.2) corresponding to the right i^{th} stuck rotor but with the wrong stuck speed Ω_{i,α_w} in comparison with the actual stuck speed $\Omega_{i,\alpha}$, then:

$$\|\boldsymbol{d}\| \approx \sqrt{2}|\Omega_{i,\alpha}^2 - \Omega_{i,\alpha_w}^2|. \tag{5.2.47}$$

Proof. We construct the proof by assuming that the torques estimator as in (5.2.42)–(5.2.43) works properly and provides that the values of the estimated normalized torque $\widetilde{\tau}_p$ from (5.2.43) which vary around the actual normalized torque value $\widetilde{\tau}$ from (5.2.40). Then, we have that:

$$\boldsymbol{d} \approx \widetilde{\tau} - \widetilde{\tau}_d = \begin{cases} W(\boldsymbol{\Omega}^2 - \boldsymbol{\Omega}_r^2), & \text{under nominal functioning,} \\ W(\boldsymbol{\Omega}_{i,\alpha}^2 - \boldsymbol{\Omega}_r^2), & \text{with the } i^{th} \text{ rotor being stuck,} \end{cases} \tag{5.2.48}$$

with W from (5.2.39), $\boldsymbol{\Omega}_r$ the reference rotor speeds and $\boldsymbol{\Omega}$, $\boldsymbol{\Omega}_{i,\alpha}$ the actual rotor speeds under nominal and faulty cases.
At first, for point 1), under both considered scenarios, the attitude controller ensures that the actual rotor speeds equal their references, i.e., $\boldsymbol{\Omega} = \boldsymbol{\Omega}_r$ as in (5.2.20) for the nominal functioning case and $\boldsymbol{\Omega}_{i,\alpha}^2 = \boldsymbol{\Omega}_r^2$ as in (5.2.30) for the under-fault case (i.e., under fault, we provide the stuck speed $\Omega_{i,\alpha}$ as the reference for the i^{th} stuck rotor). Therefore, from (5.2.48), we obtain that:

$$\boldsymbol{d} \approx \boldsymbol{0}, \tag{5.2.49}$$

which also validates (5.2.45).
Next, regarding both scenarios considered at point 2), their common problem is that the i^{th} faulty rotor ($i \in \{1, \dots, 4\}$) becomes stuck at the speed of $\Omega_{i,\alpha}$ as defined in (5.1.5) and does not follow its reference, given by the nominal reference $\Omega_{i,r}$ as in (5.2.46) and the constant reference (i.e., incorrect stuck speed) Ω_{i,α_w} as in (5.2.47). Hence, the norm of the residual vector $\boldsymbol{d}$ varies around a non-zero value:

$$\boldsymbol{d} \approx W_i \Delta\Omega_i^2, \tag{5.2.50}$$

in which, W_i is the i^{th} column of the matrix W from (5.2.39). The term $\Delta\Omega_i^2$ describing the speed tracking mismatch of the i^{th} rotor is defined as follows:

$$\Delta\Omega_i^2 = \begin{cases} \Omega_{i,\alpha}^2 - \Omega_{i,r}^2, & \text{under point 2a,} \\ \Omega_{i,\alpha}^2 - \Omega_{i,\alpha_w}^2, & \text{under point 2b,} \end{cases} \tag{5.2.51}$$

with $\Omega_{i,\alpha}$, $\Omega_{i,r}$ and Ω_{i,α_w} given as in (5.2.46)–(5.2.47).
Due to the construction of W from (5.2.39), $\|W_i\| = \sqrt{2}, \forall i \in \{1, \dots, 4\}$ which further provides $\|\boldsymbol{d}\| \approx \|W_i\||\Delta\Omega_i^2| = \sqrt{2}|\Delta\Omega_i^2|$. This validates both (5.2.46)–(5.2.47) for the cases 2a) and 2b), hence, completing the proof. □

From Proposition 5.2.7, the two faulty scenarios 2a (5.2.46) and 2b (5.2.47) can be distinguished from the point 1 (5.2.45) and be detected by checking the value of $\|\boldsymbol{d}\|$ over a threshold $\gamma \in \mathbb{R}_+$ (c.f. Figure 5.2.4):

$$\|\boldsymbol{d}\| > \gamma. \tag{5.2.52}$$

Designing the value of γ requires to take into account the realistic speed speed tracking delays of the rotors, the mismatch on the measurement of ω and the noises caused by the torque observer from (5.2.42) and, not in the least, various uncertainties affecting the system as detailed in [Hasan and Johansen, 2018]. A feasible design for γ can be taken as:

$$\gamma = \sqrt{2}\beta\Omega_{\max}^2, \tag{5.2.53}$$

with $\beta \in (0, 1)$ the tuning parameter describing the ratio of the acceptable tracking error (e.g. $|\Omega_{i,\alpha}^2 - \Omega_{i,r}^2|$ as in (5.2.46)) to the maximum squared speed $\Omega_{\max}^2$ as in (5.1.4).
After detecting the two faulty scenarios 2a (5.2.46) and 2b (5.2.47), the following proposition details how to isolate the faulty rotor and estimate its stuck speed. The work makes use of the

formulation (5.2.48) under the same assumption of the torque observer (5.2.42)–(5.2.43) properly functioning. We will show that the i^{th} stuck rotor can be isolated by comparing the values of the three elements of the residual vector $\boldsymbol{d} = \begin{bmatrix} d_1 & d_2 & d_3 \end{bmatrix}^\top$ as in (5.2.44) and its stuck speed can be obtained as a consequence.

Proposition 5.2.8 (Fault isolation and estimation [Nguyen et al., 2020d])**.** *Let us consider two scenarios 2a* (5.2.46) *and 2b* (5.2.47) *from Proposition 5.2.7. After detecting the faults, the i^{th} faulty rotor can be isolated (i.e., re-checked for scenario 2b) by using the look-up Table 5.2.1 in which the residual vector $\boldsymbol{d} = \begin{bmatrix} d_1 & d_2 & d_3 \end{bmatrix}$ is from* (5.2.44) *and the notation $\Delta\Omega_i^2$ is defined in* (5.2.51)*.*

Stuck rotor	1	2	3	4
d_1	0	$-\Delta\Omega_2^2$	0	$\Delta\Omega_4^2$
d_2	$-\Delta\Omega_1^2$	0	$\Delta\Omega_3^2$	0
d_3	$-\Delta\Omega_1^2$	$\Delta\Omega_2^2$	$-\Delta\Omega_3^2$	$\Delta\Omega_4^2$

Table 5.2.1: Stuck rotor identification.

Next, after obtaining the index i, the actual stuck speed $\Omega_{i,\alpha}$ is estimated by:

$$\Omega_{i,\alpha}^2 = \begin{cases} (-1)^i d_3 + \Omega_{i,r}^2, & \text{under point 2a of Proposition 5.2.7,} \\ (-1)^i d_3 + \Omega_{i,\alpha_w}^2, & \text{under point 2b of Proposition 5.2.7,} \end{cases} \tag{5.2.54}$$

with $\Omega_{i,\alpha}$ the actual stuck speed of the i^{th} rotor and $\Omega_{i,r}$, Ω_{i,α_w} two references under two cases 2a (5.2.46) *and 2b* (5.2.47) *of Proposition 5.2.7.*

Proof. The look-up Table 5.2.1 is constructed by using the relation $\boldsymbol{d} \approx W_i \Delta\Omega_i^2$ as in (5.2.50) with W_i the i^{th} column of W from (5.2.39). Then, by generalizing the last row of Table 5.2.1, we arrive to:

$$d_3 \approx (-1)^i \Delta\Omega_i^2, \tag{5.2.55}$$

with $\Delta\Omega_i^2$ as in (5.2.51), which further leads to the use as in (5.2.54), completing the proof. □

Note that, due to realistic noises and implementation mismatches, it is not required to check if the values of the three elements of the residual vector $\boldsymbol{d} = \begin{bmatrix} d_1 & d_2 & d_3 \end{bmatrix}$ as in (5.2.44) follow exactly the indications given in Table 5.2.1, e.g., to check the condition $d_2 = d_3$ for identifying if the 1^{st} rotor being stuck. The isolation algorithm can be relaxed by only checking the signs of the elements (d_1, d_2, d_3). Hereinafter, we summarize all the steps to establish the whole process of the proposed diagnosis module.

Procedure 5.2.9 (Fault detection, isolation and estimation procedure [Nguyen et al., 2020d])**.**

1. *Fault detection:*
   - *Check $\|\boldsymbol{d}\| > \gamma$ as in* (5.2.50)*. If Yes, continue to Step 2. If No, come back to Step 1.*
2. *Fault isolation:*
   - *Find the the element d_j having the largest magnitudes among (d_1, d_2). (since according to Table 5.2.1, the value of d_3 is always affected directly by the fault).*
   - *Check the signs of d_j and d_3 according to Table 5.2.1 in order to isolate the i^{th} stuck rotor. E.g., if $j = 1$ and $\text{sign}(d_j) = \text{sign}(d_3)$, then, $i = 4$.*

3. Fault estimation:
- *Calculate the actual stuck speed of the i^{th} faulty rotor by using* (5.2.54).

5.3 Trajectory tracking under a stuck fault event

In this section, we present the simulation validations of the proposed FTC scheme given in Figure 5.2.1 under a trajectory tracking scenario [Nguyen et al., 2020d]. We employ again the model of the Crazyflie 2.0 quadcopter system given in [Luis and Ny, 2016] as already introduced in Section 2.2.6.1. The only difference is that the input constraints as in (2.2.49) are replaced by the saturation constraints on the rotor speeds from (5.1.4):

$$\Omega_{\max} = 22000 \ rpm. \tag{5.3.1}$$

Also, the values of the physical parameters L, K_T, b employed in (5.1.3) are given by:

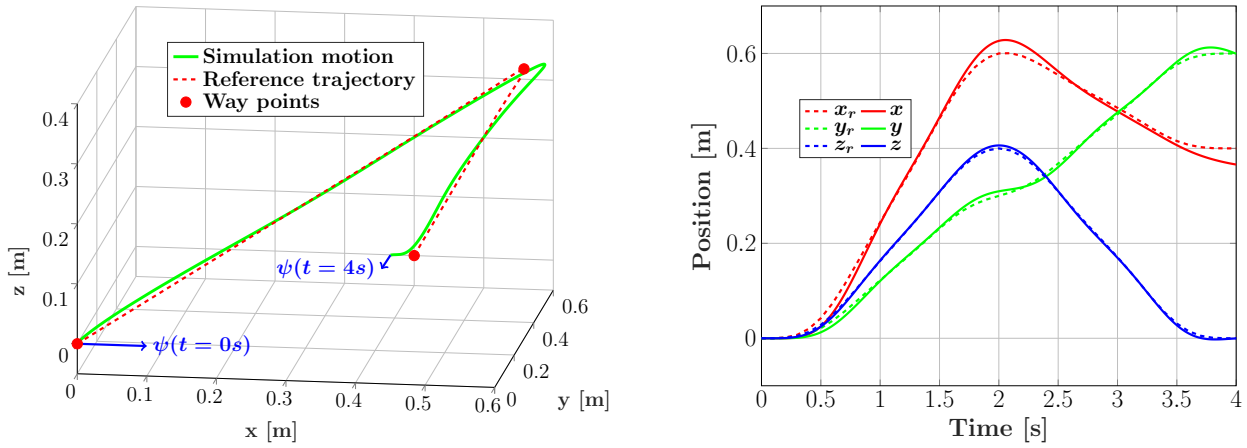
$$L = 0.065 \ m, \ K_T = 3.16 \times 10^{-10} \ N/rpm^2, \ b = 7.94 \times 10^{-12} \ Nm/rpm^2. \tag{5.3.2}$$

The implementation is as follows. The FTC controller (including the two control layers and the fault diagnosis module as in Figure 5.2.1) is implemented in Python in which, the NMPC controllers (5.2.21) and (5.2.36) employ the prediction model discretized by using the standard forward Euler method and make use of the solver IPOPT [Wächter and Biegler, 2006]. Then, the control signals are given to a model of the quadcopter system as in (2.1.10) simulated through an Ordinary Differential Equation (ODE) solver. Therefore, we can benefit from the short solving time of Python and also the simulation accuracy of the ODE solver.
The reference trajectory is generated by using the B-spline parametrization of Section 2.2.3 and by minimizing the length curve while simultaneously satisfying the system constraints, similarly to the results given in Section 2.2.6.1. The reference trajectory $\xi_r(t) = \begin{bmatrix} x_r(t) & y_r(t) & z_r(t) \end{bmatrix}^\top$ (plotted in dashed red line in Figure 5.3.1a) passes through three a priori given way-points (given by red dots in Figure 5.3.1a):

$$\xi_r(0) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top, \ \xi_r(2) = \begin{bmatrix} 0.6 & 0.3 & 0.4 \end{bmatrix}^\top, \ \xi_r(4) = \begin{bmatrix} 0.4 & 0.6 & 0 \end{bmatrix}^\top, \tag{5.3.3}$$

while the yaw angle reference is fixed at zero along the simulation horizon, i.e. $\psi_r(t) = 0$.

The position controller at the high control level employs the feedback linearization law given in (2.4.1) and also shares the same parameters as shown in Table 2.5.1. It runs at the frequency of 10 Hz (in comparison with 100 Hz for the attitude controller at the low level). The rate difference is clearly observed in Figure 5.3.4a for the desired thrust T_d (red line) and in Figure 5.3.4b for the desired roll, pitch angles (solid red and green lines, respectively). We can see that the desired angles are kept as piece-wise constant references for the attitude controller at the low control level to follow.
Next, Table 5.3.1 gives the tuning parameters of the two NMPC attitude controllers: under the nominal case as in (5.2.19) and under the stuck fault as in (5.2.36). We emphasize that the values of $T_{d_{\min}}$, $T_{d_{\max}}$ and $\tau_{\max}$ as in (5.2.2) are obtained by using the values of (U_x, U_y, U_z) given in Table 2.5.1 and the other parameters regarding the nominal NMPC controller follow the design procedure given in Section 3.5. For the under-fault mode, the chosen prediction horizon has $N_{p_f} = 8$ steps as employed in (5.2.36) and is enough to stabilize the attitude through the various simulation tests. The NMPC sampling time δ_{att} defined as in (5.2.19), (5.2.36) and the observer sampling time δ_{FDI} defined as in (5.2.42) are both fixed at 0.01 seconds.



(a) 3D trajectory tracking results.

(b) Position tracking results.

Figure 5.3.1: Trajectory tracking result of the proposed FTC scheme under stuck rotor fault.

We provide simulation results for tracking the reference trajectory given in Figure 5.3.1a using the hierarchical FTC controller detailed in Section 5.2 under the following fault scenario:
- *From* 0 *to* 2.5 *seconds:* nominal functioning;
- *From* 2.5 *to* 3 *seconds:* the 4^{th} rotor is stuck at its previous rotating speed (c.f. Fig. 5.3.2: $\omega_{4,\alpha_1} = \alpha_1 \Omega_{\max}$ with $\alpha_1 = 0.66$);
- *From* 3 *to* 4 *seconds:* the 4^{th} rotor is stuck at another speed of $\omega_{4,\alpha_2} = \alpha_2 \Omega_{\max}$ with $\alpha_2 = 0.68$.

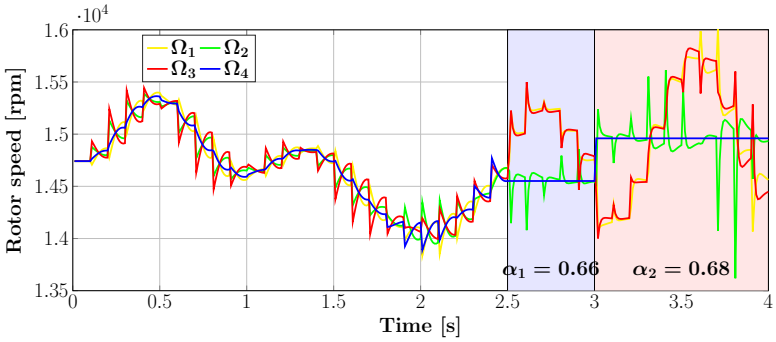


Figure 5.3.2: Rotor speeds values under stuck fault simulation.

Firstly, the proposed FTC scheme ensures the trajectory tracking capability of the quadcopter system. In Figure 5.3.1a, the simulation motion (plotted in solid green line) tracks well the reference trajectory (dashed red line) while the position converges over time along the three axes, as shown in Figure 5.3.1b. For the reader to easily understand the control challenge, we provide the results of the four rotor speeds in Figure 5.3.2. Under the nominal operation from 0 to 2.5 seconds, all the four rotor speeds are varying over time. Then, from 2.5 to 3 seconds (plotted within the blue rectangle), the 4^{th} rotor is stuck at $\alpha_1 = 0.66$ (i.e., being stuck at the previous speed) and from 3 to 4 seconds (within the red rectangle), at $\alpha_2 = 0.68$ which is illustrated by the blue line remaining constant at two different values. Note that, even though the change is only 2% but it is in the percentage of $\Omega_{\max} = 22000$ *rpm* which is actually large for the considered scenarios as can be seen by the clear jump at $t = 3$ seconds of the blue line in Figure 5.3.2. Also, the four rotor speeds clearly respect the maximum speed $\Omega_{\max}$ from (5.3.1) which proves the validation of the torque constraint sets $\mathcal{S}(T_d)$ as in (5.2.1) under nominal case and $\mathcal{S}_4(T_d)$ as in (5.2.27) when the 4^{th} rotor is faulty.

The fault diagnosis module detailed in Section 5.2.2 succeeds in detecting the stuck faults within two sampling time periods (0.02 seconds) by using the threshold γ as in (5.2.52) with $\beta = 10^{-2}$ as given in Figure 5.3.3. This value represents an acceptable speed tracking mismatch of 10% of the maximum rotor speed $\Omega_{\max}$ given in (5.3.1). Note that, a smaller value of γ can be employed but may result in excessive sensitivity (i.e., false alarms). In Figure 5.3.3, we show that, when the first stuck fault happens at $t = 2.5$ seconds for the 4^{th} rotor, both the values of d_1 (red line) and d_3 (blue line) from the residual vector $\boldsymbol{d} = [d_1\ d_2\ d_3]^\top$ as in (5.2.44) dramatically increase exceeding the threshold value which clearly indicate that $\|\boldsymbol{d}\| > \gamma$, correctly serves its role as condition for accurate fault detection (5.2.50). Furthermore, both d_1 and d_3 share the same sign, i.e., having negative values which clearly point out the 4^{th} rotor being stuck according to the fault isolation scheme given in Table 5.2.1 and, furthermore, allow us to estimate the stuck speed as in (5.2.54). After detecting the first fault, the attitude controller is reconfigured into the under-fault mode as detailed in Section 5.2.1.2 in which, we no longer control the yaw angle as can be seen from the yaw angle trajectory (blue dashed line) starting to decrease after $t = 2.5$ seconds in Figure 5.3.4b. Using the reconfigured attitude controller (5.2.36) in the appropriate mode (i.e., for the 4^{th} stuck rotor) allows the residual vector to quickly converge towards zero as can be seen from $t = 2.5$ to 3 seconds in Figure 5.3.3 until the second fault happens right after that. The stuck speed of the 4^{th} rotor changes from Ω_{4,α_1} ($\alpha_1 = 0.66$) to Ω_{4,α_2} ($\alpha_1 = 0.68$) as shown in Figure 5.3.2. Once again, the values of d_1 and d_3 from Figure 5.3.3 exceed the threshold but this time, their values are positive which allows us to validate again the 4^{th} rotor being stuck and then, estimate the new stuck speed. Right after the attitude controller (5.2.36) is reconfigured with the correct stuck speed Ω_{4,α_2}, the residual values converges back to zero. As a part of the fault diagnosis module, the result of the torque observer (5.2.42) is given in Figure 5.3.4d. We provide only the estimated value of the roll torque τ_{ϕ_p} as in (5.2.42) plotted as a solid green line since the estimated results are noisy due to the usage of the backward Euler method in (5.2.42) (can be seen from the enlarging circle) which significantly reduce the clarity of the figure.

Next, Figures 5.3.4b–5.3.4d prove the capability of the FTC attitude controller under both nominal condition (i.e., its ability to track well the three angle references until $t = 2.5$ seconds shown in Figure 5.3.4b) and under fault (i.e., to renounce the control of the yaw angle plotted in dashed blue line from $t = 2.5$ seconds but still track the roll, pitch angle references). Under fault, the quadcopter starts rotating uncontrollably around its z axis which leads to the variation of yaw angle and results in different vehicle's orientation at the end w.r.t. the initial value (the two vectors plotted in blue shown in Figure 5.3.1a). Under nominal functioning (from $t = 0$ to 2.5 seconds), employing the NMPC controller with terminal constraint set as shown in (5.2.19) ensures the tracking stability, hence, allowing the actual angles to reach their desired values within 10 steps as can be seen from the enlarging circle in Figure 5.3.4b. Bear in mind that under stuck fault, the reconfigured NMPC controller (5.2.36) does not employ the terminal stabilizing constraints, thus, results in slower convergence speeds regarding both the roll, pitch angles. Furthermore, as shown in Figure 5.3.4c, the two NMPC controllers keep the angular velocities ω_x (plotted in red line) and ω_y (plotted in green line) under their limit $\omega_{\max} = 2\ rad/s$ as given in (2.2.48). We also notice the chattering phenomena in the torque results given in Figure 5.3.4d which is caused by the sudden change of the piece-wise constant angle references (as shown in Figure 5.3.4b) and by the control effort of stabilizing the torques at their zero equilibrium. Therefore, we have reduced the phenomena by decreasing the weighting values of the torques (e.g., $R_{\widehat{\tau}} = 0.01\boldsymbol{I}_2$ as in Table 5.3.1).

Finally, the computation time of the NMPC attitude controllers given in (5.2.21) and (5.2.36) is plotted in Figure 5.3.5. It can be observed that under nominal cases, on average, the NMPC controller (5.2.21) (plotted in red line) requires 49.5 milliseconds per step (given by blue line) to

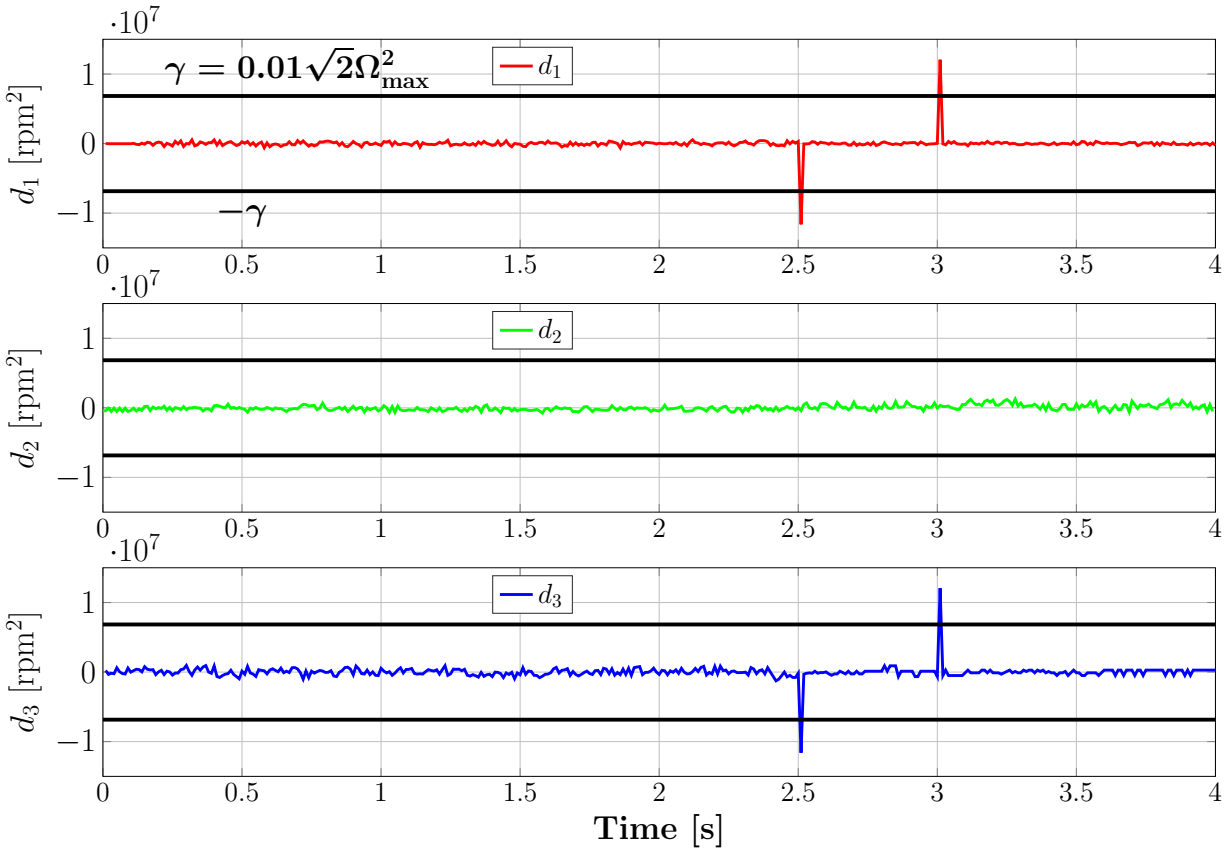
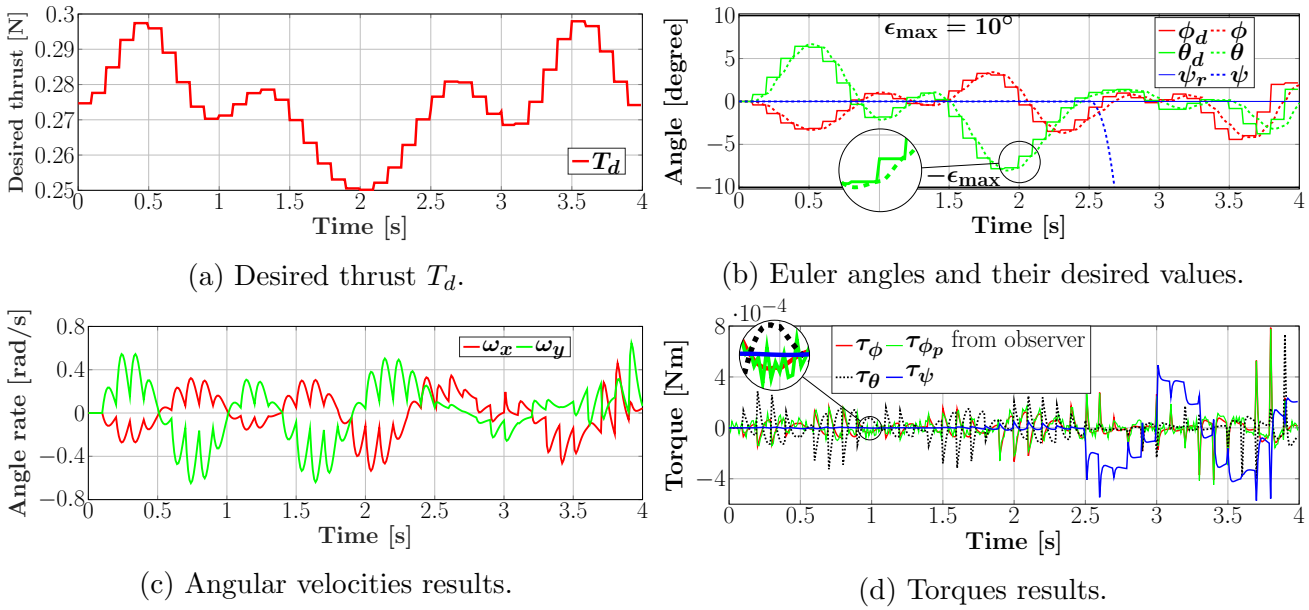


Figure 5.3.3: Values of residual vector $\boldsymbol{d} = [d_1\ d_2\ d_3]^\top$.



(a) Desired thrust T_d.

(b) Euler angles and their desired values.

(c) Angular velocities results.

(d) Torques results.

Figure 5.3.4: States and inputs results of the proposed FTC scheme under simulation.

compute which seems similar to the results given in Section 3.5.2 (where the average computing time was 54 milliseconds as shown in Figure 3.5.2). The nominal controller also yields less computing effort than the NMPC scheme under fault as in (5.2.36) (plotted in green line) with the average computing time of 60 milliseconds. This is due to the use of the prediction model for the yaw torque given by (5.2.32) and the longer prediction horizon $N_{p_f} = 8$ (w.r.t. $N_p = 5$

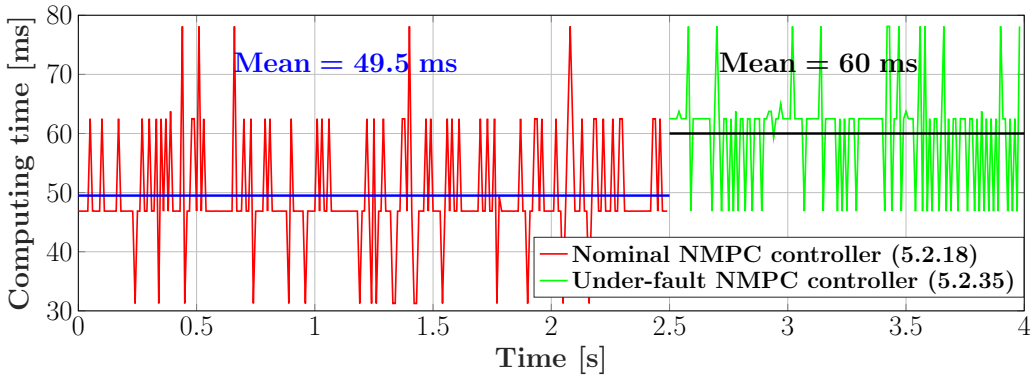


Figure 5.3.5: Computing time of the NMPC attitude controllers (5.2.21), (5.2.36).

under nominal functioning as given in Table 5.3.1). We also notice that the computation times are higher than the sampling time (i.e., 0.01 seconds) which would make the control strategy, in its present form, unsuitable for implementation. However, as already discussed at the end of Section 3.5.2, we strongly believe that employing various advanced works on speeding up NMPC design (e.g., to refine the algorithm's implementation [Zanelli et al., 2018] and/or use advanced solving techniques [Gros et al., 2012, Gulan et al., 2017]) can alleviate the computation burden of the proposed NMPC controllers (5.2.21), (5.2.36).

5.4 Concluding remarks and open questions

This chapter presented the design of a hierarchical FTC (Fault Tolerant Control) controller for the reliable trajectory tracking of a quadcopter system under rotor saturation constraints and the occurrence of a single fault of the "stuck rotor" nature. The high control level makes use of the feedback linearization controller detailed in Section 2.4 which provides bounded desired values of thrust and angles. The main contributions appear at the low control level which employs an NMPC controller, a speed calculator and a fault diagnosis module. The NMPC controller and the speed calculator function in two different modes: nominal and under-fault modes. Under both cases, the aim is to avoid the saturation effects on the four rotors, hence, allowing the actual inputs (thrust, torques) to equal their references, leading to the stability (and accurate trajectory tracking) of the whole scheme. Particular details are summarized as follows:

- The nominal controller design employs the NMPC attitude controller using terminal stabilizing constraints, as presented in Section 3.5. We also propose a transformation of the saturation constraints on the four rotor speeds into constraints on the three torques by making use of the bounded inputs property of the feedback linearization position controller at the high level. The speed calculation block is taken as the inversion of the rotor configuration matrix.

- Under stuck fault functioning, we provide an explicit formulation of a polytopic torque constraint set (defined only for two torques, roll and pitch) which guarantees the speed constraints. Then, a prediction model of the yaw torque is developed to facilitate the design of an NMPC controller for under fault model. The speed calculator block provides the speed references such that the faulty rotor receives its stuck speed as its reference.

- The fault diagnosis module constructs a residual multi-valued signal describing the differences between the speed references and their actual values. The actual speed values is

	Parameters	Value
Sampling time	δ_{att} as in (5.2.21)	0.01 seconds
Nominal functioning (5.2.19)	N_p as in (5.2.21)	5 steps
	$[T_{d_{\min}}, T_{d_{\max}}]$ as in (5.2.2)	$[0.202, 0.3486]$
	$\tau_{\max}$ as in (5.2.2)	$10^{-4} \times [25\ 25\ 50]^\top$
	$\varepsilon_{\max}$ as in (3.5.23)	10.3194
	$\mathsf{L}_{\text{CTC, max}}$ as in (3.5.36)	1.49×10^{-7}
	K as in (3.2.11) and (3.2.31)	$[-0.1\boldsymbol{I}_3\ -0.2\boldsymbol{I}_3]$
	Q as in (5.2.21)	$\text{diag}\{1, 1, 1, 0.1, 0.1, 0.1\}$
	R as in (5.2.21)	$0.01\boldsymbol{I}_3$
	R^* as in (3.5.37)	$\begin{bmatrix} 150\boldsymbol{I}_3 & 3\boldsymbol{I}_3 \\ 3\boldsymbol{I}_3 & 155\boldsymbol{I}_3 \end{bmatrix} 10^{-11}$
	P as in (5.2.21)	$\begin{bmatrix} 3.525\boldsymbol{I}_3 & 5\boldsymbol{I}_3 \\ 5\boldsymbol{I}_3 & 25.25\boldsymbol{I}_3 \end{bmatrix}$
Under-fault functioning (5.2.36)	N_{p_f} as in (5.2.36)	8 steps
	$Q_{\hat{\eta}}$ as in (5.2.36)	$\boldsymbol{I}_2$
	$Q_{\hat{\omega}}$ as in (5.2.36)	$0.1\boldsymbol{I}_2$
	$R_{\hat{\tau}}$ as in (5.2.36)	$0.01\boldsymbol{I}_2$

Table 5.3.1: Parameters of the NMPC attitude controllers (5.2.19) and (5.2.36).

estimated by a torque observer using the available feedback on the angular velocity. We employ the backward Euler discretization method within the observer.

- The fault detection and isolation is done assuming no concurrent faults (at most, a single rotor may be stuck) and is based on threshold breaking (detection) and sign and magnitude analysis (isolation) of the residual signal. Furthermore, the residual's values lead to an estimation of the fault's magnitude (i.e., the value, as percentage of the maximum velocity, at which the rotor gets stuck). Thus, the fault diagnosis module (which contains detection, isolation and estimation) is able to handle a fault occurrence and possible piece-wise changes in the fault magnitude.

The proposed FTC scheme was validated through extensive simulations and shows promise for future experimental tests. The NMPC controllers obtained an average computing time per step of 49.5 milliseconds and 60 milliseconds under nominal and faulty cases, respectively. Various questions remain open as topics of further research:

- What is the impact of the internal rotor dynamics on the control scheme, especially on the fault diagnosis module?

- How is the robustness of the control scheme under measurement errors and/or dynamics uncertainties.

- How is the performance of the proposed method under real applications? The question also includes the preparation process, i.e.: embedding the two NMPC attitude controllers, the speed calculator block and the fault diagnosis module into the on-board controller.

Chapter 6

Conclusions and future developments

6.1 Conclusions

This thesis presented the design of reliable control laws for motion planning of a multicopter system (with possible generalization for similar feedback linearizable systems) subject to state, input constraints and unexpected events occurring in the system (e.g., actuator faults). The chief theoretical notions on which this thesis hinged - differential flatness, FL (feedback linearization) and NMPC (Nonlinear Model Predictive Control) were coherently merged together, hence paving the way for original results on constrained trajectory generation and tracking control designs.

The control approach considered in the thesis was to generate off-line a feasible reference trajectory with respect to the nominal functioning of the multicopter system and then, to design an on-line tracking mechanism. Regarding the trajectory generation problem, we have exploited the differential flatness property of the system for designing robust angular constraints against possible modifications of the yaw angle during flight (e.g., for aerial photography/filming applications). The flat output was parametrized by using the B-spline curves properties (e.g.,the curve lies inside a union of convex hulls, B-spline derivatives can be expressed as linear combinations of lower order B-splines) which allowed us to consider a minimum-length trajectory generation algorithm subject to various system's constraints though an optimization problem with a standard quadratic cost function. The proposed approach was validated in simulation for a generic indoor trajectory generation application and a building inspection scenario where the vehicle's direction was varying over time for pointing the mounted camera towards the building. For the on-line tracking design, a hierarchical control architecture which decouples the scheme into position and attitude control was proposed. At the high level the position controller calculates the position error and provides the desired thrust and angles to the attitude controller at the low level to stabilize the system around the desired angles.

The originality of the contributions lies in the exploitation of FL controllers (and in particular a computed-torque control for the attitude control problem) for various tasks related to NMPC designs. We propose novel approaches for defining the necessary ingredients for guaranteeing the stability, the recursive feasibility and also for reducing the computing time (in comparison with existing approaches in the literature) of the NMPC schemes. The first direction is to construct an invariant set with unlimited expandability which leads to an NMPC design with semi-global stability guarantees. The design allows to tune the terminal constraint set such that it can ultimately cover an arbitrary compact set containing the possible initial states and hence, the prediction horizon length can be significantly reduced while still ensuring the closed-loop stability. The second approach is to employ a relaxed invariant set as the terminal constraint set

within the NMPC design. The set guarantees the presence of the state within itself at predefined periodic time instants (i.e., not at all times as considered for the standard invariant set), and hence, can be described in simple box-type constraints. These mitigate the complexity of the NMPC optimization problem and hence, remarkably reduce the computing time in comparison with using the terminal invariant set with similar size. Next, after simplifying the terminal constraints as done in the foregoing approach, we eliminated them and tried to guarantee the stability of the NMPC design only by using a "long enough" prediction horizon. We observe that using the aforementioned FL controller greatly enhances the design in the sense that the required prediction horizon length is much shorter than when using a standard linear controller (but still being impractically long for real applications).

The contribution also includes the design of an hierarchical optimization-based FTC (Fault Tolerant Control) scheme to counteract the stuck rotor fault for a quadcopter system in which the high level employs the FL controller while the low level switches between two different NMPC schemes according to the system's functioning states (healthy or under fault), both guaranteeing that the rotor speeds do not exceed their limit. We applied a set theoretic method to transform the rotor speeds constraints into the saturation constraints on the torques and hence, were able to apply the proposed NMPC design with guaranteed stability (which was done for the general multicopter system without specifying the rotors). A model of the uncontrolled yaw motion under fault was constructed which allows the prediction dynamics employed within the NMPC controller to be more realistic. We also proposed the design of a fault diagnosis module for detecting the stuck fault based on the differences between the estimated and reference rotor speeds.

Throughout the thesis, the contributions were validated under extensive simulations and part of them were applied for real experimental tests over a Crazyflie 2.0 quadcopter platform. Their effectiveness was highlighted via comparisons with several existing approaches (e.g., to employ a linear controller to design an NMPC scheme as in the quasi-infinite horizon NMPC approach).

6.2 Future developments

Throughout the manuscript, we have provided several open questions and remarks on feasible improvements at the end of each chapter. Some questions have been answered later on, at other parts of the thesis (e.g., a question about another control candidate for the position controller at the end of Chapter 2, page 52 is answered by the NMPC designs given in Chapter 4) yet, most of them still remains open and are classified into several large topics hereinafter.

1) Account for disturbances: The first improvement which can be made is to consider more uncertainties within the trajectory generation and also the control design. One example is to add bounded disturbances (e.g. caused by wind gusts [Hasan and Johansen, 2018]) into the dynamical system of the multicopter, then, take them into account when designing the controller and when analyzing the stability property. For solving this, a feasible solution is to adapt the proposed NMPC designs with (nominal) stability guarantees by using tube-based NMPC design [Alvarado et al., 2007, Mayne and Kerrigan, 2007, Yu et al., 2013]. The tube-based approach guarantees that the state always stays inside the predefined tube in the presence of bounded disturbances and furthermore, the nominal stability property established in this manuscript can be re-used for an analysis on the stability of the tube-based NMPC design [Alvarado et al., 2007].

2) Consider parameter uncertainties: Regarding the multicopter dynamical system as in (2.1.10), besides the mass m which is relatively easy to obtain, the inertia tensor J also plays an

important role in the control designs but is well-known for its complicated measuring procedure (c.f. Remark 5.2.6). Therefore, a mismatch on the value of J always exists in real applications and should be taken into account in the control design. One existing solution is to develop an adaptive law which compensates the estimated value of J as proposed in [Achtelik et al., 2011, Navabi and Soleymanpour, 2017]. However, note that these methods are not straightforward to apply under input saturation constraints as considered in the manuscript.

3) Enhance and generalize the NMPC scheme with stability guarantees: The contributions on the NMPC designs of this thesis are based on the use of the appropriate local controllers (required to design NMPC schemes with guaranteed stability), i.e., the FL (Feedback Linearization) controllers which better fit with the strongly nonlinear dynamics of a multicopter, than the standard linear constructions. Even though they succeed in dealing with the NMPC designs using terminal stabilizing constraints as considered in Chapter 3 and Section 4.2, their application for the NMPC design without terminal stabilizing constraints, i.e., still providing the extremely long prediction horizon lengths (c.f. Table 4.3.1), raise the question: *How to determine an appropriate local controller for designing an NMPC scheme for a specific system?*

Indeed, this is a big question due to the variety of system's types and various control designs possibly employed. For now, we are working on part of the problem. More precisely, by combining all the results presented in Chapters 3 and 4, we aim to construct an NMPC design framework for feedback linearizable systems including the partial FL case [Charlet et al., 1989, Spong, 1994]. The direction shows promise since the established elements, e.g., the input constraint satisfaction by using Taylor's approximation and the invariant set construction based on the resulted linear stable system as detailed in Section 3.2 for the computed-torque controller, hold for a general FL controller and hence, offer a possibility of generalizing the proposed NMPC designs for similar feedback linearizable systems. However, since a FL controller is totally based on the dynamical model of the system which probably contains errors and mismatches, we also want to take them into account as already stressed out in the foregoing paragraph as our first concern. Furthermore, we also look for different control candidates. For example, the multicopter dynamics actually admit a passivity-based controller [Ha et al., 2014, Meissen et al., 2017] which is capable of taking into account uncertainties (e.g., the mass as in [Ha et al., 2014] and supensed load as in [Meissen et al., 2017]). We positively believe that using this candidate control law can mitigate the robustness issues and provide a structural framework for designing an NMPC controller with guaranteed stability.

4) Extend for multi-agent control problems: The next extension which can be considered is to adapt the proposed controllers for multicopters formation control problems. For the trajectory generation part, the approach of using differential flatness and B-spline parametrization as proposed in Section 2.2 can be directly applied for the scenarios considering static obstacles while it needs to be simplified for on-line configuration when considering moving ones. Then, regarding the tracking mechanism, using an NMPC scheme allows us to take into account collision and obstacle avoidance by simply adding appropriate terms into the cost function [Prodan et al., 2013, Kuriki and Namerikawa, 2015, Rucco et al., 2015, Stoican et al., 2016]. Furthermore, the decentralized or distributed control approaches can be applied to reduce the complexity of the whole optimization problem and to benefit from the available processors equipped in the systems [Scattolini, 2009, Bemporad and Rocchi, 2011]. However, the resulted stability problems will become convoluted due to the fact that the local controller (i.e., having an explicit formulation) is usually not able to take into account collision and obstacle avoidance constraints. Therefore, we may mitigate the theoretical proofs of the closed-loop stability but this requires us to analyze and to demonstrate it through simulations and/or experiment tests, similarly to various related works in the literature [Bemporad and Rocchi, 2011, Rucco et al., 2015].

Appendix A

Flatness-based representation of the angular velocities

The flatness formulation of ω_x given in (2.2.8) is derived from its original formulation $\omega_x = \dot{\phi} - \dot{\psi}\sin\theta$ given in (2.1.7). At first, by using $\sin\phi = \Phi_x \sin\psi - \Phi_y \cos\psi$ from (2.2.5), we obtain the formulations of $\cos\phi$ and the derivative of the roll angle $\dot{\phi}$ as follows:

$$\cos\phi = \frac{\sqrt{(\ddot{x}\cos\psi + \ddot{y}\sin\psi)^2 + (\ddot{z}+g)^2}}{\sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z}+g)^2}} \tag{A.0.1}$$

$$\dot{\phi} = \frac{\dot{\Phi}_x \sin\psi - \dot{\Phi}_y \cos\psi + \dot{\psi}\left(\Phi_x \cos\psi + \Phi_y \sin\psi\right)}{\cos\phi}. \tag{A.0.2}$$

Next, by using $\tan\theta = \Theta_x \cos\psi + \Theta_y \sin\psi$ from (2.2.5), we have that:

$$\sin\theta = \frac{\ddot{x}\cos\psi + \ddot{y}\sin\psi}{\sqrt{(\ddot{x}\cos\psi + \ddot{y}\sin\psi)^2 + (\ddot{z}+g)^2}}. \tag{A.0.3}$$

Then, introducing (A.0.1)-(A.0.3) to $\omega_x = \dot{\phi} - \dot{\psi}\sin\theta$ from (2.1.7), we arrive to:

$$\omega_x = \frac{\dot{\Phi}_x \sin\psi - \dot{\Phi}_y \cos\psi + \dot{\psi}\left(\Phi_x \cos\psi + \Phi_y \sin\psi\right) - \dot{\psi}\left(\Phi_x \cos\psi + \Phi_y \sin\psi\right)}{\cos\phi}, \tag{A.0.4}$$

with Φ_x, Φ_y as in (2.2.7). This leads to the flatness formulation of ω_x given in (2.2.8).
Next, we show how to construct the angular velocity ω_y as in (2.2.9). At first, the derivative of the pitch angle $\dot{\theta}$ is calculated by using $\tan\theta$ from (2.2.5) as follows:

$$\dot{\theta} = \cos^2\theta\left(\dot{\Theta}_x \cos\psi + \dot{\Theta}_y \sin\psi + \dot{\psi}\left(-\Theta_x \sin\psi + \Theta_y \cos\psi\right)\right). \tag{A.0.5}$$

Introducing (A.0.5) to the formulation $\omega_y = \dot{\theta}\cos\phi + \dot{\psi}\sin\phi\cos\theta$ from (2.1.7) leads to:

$$\omega_y = \cos\phi\cos^2\theta\left(\dot{\Theta}\cos\psi + \dot{\Theta}_y \sin\psi\right) + \dot{\psi}\cos\theta\left(\cos\phi\cos\theta\left(-\Theta_x \sin\psi + \Theta_y \cos\psi\right) + \sin\phi\right), \tag{A.0.6}$$

in which, $\cos\phi\cos\theta\left(-\Theta_x \sin\psi + \Theta_y \cos\psi\right) + \sin\phi = 0$ is obtained by using Θ_x, Θ_y from (2.2.7), $\sin\phi$ from (2.2.5), $\cos\phi$ from (A.0.1) and $\cos\theta$ calculated from (A.0.3). This also validates the flatness representation of ω_y from (2.2.9).

Appendix B

Proof of Proposition 3.5.1

The explicit formulation of the remainder term $R_\varepsilon \triangleq [R_{\varepsilon,1}\ R_{\varepsilon,2}\ R_{\varepsilon,3}]^\top$ as in (3.5.17)–(3.5.19) is rewritten as follows:

$$R_{\varepsilon,1} = -J_x\left((\mathrm{s}\,\theta - \mathrm{s}\,\theta_d)\mu_\psi + \dot\theta\dot\psi\,\mathrm{c}\,\theta\right) + (J_z - J_y)(\dot\theta\,\mathrm{c}\,\phi + \dot\psi\,\mathrm{s}\,\phi\,\mathrm{c}\,\theta)(-\dot\theta\,\mathrm{s}\,\phi + \dot\psi\,\mathrm{c}\,\phi\,\mathrm{c}\,\theta), \tag{B.0.1}$$

$$\begin{aligned} R_{\varepsilon,2} =& J_y\left((\mathrm{c}\,\phi - \mathrm{c}\,\phi_d)\mu_\theta + (\mathrm{s}\,\phi\,\mathrm{c}\,\theta - \mathrm{s}\,\phi_d\,\mathrm{c}\,\theta_d)\mu_\psi - \dot\phi\dot\theta\,\mathrm{s}\,\phi + \dot\psi(\dot\theta\,\mathrm{c}\,\phi\,\mathrm{c}\,\theta - \dot\theta\,\mathrm{s}\,\phi\,\mathrm{s}\,\theta)\right) \\ &+ (J_x - J_z)\left(\dot\phi - \mathrm{s}\,\theta\dot\psi\right)\left(-\mathrm{s}\,\phi\dot\theta + \mathrm{c}\,\phi\,\mathrm{c}\,\theta\dot\psi\right), \end{aligned} \tag{B.0.2}$$

$$\begin{aligned} R_{\varepsilon,3} =& J_z\left(-(\mathrm{s}\,\phi - \mathrm{s}\,\phi_d)\mu_\theta + (\mathrm{c}\,\phi\,\mathrm{c}\,\theta - \mathrm{c}\,\phi_d\,\mathrm{c}\,\theta_d)\mu_\psi - \dot\phi\dot\theta\,\mathrm{c}\,\phi - \dot\psi\dot\phi\,\mathrm{s}\,\phi\,\mathrm{c}\,\theta - \dot\psi\dot\theta\,\mathrm{c}\,\phi\,\mathrm{s}\,\theta\right) \\ &+ (J_y - J_x)(\dot\phi - \mathrm{s}\,\theta\dot\psi)(\mathrm{c}\,\phi\dot\theta + \mathrm{s}\,\phi\,\mathrm{c}\,\theta\dot\psi). \end{aligned} \tag{B.0.3}$$

Next, for all $[\eta, \dot\eta, \mu_\eta]^\top \in \mathcal{B}(\eta_d, \varepsilon)$, the remainder term R_ε is bounded as follows:

$$\begin{aligned} |R_{\varepsilon,1}| &\le \frac{J_x}{2}\left(4\,\mathrm{s}^2\left(\frac{\theta-\theta_d}{2}\right) + \mu_\psi^2 + \dot\phi^2 + \dot\psi^2\right) + (J_z - J_y)\sqrt{(\dot\theta^2 + \dot\psi^2)(\dot\theta^2 + \dot\psi^2)} \\ &\le \left(\frac{J_x}{2} + J_z - J_y\right)\left((\phi-\phi_d)^2 + (\theta-\theta_d)^2 + \|\dot\eta\|^2 + \|\mu_\eta\|^2\right), \end{aligned} \tag{B.0.4}$$

$$\begin{aligned} |R_{\varepsilon,2}| &\le \frac{J_y}{2}\left(4\,\mathrm{s}^2\left(\frac{\phi-\phi_d}{2}\right) + \mu_\theta^2 + 4\,\mathrm{s}^2\left(\frac{\phi-\phi_d}{2}\right) + \mu_\psi^2 + 4\,\mathrm{s}^2\left(\frac{\theta-\theta_d}{2}\right) + \mu_\psi^2 + \dot\phi^2 + \dot\theta^2 + \dot\psi^2\right) \\ &\quad + |J_z - J_x|\sqrt{(1+\mathrm{s}^2\,\phi)(\dot\phi^2 + \dot\psi^2)(\mathrm{c}^2\,\phi + \mathrm{s}^2\,\phi\,\mathrm{c}^2\,\theta)(\dot\theta^2 + \dot\psi^2)} \\ &\le \left(J_y + |J_z - J_x|\sqrt{1+\mathrm{s}^2\,\epsilon_{\max}}\right)\left((\phi-\phi_d)^2 + (\theta-\theta_d)^2 + \|\dot\eta\|^2 + \|\mu_\eta\|^2\right), \end{aligned} \tag{B.0.5}$$

$$\begin{aligned} |R_{\varepsilon,3}| &\le \frac{J_z}{2}\left(4\,\mathrm{s}^2\left(\frac{\phi-\phi_d}{2}\right) + \mu_\theta^2 + 4\,\mathrm{s}^2\left(\frac{\phi-\phi_d}{2}\right) + \mu_\psi^2 + 4\,\mathrm{s}^2\left(\frac{\theta-\theta_d}{2}\right) + \mu_\psi^2 + \dot\phi^2 + \dot\theta^2 + \dot\psi^2\right) \\ &\quad + |J_y - J_x|\sqrt{(1+\mathrm{s}^2\,\theta)(\dot\phi^2 + \dot\psi^2)(\mathrm{c}^2\,\phi + \mathrm{s}^2\,\phi\,\mathrm{c}^2\,\theta)(\dot\theta^2 + \dot\psi^2)} \\ &\le \left(J_z + |J_y - J_x|\sqrt{1+\mathrm{s}^2\,\epsilon_{\max}}\right)\left((\phi-\phi_d)^2 + (\theta-\theta_d)^2 + \|\dot\eta\|^2 + \|\mu_\eta\|^2\right). \end{aligned} \tag{B.0.6}$$

Appendix C

Stability of error dynamics (2.4.16) using the nested control design

This section proves the stability of the error dynamics (2.4.16) which can be split into three scalar cases due to the definition of the error vector $e_\xi = [e_x \ e_y \ e_z]^\top \in \mathbb{R}^3$ and the usage of the diagonal weighting matrices K_1, K_2 as in (2.4.12). We generalize all the three scalar dynamics by the following system using the indice q ($q \in \{x, y, z\}$):

$$\ddot{e}_q = \text{sat}\left(K_{1_q}\dot{e}_q + \text{sat}\left(-K_{1_q}K_{2_q}e_q + K_{2_q}\dot{e}_q,\ \frac{\lambda_q}{2}\right), \lambda_q\right) \tag{C.0.1}$$

with $e_q \in \mathbb{R}$ representing for any element of e_ξ. The control parameters $K_{q_1}, K_{q_2} < 0$ and $\lambda_q > 0$ taken from the diagonal matrices $K_1 \triangleq \text{diag}\{K_{1_x}, K_{1_y}, K_{1_z}\}$, $K_2 \triangleq \text{diag}\{K_{2_x}, K_{2_y}, K_{2_z}\}$ from (2.4.12) and $\lambda \triangleq [\lambda_x \ \lambda_y \ \lambda_z]^\top$ as in (2.4.13). The forthcoming result is a generalization of the proof for the particular case of $K_{q_1} = K_{q_2} = 1$ which was introduced in [Teel, 1992]. Considering the new variables e_1, e_2 defined as:

$$e_1 = -K_{1_q}e_q + \dot{e}_q, \quad e_2 = \dot{e}_q. \tag{C.0.2}$$

Introducing (C.0.2) into (C.0.1) leads to:

$$\dot{e}_1 = -K_{1_q}e_2 + \text{sat}\left(K_{2_q}e_2 + \text{sat}\left(K_{2_q}e_1,\ \frac{\lambda_q}{2}\right), \lambda_q\right), \tag{C.0.3}$$

$$\dot{e}_2 = \text{sat}\left(K_{1_q}e_2 + \text{sat}\left(K_{2_q}e_1,\ \frac{\lambda_q}{2}\right), \lambda_q\right). \tag{C.0.4}$$

Then, we will prove that the equilibrium ($e_1 = 0, e_2 = 0$) of the system (C.0.3)-(C.0.4) is globally asymptotically stable.
Firstly, we consider the "big" value of e_2, such that:

$$|e_2| > \frac{\lambda_q}{|K_{1_q}|}, \tag{C.0.5}$$

By defining the Lyapunov function $V_2 = e_2^2$, we have that:

$$\dot{V}_2 = 2e_2\,\text{sat}\left(K_{1_q}e_2 + \text{sat}\left(K_{2_q}e_1,\ \frac{\lambda_q}{2}\right), \lambda_q\right). \tag{C.0.6}$$

Case 1: $e_2 > 0$, (C.0.5) leads to:

$$K_{1_q}e_2 + \text{sat}\left(K_{2_q}e_1,\ \frac{\lambda_q}{2}\right) < \frac{\lambda_q}{2} - \frac{\lambda_q}{2} = 0. \tag{C.0.7}$$

Introducing condition (C.0.7) and $e_2 > 0$ into (C.0.6) strictly leads to $\dot{V}_2 < 0$.
Case 2: $e_2 < 0$, (C.0.5) leads to:

$$K_{1_q} e_2 + \text{sat}\left(K_{2_q} e_1,\ \frac{\lambda_q}{2}\right) > -\frac{\lambda_q}{2} + \frac{\lambda_q}{2} = 0. \tag{C.0.8}$$

Introducing condition (C.0.8) and $e_2 < 0$ into (C.0.6) strictly leads to $\dot{V}_2 < 0$.
Thus, if $e_2 \notin E_2 = \{e_2 : |e_2| \leq \lambda_q/|K_{1_q}|\}$, $V_2 = e_2^2$ strictly decreases. Consequentially, e_2 enters E_2 in a finite time and remains in E_2.
Secondly, we consider that $e_2 \in E_2$, i.e. $|e_2| \leq \lambda_q/|K_{1_q}|$, we arrive to:

$$\left| K_{1_q} e_2 + \text{sat}\left(K_{2_q} e_1,\ \frac{\lambda_q}{2}\right) \right| \leq \frac{\lambda_q}{2} + \frac{\lambda_q}{2} = \lambda_q. \tag{C.0.9}$$

Thus, we have that:

$$\text{sat}\left(K_{1_q} e_2 + \text{sat}\left(K_{2_q} e_1,\ \frac{\lambda_q}{2}\right)\right) = K_{1_q} e_2 + \text{sat}\left(K_{2_q} e_1,\ \frac{\lambda_q}{2}\right). \tag{C.0.10}$$

Introducing (C.0.10) into (C.0.3) leads to:

$$\dot{e}_1 = \text{sat}\left(K_{2_q} e_1,\ \frac{\lambda_q}{2}\right). \tag{C.0.11}$$

By considering the Lyapunov function $V_1 = e_1^2$, we have that:

$$\dot{V}_1 = 2 e_1 \,\text{sat}\left(K_{2_q} e_1,\ \frac{\lambda_q}{2}\right) < 0. \tag{C.0.12}$$

Since $K_{2_q} < 0$, V_1 is decreasing which indicates that e_1 enters $E_1 = \{e_1, |e_1| \leq \lambda_q/(2|K_{2_q}|)\}$ in a finite time. When $e_1 \in E_1$, the dynamics (C.0.3)-(C.0.4) become:

$$\dot{e}_1 = K_{2_q} e_1, \tag{C.0.13}$$
$$\dot{e}_2 = K_{2_q} e_1 + K_{1_q} e_2, \tag{C.0.14}$$

which is exponentially stable around the origin ($e_1 = 0, e_2 = 0$) since $\langle K_{1_q}, K_{2_q} \rangle < 0$. Thus, ($e_q = 0, \dot{e}_q = 0$) is also globally asymptotically stable for the error dynamics (C.0.1), hence, completing the proof.

Appendix D

Proof of Proposition 2.4.5

From the polytopic constraint (2.2.40) (c.f. Figure 2.2.2), we obtain that:

$$\ddot{x}_r^2 + \ddot{y}_r^2 \leq \frac{T_{r_{\max}}^2}{m^2} \sin^2 \epsilon_{r_{\max}}, \tag{D.0.1}$$

$$\frac{T_{r_{\min}}^2}{m^2} \cos^2 \epsilon_{r_{\max}} \leq (\ddot{z}_r + g)^2 \leq \frac{T_{r_{\max}}^2}{m^2}. \tag{D.0.2}$$

with $\ddot{q}_r$, $q \in \{x, y, z\}$ the references of the translation accelerations, m the system mass, $[T_{r_{\min}}, T_{r_{\max}}]$ the range of the thrust reference as in (2.2.37) and $\epsilon_{r_{\max}}$ the maximum value of the roll, pitch angles references as in (2.2.36). Then, from (D.0.1), we obtain that:

$$\langle \max |\ddot{x}_r|, \max |\ddot{y}_r| \rangle \leq \frac{T_{r_{\max}}}{m} \sin \epsilon_{r_{\max}}. \tag{D.0.3}$$

Therefore, in order to guarantee the existence of (U_x, U_y) such that $U_x > \max |\ddot{x}_r|$ and $U_y > \max |\ddot{y}_r|$ as required in (2.4.14), we need to ensure:

$$\langle U_x, U_y \rangle > \frac{T_{r_{\max}}}{m} \sin \epsilon_{r_{\max}}. \tag{D.0.4}$$

Next, by introducing $|\ddot{z}_r| < \max |\ddot{z}_r| < U_z < g$ (obtained from (2.4.5) and (2.4.14)) to (D.0.2), we have that:

$$\ddot{z}_r \in \left[\frac{T_{r_{\min}}}{m} \cos \epsilon_{r_{\max}} - g, \frac{T_{r_{\max}}}{m} - g \right], \tag{D.0.5}$$

which leads to to:

$$\max |\ddot{z}_r| \leq \max \left(g - \frac{T_{r_{\min}}}{m} \cos \epsilon_{r_{\max}}, \frac{T_{r_{\max}}}{m} - g \right). \tag{D.0.6}$$

Therefore, it is necessary to constrain U_z as in (2.4.14) such that:

$$U_z > \max \left(g - \frac{T_{r_{\min}}}{m} \cos \epsilon_{r_{\max}}, \frac{T_{r_{\max}}}{m} - g \right). \tag{D.0.7}$$

By considering two conditions on (U_x, U_y, U_z) (2.4.6)–(2.4.7), we arrive to:

$$U_z \leq g - \operatorname{cotan}(\epsilon_{\max}) \sqrt{U_x^2 + U_y^2}, \tag{D.0.8}$$

$$U_z \leq \sqrt{\frac{T_{\max}^2}{m^2} - U_x^2 - U_y^2} - g, \tag{D.0.9}$$

with $T_{\max}$ the maximum thrust of the multicopter system as in (2.1.13) (i.e, $T_{\max} > T_{r_{\max}}$ from (D.0.7)).
If $T_{r_{\min}} \cos \epsilon_{r_{\max}} + T_{r_{\max}} < 2mg$, introducing (D.0.4) and (D.0.7) to (D.0.8)–(D.0.9) leads to the following:

$$\sqrt{2} \tan \epsilon_{r_{\max}} \frac{T_{r_{\max}}}{T_{r_{\min}}} < \tan(\epsilon_{\max}), \tag{D.0.10}$$

$$\sqrt{(2mg - T_{r_{\min}} \cos \epsilon_{r_{\max}})^2 + 2T_{r_{\max}}^2 \sin^2 \epsilon_{r_{\max}}} < T_{\max}, \tag{D.0.11}$$

otherwise, if $T_{r_{\min}} \cos \epsilon_{r_{\max}} + T_{r_{\max}} \geq 2mg$, we have that:

$$T_{r_{\max}}(1 + \sqrt{2} \operatorname{cotan}(\epsilon_{\max}) \sin \epsilon_{r_{\max}}) \leq 2mg, \tag{D.0.12}$$

$$T_{r_{\max}} \sqrt{1 + \sin^2 \epsilon_{r_{\max}}} \leq T_{\max}. \tag{D.0.13}$$

The case of $T_{r_{\min}} \cos \epsilon_{r_{\max}} + T_{r_{\max}} < 2mg$ and the corresponding results (D.0.10)–(D.0.11) are introduced in Proposition 2.4.5 since they are very easy to achieve. This also completes the proof.

Appendix E

Proof of Lemma 4.2.16

It's straightforward to obtain the solutions of (4.2.75) given as follows:

$$\mathsf{x}_1(t) = \mathsf{x}_{1,0}\lambda_1(s,t) + \mathsf{x}_{2,0}\lambda_2(s,t), \tag{E.0.1a}$$
$$\mathsf{x}_2(t) = \mathsf{x}_{1,0}\lambda_3(s,t) + \mathsf{x}_{2,0}\lambda_4(s,t), \tag{E.0.1b}$$

where the initial conditions are $\mathsf{x}(0) = [\mathsf{x}_{1,0}\ \mathsf{x}_{2,0}]^\top$ and the functions $\lambda_i(\cdot)$, with $i \in \{1,\dots,4\}$ are given as in (4.2.77). Note that, by considering $s_1 < s_2 < 0$, we obtain the following results:

$$0 < \lambda_1(s,t) \leq 1, \qquad \lambda_2(s,t) > 0, \tag{E.0.2a}$$
$$\lambda_3(s,t) < 0, \qquad |\lambda_4(s,t)| \leq 1. \tag{E.0.2b}$$

Then, by applying the Triangle Inequality (and also considering (E.0.2)) to (E.0.1) we obtain:

$$|\mathsf{x}_1(t)| \leq \ \ |\mathsf{x}_{1,0}|\lambda_1(s,t) + |\mathsf{x}_{2,0}|\lambda_2(s,t), \tag{E.0.3a}$$
$$|\mathsf{x}_2(t)| \leq -|\mathsf{x}_{1,0}|\lambda_3(s,t) + |\mathsf{x}_{2,0}\lambda_4(s,t)|, \tag{E.0.3b}$$

with $\mathsf{x}_{1,0} = \mathsf{x}_1(0)$ and $\mathsf{x}_{2,0} = \mathsf{x}_2(0)$. Let us consider the initial state within the set $\mathcal{R}_\mathsf{x}$ (4.2.81), i.e., $|\mathsf{x}_{1,0}| \leq X_{1_{\max}}$ and $|\mathsf{x}_{2,0}| \leq X_{2_{\max}}$. Then, (E.0.3) leads to:

$$|\mathsf{x}_1(t)| \leq \ \ X_{1_{\max}}\lambda_1(s,t) + X_{2_{\max}}\lambda_2(s,t), \tag{E.0.4a}$$
$$|\mathsf{x}_2(t)| \leq -X_{1_{\max}}\lambda_3(s,t) + X_{2_{\max}}|\lambda_4(s,t)|. \tag{E.0.4b}$$

Furthermore, from (4.2.80) and (E.0.2), X, V will satisfy:

$$\lambda_2(s,\delta)X_{2_{\max}} \leq (1 - \lambda_1(s,\delta))X_{1_{\max}}, \tag{E.0.5a}$$
$$-\lambda_3(s,\delta)X_{1_{\max}} \leq (1 - |\lambda_4(s,\delta)|)X_{2_{\max}}. \tag{E.0.5b}$$

By introducing (E.0.5) to (E.0.4) at $t = \delta$, it is straightforward to obtain:

$$|\mathsf{x}_1(\delta)| \leq X_{1_{\max}},\ |\mathsf{x}_2(\delta)| \leq X_{2_{\max}}. \tag{E.0.6}$$

This proves the δ-invariant property (4.2.73) of $\mathcal{R}_\mathsf{x}$ (4.2.81).

Then, by considering (E.0.4), for any initial state within the set $\mathcal{R}_\mathsf{x}$ (4.2.81), we have that:

$$\max_{t\in[0,\delta]} |\mathsf{x}_1(t)| \leq \max_{t\in[0,\delta]} \{X_{1_{\max}}\lambda_1(s,t) + X_{2_{\max}}\lambda_2(s,t)\}, \tag{E.0.7}$$
$$\max_{t\in[0,\delta]} |\mathsf{x}_2(t)| \leq \max_{t\in[0,\delta]} \{-X_{1_{\max}}\lambda_3(s,t) + X_{2_{\max}}|\lambda_4(s,t)|\},$$

in which the two right hand sides are exactly $\widetilde{X}_{1_{\max}}$ (4.2.83a) and $\widetilde{X}_{2_{\max}}$ (4.2.83b). Furthermore, the followings hold for any $t \in [0, \delta]$:

$$|\mathsf{x}_1(t)| \leq \max_{t\in[0,\delta]} |\mathsf{x}_1(t)|, \ |\mathsf{x}_2(t)| \leq \max_{t\in[0,\delta]} |\mathsf{x}_2(t)|. \tag{E.0.8}$$

Thus, (E.0.7) and (E.0.8) lead to $\mathsf{x}(t) \in \mathcal{B}_{\mathcal{R}_\mathsf{x}}$ for all $t \in [0, \delta]$. Since $\mathsf{x}(\delta) \in \mathcal{R}_\mathsf{x}$ from (E.0.6), we again obtain $\mathsf{x}(t) \in \mathcal{B}_{\mathcal{R}_\mathsf{x}}$ for all $t \in [\delta, 2\delta]$. Hence, by further recursive analysis, $\mathsf{x}(t) \in \mathcal{B}_{\mathcal{R}_\mathsf{x}}$ for any $t \geq 0$. Thus, the safe set $\mathcal{B}_{\mathcal{R}_\mathsf{x}}$ as in (4.2.82) is validated.

Appendix F

Proof of the closed-loop stability of Lemma 4.2.20

Recursive feasibility: Assuming that the first NMPC iteration succeeds at time t, we obtain the following results from the first optimal input, $\bar{\boldsymbol{u}}^*(\tau, t)$, and state, $\bar{\boldsymbol{x}}^*(\tau, t)$, trajectories:

$$\begin{cases} \bar{\boldsymbol{u}}^*(\tau, t) \in \mathcal{U}, \ \forall \tau \in [t, t + T_p], \\ \bar{\boldsymbol{x}}^*(\tau, t) \in \mathcal{X}, \ \forall \tau \in [t, t + T_p], \\ \bar{\boldsymbol{x}}^*(t + T_p, t + \delta) \in \mathcal{R}, \end{cases} \tag{F.0.1}$$

in which the terminal δ-invariant set $\mathcal{R}$ follows the condition **C2**. Furthermore, by applying the control action $\boldsymbol{u}_{\text{MPC}}$ (4.2.3) to the nominal system (3.1.1), the resulted state trajectory $\boldsymbol{x}(\tau) = \bar{\boldsymbol{x}}^*(\tau, t)$ for $\tau \in [t, t + \delta]$.
At the next time step $t + \delta$, let us consider a candidate input trajectory chosen as follows:

$$\bar{\boldsymbol{u}}_c(\tau, t + \delta) \triangleq \begin{cases} \bar{\boldsymbol{u}}^*(\tau, t), & \tau \in [t + \delta, t + T_p], \\ \boldsymbol{u}_{\text{loc}}(\bar{\boldsymbol{x}}_c(\tau, t + \delta)), & \tau \in [t + T_p, t + T_p + \delta], \end{cases} \tag{F.0.2}$$

where the resulted candidate state $\bar{\boldsymbol{x}}_c(\tau, t + \delta)$ is obtained by introducing $\bar{\boldsymbol{u}}_c(\tau, t + \delta)$ to the dynamics (3.1.1). Within the time interval $[t + \delta, t + T_p]$, by applying the candidate input trajectory $\bar{\boldsymbol{u}}_c$ (F.0.2) to the nominal system (3.1.1), it is trivial to have that:

$$\begin{cases} \bar{\boldsymbol{u}}_c(\tau, t + \delta) \in \mathcal{U}, \ \forall \tau \in [t + \delta, t + T_p], \\ \bar{\boldsymbol{x}}_c(\tau, t + \delta) = \bar{\boldsymbol{x}}^*(\tau, t) \in \mathcal{X}, \ \forall \tau \in [t + \delta, t + T_p], \\ \bar{\boldsymbol{x}}_c(t + T_p, t + \delta) \in \mathcal{R}. \end{cases} \tag{F.0.3}$$

Thus, the δ-invariant property of $\mathcal{R}$ implies that:

$$\begin{cases} \bar{\boldsymbol{x}}_c(t + T_p + \delta, t + \delta) \in \mathcal{R}, \\ \bar{\boldsymbol{x}}_c(\tau, t + \delta) \in \mathcal{B}_{\mathcal{R}} \in \mathcal{X}, \ \forall \tau \in [t + T_p, t + T_p + \delta], \\ \bar{\boldsymbol{u}}_c(\tau, t + \delta) \triangleq \boldsymbol{u}_{\text{loc}}(\bar{\boldsymbol{x}}_c(\tau, t + \delta)) \in \mathcal{U}, \end{cases} \tag{F.0.4}$$

in which the third assertion follows condition **C1**, i.e., the safe set $\mathcal{B}_{\mathcal{R}}$ is constraints admissible. Then, (F.0.3)-(F.0.4) validate the candidate input trajectory $\bar{\boldsymbol{u}}_c(\cdot, t + \delta)$ given in (F.0.2). Hence, recursive feasibility is obtained by further recursion.
Asymptotic stability: The closed-loop stability is proved similarly to various existing works in the

literature [Chen and Allgöwer, 1998, Mayne et al., 2000] by employing the optimal cost function as a Lyapunov function candidate:

$$V(t) = J(\boldsymbol{x}(t), \bar{\boldsymbol{u}}^*(\cdot, t)). \tag{F.0.5}$$

Then, by employing the candidate input trajectory $\bar{\boldsymbol{u}}_c(\cdot, t+\delta)$ as in (F.0.2) and its resulted state $\bar{\boldsymbol{x}}_c(\cdot, t+\delta)$ as in (F.0.3)-(F.0.4), we obtain that:

$$\begin{aligned} V(t+\delta) - V(t) \leq & - \int_t^{t+\delta} \ell(\boldsymbol{x}^*(\tau, t), \boldsymbol{u}^*(\tau, t)) d\tau \\ & \int_{t+T_p}^{t+T_p+\delta} \left(\frac{dF(\bar{\boldsymbol{x}}_c(\tau))}{d\tau} + \ell(\bar{\boldsymbol{x}}_c(\tau), \bar{\boldsymbol{u}}_c(\tau)) \right) d\tau, \end{aligned} \tag{F.0.6}$$

where the candidate state $\bar{\boldsymbol{x}}_c(\tau, t+\delta)$ and input $\bar{\boldsymbol{u}}_c(\tau, t+\delta)$ are shortened as $\bar{\boldsymbol{x}}_c(\tau)$ and $\bar{\boldsymbol{u}}_c(\tau)$, respectively. Next, $\bar{\boldsymbol{u}}_c(\tau) = \boldsymbol{u}_{\text{loc}}(\bar{\boldsymbol{x}}_c(\tau)), \forall \tau \in [t+T_p, t+T_p+\delta]$ from (F.0.2) and the state $\bar{\boldsymbol{x}}_c(t+T_p) \in \mathcal{R}$ from (F.0.3), hence, from condition **C3**, we arrive to:

$$V(t+\delta) - V(t) \leq - \int_t^{t+\delta} \ell(\boldsymbol{x}^*(\tau, t), \boldsymbol{u}^*(\tau, t)) d\tau \leq 0. \tag{F.0.7}$$

Thus, the non-negative Lyapunov function $V(t)$ (F.0.5) is decreasing between two consecutive time instants. This validates the asymptotic stability. Similar proofs can be found in [Chen and Allgöwer, 1998, Mayne et al., 2000] for the case of using the standard terminal invariant sets.

Appendix G

Proof of Proposition 4.3.4

Since the matrices Q as in (4.2.4) and P_d as in (4.3.14) are both symmetric and positive definite, we have the following chain of inequalities:

$$\|\mathsf{p}_k\|_{P_d}^2 \leq \max(\text{eig}(P_d))\|\mathsf{p}_k\|^2 \leq \frac{\max(\text{eig}(P_d))}{\min(\text{eig}(Q))}\|\mathsf{p}_k\|_Q^2, \tag{G.0.1}$$

which further leads to:

$$\underbrace{\|\mathsf{p}_k\|_{P_d}^2 \leq r_d^2 \min(\text{eig}(P_d))}_{\mathsf{p}_k \in \mathcal{S}(P_d, r_d)}, \forall \|\mathsf{p}_k\|_Q^2 \leq c, \tag{G.0.2}$$

with c defined in (4.3.20) and $\mathcal{S}(P_d, r_d)$ the admissible invariant set as in (4.3.15).
Next, let us consider the NMPC optimization problem (4.3.1)–(4.3.3) with an initial state p_0 satisfying $\|\mathsf{p}_0\|_Q^2 \leq c$ as implied in (4.3.8). Next, we propose a feasible candidate for the NMPC optimization problem (4.3.1)–(4.3.3) consisting of the predicted state and input trajectories $(\bar{\mathsf{p}}_0(i), \bar{u}_0(i))$ as employed in (4.3.1) for all $i \in \{0, N_p - 1\}$ (with N_p the prediction horizon length as in (4.3.1)):

$$\bar{\mathsf{p}}_0(i+1) = \mathcal{F}(\bar{\mathsf{p}}_0(i), \bar{u}_0(i), \psi), \tag{G.0.3}$$

$$\bar{u}_0(i) = u_{\text{FL}}(K_d \bar{\mathsf{p}}_0(i), \psi), \tag{G.0.4}$$

with $u_{\text{FL}}(K_d \bar{\mathsf{p}}_0(i), \psi)$ the FL controller as in (4.1.4) and K_d from (4.3.13) being admissible for all $\bar{\mathsf{p}}_0(i) \in \mathcal{S}(P_d, r_d)$ as in (4.3.15). Since $\bar{\mathsf{p}}_0(0) = \mathsf{p}_0 \in \mathcal{S}(P_d, r_d)$ (due to (G.0.2)), the whole state trajectory $\bar{\mathsf{p}}_0(\cdot)$ remains within the invariant set $\mathcal{S}(P_d, r_d)$ as in (4.3.15) under the FL controller $u_{\text{FL}}(\cdot)$. These validate the the candidates (G.0.3)–(G.0.4).
Then, since V_{N_p} as in (4.3.1) is the optimal value function of the optimization problem (4.2.1), it satisfies:

$$V_{N_p}(\mathsf{p}_0) \leq J_{N_p}(\mathsf{p}_0, \bar{u}_0(\cdot)) = \sum_{i=0}^{N_p-1} \ell\left(\bar{\mathsf{p}}_0(i), u_{\text{FL}}(K_d \bar{\mathsf{p}}_0(i), \psi)\right), \tag{G.0.5}$$

with $\bar{u}_0(i) = u_{\text{FL}}(K_d \bar{\mathsf{p}}_0(i), \psi)$ as in (G.0.4). Furthermore, the stage cost $\ell\left(\bar{\mathsf{p}}_0(i), u_{\text{FL}}(K_d \bar{\mathsf{p}}_0(i), \psi)\right)$ as in (G.0.5) is bounded as follows:

$$\ell\left(\bar{\mathsf{p}}_0(i), u_{\text{FL}}(K_d \bar{\mathsf{p}}_0(i), \psi)\right) = \|\bar{\mathsf{p}}_0(i)\|_Q^2 + \|u_{\text{FL}}(K_d \bar{\mathsf{p}}_0(i), \psi) - u_e\|_R^2 \tag{G.0.6}$$

$$\leq \|\bar{\mathsf{p}}_0(i)\|_Q^2 + \max(\text{eig}(R)) \mathsf{L} \bar{\mathsf{p}}_0^\top(i) K_d^\top K_d \, \bar{\mathsf{p}}_0(i) = \|\bar{\mathsf{p}}_0(i)\|_{Q^*}^2, \tag{G.0.7}$$

with Q^* in (4.3.21). Then, since the whole predicted state trajectory $\bar{\mathsf{p}}_0(\cdot)$ as in (G.0.3), it exponentially converges as proven in (4.3.17):

$$\begin{aligned}\ell\left(\bar{\mathsf{p}}_0(i), u_{\text{FL}}(K_d\bar{\mathsf{p}}_0(i), \psi)\right) \leq \|\bar{\mathsf{p}}_0(i)\|^2_{Q^*} &\leq \frac{\max(\text{eig}(Q^*))}{\min(\text{eig}(P_d))}\rho^i\|\mathsf{p}_0\|^2_{P_d} \\ &\leq \frac{\max(\text{eig}(Q^*))\max(\text{eig}(P_d))}{\min(\text{eig}(Q))\min(\text{eig}(P_d))}\rho^i\|\mathsf{p}_0\|^2_{Q}. \end{aligned} \tag{G.0.8}$$

Lastly, combining (G.0.5) and (G.0.8) implies that:

$$V_{N_p}(\mathsf{p}_0) \leq \sum_{i=0}^{\infty}\rho^i\frac{\max(\text{eig}(Q^*))\max(\text{eig}(P_d))}{\min(\text{eig}(Q))\min(\text{eig}(P_d))}\|\mathsf{p}_0\|^2_Q = \gamma\|\mathsf{p}_0\|^2_Q, \tag{G.0.9}$$

with γ from (4.3.20). Thus, Assumption 4.3.1 is validated and the stability of the corresponding u-NMPC design follows Theorem 4.3.2. Hence, completing the proof.

Appendix H

Construction of function γ_2 defined in (4.3.22)

In a very similar fashion as solving the continuous-time Lyapunov equation (4.2.6) detailed in Proposition 4.2.8, the symmetric matrix P_d obtained from (4.3.14) is given by:

$$P_d = \begin{bmatrix} p_1 \boldsymbol{I}_3 & p_3 \boldsymbol{I}_3 \\ p_3 \boldsymbol{I}_3 & p_2 \boldsymbol{I}_3 \end{bmatrix}, \tag{H.0.1}$$

with $(p_1, p_2, p_3) \in \mathbb{R}$ satisfying:

$$\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} 1 - a_1^2 & -2a_1 a_3 & -a_3^2 \\ -a_2^2 & -2a_2 a_4 & 1 - a_4^2 \\ -a_1 a_2 & 1 - a_1 a_4 - a_2 a_3 & -a_3 a_4 \end{bmatrix}^{-1} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix}, \tag{H.0.2}$$

in which, (m_1, m_2, m_3) are the parametrization of the matrix M as in (4.2.13) and (a_1, a_2, a_3, a_4) are taken from the matrix A_{K_d} as in (4.3.26):

$$\begin{aligned} a_1 &= 1 + \frac{\delta^2}{2} k_1, & a_2 &= \delta \left(1 + \frac{\delta}{2} k_2 \right) \\ a_3 &= \delta k_1, & a_4 &= 1 + \delta k_2, \end{aligned} \tag{H.0.3}$$

with δ as in (4.3.26) the sampling time and (k_1, k_2) the control gains as in (4.3.24)–(4.3.25). Then, as similar to the results of Corollary 4.2.9, the eigenvalues of the matrix P_d as in (H.0.1) is given by:

$$\min(\text{eig}(P_d)) = \frac{1}{2} \left(p_1 + p_2 - \sqrt{(p_1 - p_2)^2 + 4p_3^2} \right), \tag{H.0.4}$$

$$\max(\text{eig}(P_d)) = \frac{1}{2} \left(p_1 + p_2 + \sqrt{(p_1 - p_2)^2 + 4p_3^2} \right), \tag{H.0.5}$$

with (p_1, p_2, p_3) as in (H.0.2). Note that, the minimum eigenvalue of the matrix M from (4.2.13) is also constructed similarly to (H.0.4):

$$\min(\text{eig}(M)) = \frac{1}{2} \left(1 + m_2 - \sqrt{(1 - m_2)^2 + 4m_3^2} \right), \tag{H.0.6}$$

which is due to $m_1 = 1$ as in (4.2.13). Then, by introducing the three eigenvalues formulations (H.0.4)–(H.0.6) to γ_2 defined in (4.3.22), we have that:

$$\gamma_2 \triangleq \gamma_2(k_1, k_2, m_2, m_3). \tag{H.0.7}$$

Bibliography

[IMU, 2019] (2019). IMU-P InertialLabs. https://inertiallabs.com/imup.html. Accessed: 2019-10-09. 144

[Abadi et al., 2019] Abadi, A., El Amraoui, A., Mekki, H., and Ramdani, N. (2019). Optimal trajectory generation and robust flatness–based tracking control of quadrotors. *Optimal Control Applications and Methods*, pages 1–22, Article in Press. 12

[Achtelik et al., 2011] Achtelik, M., Bierling, T., Wang, J., Höcht, L., and Holzapfel, F. (2011). Adaptive control of a quadcopter in the presence of large/complete parameter uncertainties. In *Infotech@ Aerospace 2011*, page 1485. 156

[Alamir and Bornard, 1994] Alamir, M. and Bornard, G. (1994). On the stability of receding horizon control of nonlinear discrete-time systems. *Systems & Control Letters*, 23(4):291–296. 7, 12

[Alamir and Murilo, 2008] Alamir, M. and Murilo, A. (2008). Swing-up and stabilization of a twin-pendulum under state and control constraints by a fast nmpc scheme. *Automatica*, 44(5):1319–1324. 12, 54

[Alexis et al., 2014] Alexis, K., Nikolakopoulos, G., and Tzes, A. (2014). Experimental constrained optimal attitude control of a quadrotor subject to wind disturbances. *International Journal of Control, Automation and Systems*, 12(6):1289–1302. 11

[Altug et al., 2002] Altug, E., Ostrowski, J. P., and Mahony, R. (2002). Control of a quadrotor helicopter using visual feedback. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 1, pages 72–77. IEEE. 1

[Alvarado et al., 2007] Alvarado, I., Limón, D., Alamo, T., and Camacho, E. F. (2007). Output feedback robust tube based mpc for tracking of piece-wise constant references. In *2007 46th IEEE Conference on Decision and Control*, pages 2175–2180. IEEE. 155

[Ata, 2007] Ata, A. A. (2007). Optimal trajectory planning of manipulators: a review. *Journal of Engineering Science and Technology*, 2(1):32–54. 6

[Avram et al., 2017] Avram, R. C., Zhang, X., and Muse, J. (2017). Quadrotor actuator fault diagnosis and accommodation using nonlinear adaptive estimators. *IEEE Transactions on Control Systems Technology*. 9, 11, 128

[Badgwell and Qin, 2015] Badgwell, T. A. and Qin, S. J. (2015). Model-predictive control in practice. *Encyclopedia of Systems and Control*, pages 756–760. 54, 79, 82

[Başak and Prempain, 2015] Başak, H. and Prempain, E. (2015). Switching recovery control of a quadcopter uav. In *European Control Conference (ECC)*, pages 3641–3646. IEEE. 128

[Bazin et al., 2008] Bazin, J.-C., Kweon, I., Demonceaux, C., and Vasseur, P. (2008). Uav attitude estimation by vanishing points in catadioptric images. In *2008 IEEE International Conference on Robotics and Automation*, pages 2743–2749. IEEE. 40

[Bemporad et al., 2009] Bemporad, A., Pascucci, C. A., and Rocchi, C. (2009). Hierarchical and hybrid model predictive control of quadcopter air vehicles. *IFAC Proceedings Volumes*, 42(17):14–19. 41

[Bemporad and Rocchi, 2011] Bemporad, A. and Rocchi, C. (2011). Decentralized hybrid model predictive control of a formation of unmanned aerial vehicles. *IFAC Proceedings Volumes*, 44(1):11900–11906. 156

[Bhatia and Davis, 1995] Bhatia, R. and Davis, C. (1995). A cauchy-schwarz inequality for operators with applications. *Linear algebra and its applications*, 223:119–129. 27

[Bipin et al., 2014] Bipin, K., Duggal, V., and Krishna, K. M. (2014). Autonomous navigation of generic quadrocopter with minimum time trajectory planning and control. In *Proceedings of the 2014 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 66–71. IEEE. 2, 7, 19, 24, 26

[Bipin et al., 2015] Bipin, K., Duggal, V., and Krishna, K. M. (2015). Autonomous navigation of generic monocular quadcopter in natural environment. In *Proceedings of the 2015 International Conference on Robotics and Automation (ICRA)*, pages 1063–1070. IEEE. 4, 7

[Boccia et al., 2014] Boccia, A., Grüne, L., and Worthmann, K. (2014). Stability and feasibility of state constrained mpc without stabilizing terminal constraints. *Systems & control letters*, 72:14–21. 114

[Bolandi et al., 2013] Bolandi, H., Rezaei, M., Mohsenipour, R., Nemati, H., and Smailzadeh, S. M. (2013). Attitude control of a quadrotor with optimized pid controller. *Intelligent Control and Automation*, 4(03):335. 11

[Borisov et al., 2017] Borisov, O. I., Bobtsov, A. A., Pyrkin, A. A., and Gromov, V. S. (2017). Simple adaptive control for quadcopters with saturated actuators. In *AIP Conference Proceedings*, volume 1798, page 020031. AIP Publishing. 9, 11

[Borowczyk et al., 2017] Borowczyk, A., Nguyen, D.-T., Nguyen, A. P.-V., Nguyen, D. Q., Saussié, D., and Le Ny, J. (2017). Autonomous landing of a quadcopter on a high-speed ground vehicle. *Journal of Guidance, Control, and Dynamics*, 40(9):2378–2385. 3

[Bottasso et al., 2009] Bottasso, C. L., Leonello, D., Maffezzoli, A., and Riccardi, F. (2009). A procedure for the identification of the inertial properties of small-size uavs. In *XX AIDAA Congress*, volume 3. 144

[Bouabdallah et al., 2004] Bouabdallah, S., Murrieri, P., and Siegwart, R. (2004). Design and control of an indoor micro quadrotor. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 5, pages 4393–4398. IEEE. 1, 9

[Bouktir et al., 2008] Bouktir, Y., Haddad, M., and Chettibi, T. (2008). Trajectory planning for a quadrotor helicopter. In *2008 Mediterranean Conference on Control and Automation*, pages 1258–1263. IEEE. 5

[Braslavsky and Middleton, 1996] Braslavsky, J. and Middleton, R. (1996). Global and semi-global stabilizability in certain cascade nonlinear systems. *IEEE transactions on automatic control*, 41(6):876–881. 89

[Cannon et al., 2003] Cannon, M., Deshmukh, V., and Kouvaritakis, B. (2003). Nonlinear model predictive control with polytopic invariant sets. *Automatica*, 39(8):1487–1494. 54, 61, 89, 101, 112

[Cao and Lynch, 2016] Cao, N. and Lynch, A. F. (2016). Inner–outer loop control for quadrotor uavs with input and state constraints. *IEEE Transactions on Control Systems Technology*, 24(5):1797–1804. 4, 20, 41, 82

[Carino et al., 2015] Carino, J., Abaunza, H., and Castillo, P. (2015). Quadrotor quaternion control. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 825–831. IEEE. 41, 53

[Chamseddine et al., 2012] Chamseddine, A., Zhang, Y., Rabbath, C., Join, C., and Theilliol, D. (2012). Flatness-based trajectory planning/replanning for a quadrotor unmanned aerial vehicle. *IEEE Transactions on Aerospace and Electronic Systems*, 48(4):2832–2848. 2, 4, 5, 6, 9, 11, 19, 24, 26, 33, 128

[Chand and Doty, 1985] Chand, S. and Doty, K. L. (1985). On-line polynomial trajectories for robot manipulators. *The International Journal of Robotics Research*, 4(2):38–48. 6

[Chang and Eun, 2014] Chang, D. E. and Eun, Y. (2014). Construction of an atlas for global flatness-based parameterization and dynamic feedback linearization of quadcopter dynamics. In *53rd IEEE Conference on Decision and Control*, pages 686–691. IEEE. 3

[Charlet et al., 1989] Charlet, B., Lévine, J., and Marino, R. (1989). On dynamic feedback linearization. *Systems & Control Letters*, 13(2):143–151. 156

[Chaturvedi et al., 2011] Chaturvedi, N. A., Sanyal, A. K., and McClamroch, N. H. (2011). Rigid-body attitude control. *IEEE control systems magazine*, 31(3):30–51. 22, 53

[Chen and Allgöwer, 1998] Chen, H. and Allgöwer, F. (1998). A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1217. 54, 55, 56, 57, 59, 61, 68, 69, 70, 81, 83, 89, 98, 99, 101, 110, 112, 121, 122, 125, 166

[Chen and Huzmezan, 2003] Chen, M. and Huzmezan, M. (2003). A combined mbpc/2 dof h infinity controller for a quad rotor uav. In *AIAA guidance, navigation, and control conference and exhibit*, page 5520. 9

[Chen and Jiang, 2005] Chen, W. and Jiang, J. (2005). Fault-tolerant control against stuck actuator faults. *IEE Proceedings-Control Theory and Applications*, 152(2):138–146. 12, 13, 128, 130

[Cheung et al., 2017] Cheung, K. K., Wagster IV, J. A., Tischler, M. B., Ivler, C. M., Berrios, M. G., Berger, T., and Lehmann, R. M. (2017). An overview of the us army aviation development directorate quadrotor guidance, navigation, and control project. In *Vertical Flight Society Forum*, volume 73. 2

[Choi and Ahn, 2014] Choi, Y.-C. and Ahn, H.-S. (2014). Nonlinear control of quadrotor for point tracking: Actual implementation and experimental tests. *IEEE/ASME transactions on mechatronics*, 20(3):1179–1192. 11

[Convens et al., 2017] Convens, B., Merckaert, K., Nicotra, M. M., Naldi, R., and Garone, E. (2017). Control of fully actuated unmanned aerial vehicles with actuator saturation. *IFAC-PapersOnLine*, 50(1):12715–12720. 9, 11

[Cowling et al., 2007] Cowling, I. D., Yakimenko, O. A., Whidborne, J. F., and Cooke, A. K. (2007). A prototype of an autonomous controller for a quadrotor uav. In *Proceedings of the IEEE European Control Conference (ECC)*, pages 4001–4008. 1, 3, 5, 6, 12, 19, 26, 39, 53, 83

[Cowling et al., 2010] Cowling, I. D., Yakimenko, O. A., Whidborne, J. F., and Cooke, A. K. (2010). Direct method based control system for an autonomous quadrotor. *Journal of Intelligent & Robotic Systems*, 60(2):285–316. 5, 6

[Craig, 2018] Craig, J. (2018). *Introduction to Robotics*. Pearson Education. 59, 60

[Craig, 2005] Craig, J. J. (2005). *Introduction to robotics: mechanics and control*, volume 3. Pearson/Prentice Hall Upper Saddle River, NJ, USA:. 10, 41, 42, 130

[Cutler and How, 2012] Cutler, M. and How, J. (2012). Actuator constrained trajectory generation and control for variable-pitch quadrotors. In *Proceedings of the 2012 AIAA Guidance, Navigation, and Control Conference*, pages 4777–4791. 1

[Cutler et al., 2011] Cutler, M., Ure, N.-K., Michini, B., and How, J. (2011). Comparison of fixed and variable pitch actuators for agile quadrotors. In *AIAA Guidance, Navigation, and Control Conference*, page 6406. 21

[Das et al., 2009] Das, A., Subbarao, K., and Lewis, F. (2009). Dynamic inversion with zero-dynamics stabilisation for quadrotor control. *IET control theory & applications*, 3(3):303–314. 11

[Do, 2015] Do, K. D. (2015). Coordination control of quadrotor vtol aircraft in three-dimensional space. *International Journal of Control*, 88(3):543–558. 11, 22

[Doban and Lazar, 2018] Doban, A. I. and Lazar, M. (2018). Computation of lyapunov functions for nonlinear differential equations via a massera-type construction. *IEEE Transactions on Automatic Control*, 63(5):1259–1272. 86, 101, 102

[Doniselli et al., 2002] Doniselli, C., Gobbi, M., and Mastinu, G. (2002). Measuring the inertia tensor of vehicles. *Vehicle System Dynamics*, 37(sup1):301–313. 144

[Egeland, 1986] Egeland, O. (1986). On the robustness of the computed torque technique in manipulator control. In *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 1203–1208. IEEE. 42

[El Assoudi et al., 2002] El Assoudi, A., El Yaagoubi, E., and Hammouri, H. (2002). Non-linear observer based on the euler discretization. *International Journal of Control*, 75(11):784–791. 143

[Eliker et al., 2018] Eliker, K., Zhang, G., Grouni, S., and Zhang, W. (2018). An optimization problem for quadcopter reference flight trajectory generation. *Journal of Advanced Transportation*, 2018. 4, 24, 33

[Engelhardt et al., 2016] Engelhardt, T., Konrad, T., Schäfer, B., and Abel, D. (2016). Flatness-based control for a quadrotor camera helicopter using model predictive control trajectory generation. In *Proceedings of the 24th Mediterranean Conference on Control and Automation (MED'16)*, pages 852–859, Athens, Greece. MCA, IEEE. 2, 5, 10, 12, 15, 19, 26, 40

[Eudes et al., 2018] Eudes, A., Marzat, J., Sanfourche, M., Moras, J., and Bertrand, S. (2018). Autonomous and safe inspection of an industrial warehouse by a multi-rotor mav. In *Field and Service Robotics*, pages 221–235. Springer. 3, 9, 10

[Fahroo and Ross, 2002] Fahroo, F. and Ross, I. M. (2002). Direct trajectory optimization by a chebyshev pseudospectral method. *Journal of Guidance, Control, and Dynamics*, 25(1):160–166. 6

[Falconí et al., 2016] Falconí, G. P., Angelov, J., and Holzapfel, F. (2016). Hexacopter outdoor flight test results using adaptive control allocation subject to an unknown complete loss of one propeller. In *2016 3rd Conference on Control and Fault-Tolerant Systems (SysTol)*, pages 373–380. IEEE. 9, 12

[Ferrara et al., 2013] Ferrara, A., Incremona, G. P., and Magni, L. (2013). A robust mpc/ism hierarchical multi-loop control scheme for robot manipulators. In *IEEE 52nd Annual Conference on Decision and Control (CDC)*, pages 3560–3565. IEEE. 54

[Fliess et al., 1995] Fliess, M., Lévine, J., and Rouchon, P. (1995). Flatness and defect of non-linear systems: introductory theory and examples. *International Journal of Control*, 61:1327–1361. 5, 19, 24, 25

[Folland, 1990] Folland, G. (1990). Remainder estimates in taylor's theorem. *The American Mathematical Monthly*, 97(3):233–235. 62

[Formentin and Lovera, 2011] Formentin, S. and Lovera, M. (2011). Flatness-based control of a quadrotor helicopter via feedforward linearization. In *Proceedings of the 50th IEEE Conference on Decision and Control (CDC-ECE'50)*, pages 6171–6176. 3, 4, 9, 10, 11, 15, 19, 21, 22, 24, 26, 33, 39, 40, 41, 42, 43

[Förster, 2015] Förster, J. (2015). System identification of the crazyflie 2.0 nano quadrocopter. B.S. thesis, ETH Zurich. 144

[Freddi et al., 2011] Freddi, A., Lanzon, A., and Longhi, S. (2011). A feedback linearization approach to fault tolerance in quadrotor vehicles. *IFAC Proceedings Volumes*, 44(1):5413–5418. 3, 9, 11, 12, 13, 15, 19, 22, 39, 40, 41, 42, 53, 128

[Freddi et al., 2010] Freddi, A., Longhi, S., and Monteriu, A. (2010). Actuator fault detection system for a mini-quadrotor. In *2010 IEEE International Symposium on Industrial Electronics*, pages 2055–2060. IEEE. 1, 2

[Gandolfo et al., 2016] Gandolfo, D. C., Salinas, L. R., Brandão, A., and Toibero, J. M. (2016). Stable path-following control for a quadrotor helicopter considering energy consumption. *IEEE Transactions on Control Systems Technology*, 25(4):1423–1430. 26

[Giernacki et al., 2017] Giernacki, W., Skwierczyński, M., Witwicki, W., Wroński, P., and Kozierski, P. (2017). Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering. In *22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 37–42. IEEE. 3, 8, 9, 10, 20, 47, 82, 123

[Giones and Brem, 2017] Giones, F. and Brem, A. (2017). From toys to tools: The co-evolution of technological and entrepreneurial developments in the drone industry. *Business Horizons*, 60(6):875–884. 2

[González-Jorge et al., 2017] González-Jorge, H., Martínez-Sánchez, J., Bueno, M., et al. (2017). Unmanned aerial systems for civil applications: A review. *Drones*, 1(1):2. 1, 2

[Gros et al., 2012] Gros, S., Quirynen, R., and Diehl, M. (2012). Aircraft control based on fast non-linear mpc & multiple-shooting. In *51st IEEE Conference on Decision and Control (CDC)*, pages 1142–1147. IEEE. 12, 80, 82, 151

[Grüne, 2009] Grüne, L. (2009). Analysis and design of unconstrained nonlinear mpc schemes for finite and infinite dimensional systems. *SIAM Journal on Control and Optimization*, 48(2):1206–1228. 122

[Grüne, 2012] Grüne, L. (2012). NMPC without terminal constraints. *4th IFAC Nonlinear Model Predictive Control Conference (NMPC 2012). IFAC Proceedings Volumes*, 45(17):1–13. 82, 110, 112, 114, 126

[Grüne and Pannek, 2011] Grüne, L. and Pannek, J. (2011). *Nonlinear Model Predictive Control: Theory and Algorithms*. Springer-Verlag, London. 7, 12, 111

[Gulan et al., 2017] Gulan, M., Takács, G., Nguyen, N. A., Olaru, S., Rodriguez-Ayerbe, P., and Rohal'-Ilkiv, B. (2017). Embedded linear model predictive control for 8-bit microcontrollers via convex lifting. *IFAC-PapersOnLine*, 50(1):10697–10704. 80, 124, 151

[Ha et al., 2014] Ha, C., Zuo, Z., Choi, F. B., and Lee, D. (2014). Passivity-based adaptive backstepping control of quadrotor-type uavs. *Robotics and Autonomous Systems*, 62(9):1305–1315. 10, 11, 19, 120, 156

[Hagenmeyer and Delaleau, 2003] Hagenmeyer, V. and Delaleau, E. (2003). Robustness analysis of exact feedforward linearization based on differential flatness. *Automatica*, 39(11):1941–1946. 6, 11

[Hager, 2000] Hager, W. W. (2000). Runge-kutta methods in optimal control and the transformed adjoint system. *Numerische Mathematik*, 87(2):247–282. 111

[Hart et al., 2011] Hart, W. E., Watson, J.-P., and Woodruff, D. L. (2011). Pyomo: modeling and solving mathematical programs in python. *Mathematical Programming Computation*, 3(3):219–260. 78, 123

[Hasan and Johansen, 2018] Hasan, A. and Johansen, T. A. (2018). Model-based actuator fault diagnosis in multirotor uavs. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1017–1024. IEEE. 1, 2, 9, 21, 22, 128, 129, 145, 155

[Hehn and D'Andrea, 2011] Hehn, M. and D'Andrea, R. (2011). Quadrocopter trajectory generation and control. *IFAC Proceedings Volumes*, 44(1):1485–1491. 5, 6, 10, 19, 32

[Hehn and D'Andrea, 2015] Hehn, M. and D'Andrea, R. (2015). Real-time trajectory generation for quadrocopters. *IEEE Transactions on Robotics*, 31(4):877–892. 4, 5, 6, 7, 8, 10, 15, 19, 22, 23, 24, 28, 32

[Hoffmann et al., 2007] Hoffmann, G., Huang, H., Waslander, S., and Tomlin, C. (2007). Quadrotor helicopter flight dynamics and control: Theory and experiment. In *AIAA guidance, navigation and control conference and exhibit*, page 6461. 1

[Hoffmann et al., 2008] Hoffmann, G., Waslander, S., and Tomlin, C. (2008). Quadrotor helicopter trajectory tracking control. In *AIAA guidance, navigation and control conference and exhibit*, page 7410. 4, 7, 11

[Houska et al., 2011] Houska, B., Ferreau, H. J., and Diehl, M. (2011). Acado toolkitan open-source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 32(3):298–312. 54

[Hua et al., 2009] Hua, M.-D., Hamel, T., Morin, P., and Samson, C. (2009). A control approach for thrust-propelled underactuated vehicles and its application to vtol drones. *IEEE Transactions on Automatic Control*, 54(8):1837–1853. 1, 3, 9, 10, 11

[Hua et al., 2013] Hua, M.-D., Hamel, T., Morin, P., and Samson, C. (2013). Introduction to feedback control of underactuated vtolvehicles: A review of basic control design ideas and principles. *IEEE Control Systems Magazine*, 33(1):61–75. 8

[Intwala and Parikh, 2015] Intwala, A. and Parikh, Y. (2015). A review on vertical take off and landing (vtol) vehicles. *International Journal of Innovative Research in Advanced Engineering (IJIRAE)*, 2(2):187–191. 1, 3, 24, 53

[Jiang et al., 2016] Jiang, B., Hu, Q., and Friswell, M. I. (2016). Fixed-time attitude control for rigid spacecraft with actuator saturation and faults. *IEEE Transactions on Control Systems Technology*, 24(5):1892–1898. 9, 12, 40, 128

[Jiang et al., 2012] Jiang, C., Xue, L., Chang, H., Yuan, G., and Yuan, W. (2012). Signal processing of mems gyroscope arrays to improve accuracy using a 1st order markov for rate signal modeling. *Sensors*, 12(2):1720–1737. 144

[Kaminer et al., 2006] Kaminer, I., Yakimenko, O., Pascoal, A., and Ghabcheloo, R. (2006). Path generation, path following and coordinated control for timecritical missions of multiple uavs. In *2006 American Control Conference*, pages 4906–4913. IEEE. 6

[Kerma et al., 2012] Kerma, M., Mokhtari, A., Abdelaziz, B., and Orlov, Y. (2012). Nonlinear h control of a quadrotor (uav), using high order sliding mode disturbance estimator. *International Journal of Control*, 85(12):1876–1885. 120

[Klausen et al., 2014] Klausen, K., Fossen, T. I., and Johansen, T. A. (2014). Suspended load motion control using multicopters. In *22nd Mediterranean Conference on Control and Automation*, pages 1371–1376. IEEE. 9

[Klausen et al., 2017] Klausen, K., Fossen, T. I., and Johansen, T. A. (2017). Nonlinear control with swing damping of a multirotor uav with suspended load. *Journal of Intelligent & Robotic Systems*, 88(2-4):379–394. 1, 3, 4

[Klausen et al., 2018] Klausen, K., Fossen, T. I., and Johansen, T. A. (2018). Autonomous recovery of a fixed-wing uav using a net suspended by two multirotor uavs. *Journal of Field Robotics*, 35(5):717–731. 2

[Kohler et al., 2018] Kohler, J., Muller, M. A., and Allgower, F. (2018). Nonlinear reference tracking: An economic model predictive control perspective. *IEEE Transactions on Automatic Control*. xii, 54, 61, 64, 82, 110, 111, 112, 114, 119, 121, 122, 126

[Kownacki, 2011] Kownacki, C. (2011). Optimization approach to adapt kalman filters for the real-time application of accelerometer and gyroscope signals' filtering. *Digital Signal Processing*, 21(1):131–140. 144

[Kuantama et al., 2018] Kuantama, E., Tarca, I., and Tarca, R. (2018). Feedback linearization lqr control for quadcopter position tracking. In *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 204–209. IEEE. 11, 19, 42

[Kuriki and Namerikawa, 2015] Kuriki, Y. and Namerikawa, T. (2015). Formation control with collision avoidance for a multi-uav system using decentralized mpc and consensus-based control. *SICE Journal of Control, Measurement, and System Integration*, 8(4):285–294. 156

[Landry et al., 2016] Landry, B., Deits, R., Florence, P. R., and Tedrake, R. (2016). Aggressive quadrotor flight through cluttered environments using mixed integer programming. In *International Conference on Robotics and Automation (ICRA)*, pages 1469–1475. IEEE. 46

[Lazar and Spinu, 2015] Lazar, M. and Spinu, V. (2015). Finite-step terminal ingredients for stabilizing model predictive control. *IFAC-PapersOnLine*, 48(23):9–15. 86, 101, 102

[Leishman, 2002] Leishman, J. G. (2002). The breguet-richet quad-rotor helicopter of 1907. *Vertiflite*, 47(3):58–60. 1

[Lévine, 2011] Lévine, J. (2011). On necessary and sufficient conditions for differential flatness. *Applicable Algebra in Engineering, Communication and Computing*, 22(1):47–90. 6, 24

[Lewis et al., 2003] Lewis, F. L., Dawson, D. M., and Abdallah, C. T. (2003). *Robot manipulator control: theory and practice*. CRC Press. 41, 54, 59, 60

[Limaverde Filho et al., 2016] Limaverde Filho, J. O. d. A., Lourenço, T. S., Fortaleza, E., Murilo, A., and Lopes, R. V. (2016). Trajectory tracking for a quadrotor system: A flatness-based nonlinear predictive control approach. In *IEEE Conference on Control Applications (CCA)*, pages 1380–1385. IEEE. 9, 12

[Limón et al., 2006] Limón, D., Alamo, T., Salas, F., and Camacho, E. F. (2006). On the stability of constrained mpc without terminal constraint. *IEEE transactions on automatic control*, 51(5):832–836. 114

[Liu et al., 2015] Liu, H., Wang, X., and Zhong, Y. (2015). Quaternion-based robust attitude control for uncertain robotic quadrotors. *IEEE Transactions on Industrial Informatics*, 11(2):406–415. 11

[Löfberg, 2004] Löfberg, J. (2004). Yalmip : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan. 35

[Lotufo et al., 2019] Lotufo, M. A., Colangelo, L., Perez-Montenegro, C., Canuto, E., and Novara, C. (2019). Uav quadrotor attitude control: An adrc-emc combined approach. *Control Engineering Practice*, 84:13–22. 11

[Lu et al., 2017] Lu, H., Liu, C., Guo, L., and Chen, W.-H. (2017). Constrained anti-disturbance control for a quadrotor based on differential flatness. *International Journal of Systems Science*, 48(6):1182–1193. 9, 10, 11, 12, 19, 23, 26, 39, 82, 83

[Luis and Ny, 2016] Luis, C. and Ny, J. L. (2016). Design of a trajectory tracking controller for a nanoquadcopter. Technical report, Mobile Robotics and Autonomous System Laboratory, Polytechnique Montreal. 34, 47, 48, 144, 147

[Magni and Scattolini, 2004] Magni, L. and Scattolini, R. (2004). Model predictive control of continuous-time nonlinear systems with piecewise constant control. *IEEE Transactions on Automatic Control*, 49(6):900–906. 58, 59, 89, 102

[Marchand and Alamir, 2003] Marchand, N. and Alamir, M. (2003). Receding horizon stabilization of a rigid spacecraft with two actuators. *Journal of dynamic systems, measurement, and control*, 125(3):489–491. 54

[Marzat et al., 2012] Marzat, J., Piet-Lahanier, H., Damongeot, F., and Walter, E. (2012). Model-based fault diagnosis for aerospace systems: a survey. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of aerospace engineering*, 226(10):1329–1360. 12

[Mayne, 2014] Mayne, D. Q. (2014). Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967–2986. 54

[Mayne and Kerrigan, 2007] Mayne, D. Q. and Kerrigan, E. C. (2007). Tube-based robust nonlinear model predictive control1. *IFAC Proceedings Volumes*, 40(12):36–41. 155

[Mayne et al., 2000] Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814. 7, 10, 12, 19, 53, 54, 55, 56, 57, 59, 81, 82, 107, 125, 166

[Mazenc and Praly, 1996] Mazenc, F. and Praly, L. (1996). Adding integrations, saturated controls, and stabilization for feedforward systems. *IEEE Transactions on Automatic Control*, 41(11):1559–1578. 68

[Meissen et al., 2017] Meissen, C., Klausen, K., Arcak, M., Fossen, T. I., and Packard, A. (2017). Passivity-based formation control for uavs with a suspended load. *IFAC-PapersOnLine*, 50(1):13150–13155. 156

[Mellinger and Kumar, 2011] Mellinger, D. and Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2520–2525, Shanghai, China. IEEE. 1, 4, 5, 6, 9, 10, 11, 19, 41

[Merabti et al., 2015] Merabti, H., Bouchachi, I., and Belarbi, K. (2015). Nonlinear model predictive control of quadcopter. In *2015 16th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, pages 208–211. IEEE. 11

[Mercy et al., 2017] Mercy, T., Van Parys, R., and Pipeleers, G. (2017). Spline-based motion planning for autonomous guided vehicles in a dynamic environment. *IEEE Transactions on Control Systems Technology*, 26(6):2182–2189. 7

[Meyer, 2000] Meyer, C. D. (2000). *Matrix analysis and applied linear algebra*, volume 71. Siam. 44, 63

[Mogili and Deepak, 2018] Mogili, U. R. and Deepak, B. (2018). Review on application of drone systems in precision agriculture. *Procedia computer science*, 133:502–509. 1, 2

[Monteiro et al., 2016] Monteiro, J. C., Lizarralde, F., and Hsu, L. (2016). Optimal control allocation of quadrotor uavs subject to actuator constraints. In *2016 American Control Conference (ACC)*, pages 500–505. IEEE. 9

[Mueller and D'Andrea, 2013] Mueller, M. W. and D'Andrea, R. (2013). A model predictive controller for quadrocopter state interception. In *Proceedings of the IEEE European Control Conference (ECC)*, pages 1383–1389. 5, 10, 12, 19, 26, 39, 82

[Mueller and D'Andrea, 2014] Mueller, M. W. and D'Andrea, R. (2014). Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers. In *International Conference on Robotics and Automation (ICRA)*, pages 45–52. IEEE. 9, 12, 40, 53, 128

[Muñoz et al., 2017] Muñoz, F., González-Hernández, I., Salazar, S., Espinoza, E. S., and Lozano, R. (2017). Second order sliding mode controllers for altitude control of a quadrotor uas: Real-time implementation in outdoor environments. *Neurocomputing*, 233:61–71. 11

[Nascimento and Saska, 2019] Nascimento, T. P. and Saska, M. (2019). Position and attitude control of multi-rotor aerial vehicles: A survey. *Annual Reviews in Control*. 1, 2, 3, 8, 11

[Navabi and Soleymanpour, 2017] Navabi, M. and Soleymanpour, S. (2017). Immersion and invariance based adaptive control of aerial robot in presence of inertia uncertainty. In *2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, pages 0959–0964. IEEE. 156

[Nex and Remondino, 2014] Nex, F. and Remondino, F. (2014). Uav for 3d mapping applications: a review. *Applied geomatics*, 6(1):1–15. 1, 53

[Nguyen et al., 2020a] Nguyen, H. T., Nguyen, N. T., and Prodan, I. (2020a). Trajectory tracking for a multicopter under a quaternion representation. *Accepted for publication at the 2020 IFAC World Congress*. 53

[Nguyen et al., 2017a] Nguyen, N. T., Prodan, I., and Lefèvre, L. (2017a). Multi-layer optimization-based control design for quadcopter trajectory tracking. In *Proceedings of the 25th IEEE Mediterranean Conference on Control and Automation (MED'17)*, pages 601–606, Valletta, Malta. MCA, IEEE. 9, 10, 11, 14, 19, 82

[Nguyen et al., 2018a] Nguyen, N. T., Prodan, I., and Lefèvre, L. (2018a). Effective angular constrained trajectory generation for thrust-propelled vehicles. In *Proceedings of the 16th European Control Conference (ECC'18)*, pages 1833–1838, Limassol, Cyprus. EUCA, IEEE. 4, 6, 14, 20, 26, 27, 82

[Nguyen et al., 2018b] Nguyen, N. T., Prodan, I., and Lefèvre, L. (2018b). Flat trajectory design and tracking with saturation guarantees: a nano-drone application. *International Journal of Control*, pages 1–14, Article in Press. x, 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 14, 15, 19, 20, 24, 26, 29, 30, 32, 39, 42, 43, 44, 46, 49, 50, 82, 98, 123

[Nguyen et al., 2019a] Nguyen, N. T., Prodan, I., and Lefèvre, L. (2019a). On the use of a computed-torque control law for the terminal region of an nmpc scheme. In *Proceedings of the 2019 American Control Conference (ACC19)*, pages 1008–1013, Philadelphia, USA. IEEE. 61, 62

[Nguyen et al., 2019b] Nguyen, N. T., Prodan, I., and Lefèvre, L. (2019b). A stablizing nmpc design for thrust-propelled vehicles dynamics via feedback linearization. In *Proceedings of the 2019 American Control Conference (ACC19)*, pages 2909–2914, Philadelphia, USA. IEEE. 2, 4, 10, 11, 14, 19, 54, 61, 85, 89, 91

[Nguyen et al., 2020b] Nguyen, N. T., Prodan, I., and Lefèvre, L. (2020b). Multicopter attitude control through nmpc design with guaranteed stability. *Accepted for publication at the 2020 IFAC World Congress*. 73, 74, 75, 76, 77

[Nguyen et al., 2020c] Nguyen, N. T., Prodan, I., and Lefèvre, L. (2020c). Stability guarantees for translational thrust-propelled vehicles dynamics through nmpc designs. *IEEE Transactions on Control Systems Technology*, pages 1–13, Article in Press. xi, xii, 14, 113, 114, 115, 118

[Nguyen et al., TSMC] Nguyen, N. T., Prodan, I., and Lefèvre, L. (TSMC). Stabilising vtol system through an nmpc design with a relaxed terminal region. *Submitted to IEEE Transactions on Systems, Man, and Cybernetics: Systems*. 14

[Nguyen et al., 2020d] Nguyen, N. T., Prodan, I., Petzke, F., Streif, S., and Lefèvre, L. (2020d). Hierarchical control of a quadcopter under stuck actuator fault. *Accepted for publication at the 2020 IFAC World Congress*. 128, 129, 132, 133, 139, 144, 146, 147

[Nguyen et al., 2017b] Nguyen, N. T., Prodan, I., Stoican, F., and Lefèvre, L. (2017b). Reliable nonlinear control for quadcopter trajectory tracking through differential flatness. *IFAC-PapersOnLine*, 50(1):6971–6976. 2, 3, 9, 10, 11, 12, 13, 15, 19, 22, 24, 26, 39, 40, 41, 42, 43, 53, 74, 128, 129

[Passaro et al., 2017] Passaro, V., Cuccovillo, A., Vaiani, L., De Carlo, M., and Campanella, C. E. (2017). Gyroscope technology and applications: A review in the industrial perspective. *Sensors*, 17(10):2284. 144

[Piegl and Tiller, 1995] Piegl, L. and Tiller, W. (1995). B-spline curves and surfaces. In *The NURBS Book*, pages 81–116. Springer. 4, 5, 7, 24, 29, 30

[Poignet and Gautier, 2000] Poignet, P. and Gautier, M. (2000). Nonlinear model predictive control of a robot manipulator. In *Proceedings. 6th International Workshop on Advanced Motion Control*, pages 401–406. IEEE. 54

[Prabha et al., 2016] Prabha, M., Thottungal, R., and Kaliappan, S. (2016). Modeling and simulation of x quadcopter control. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, 4:282–287. 129

[Prach and Kayacan, 2018] Prach, A. and Kayacan, E. (2018). An mpc-based position controller for a tilt-rotor tricopter vtol uav. *Optimal Control Applications and Methods*, 39(1):343–356. 23, 128

[Pretorius and Boje, 2014] Pretorius, A. and Boje, E. (2014). Design and modelling of a quadrotor helicopter with variable pitch rotors for aggressive manoeuvres. *IFAC Proceedings Volumes*, 47(3):12208–12213. 1

[Previati et al., 2009] Previati, G., Gobbi, M., and Mastinu, G. (2009). Improved measurement method for the identification of the centre of gravity location and of the inertia tensor of rigid bodies. In *ASME 2008 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 841–850. American Society of Mechanical Engineers Digital Collection. 144

[Prodan et al., 2013] Prodan, I., Olaru, S., Bencatel, R., de Sousa, J. B., Stoica, C., and Niculescu, S.-I. (2013). Receding horizon flight control for trajectory tracking of autonomous aerial vehicles. *Control Engineering Practice*, 21(10):13341349. 4, 5, 6, 7, 12, 15, 19, 29, 32, 156

[Prodan et al., 2015] Prodan, I., Stoican, F., Olaru, S., and Niculescu, S.-I. (2015). *Mixed-integer representations in control design: Mathematical foundations and applications*. Springer. 36

[Rawlings et al., 2012] Rawlings, J. B., Angeli, D., and Bates, C. N. (2012). Fundamentals of economic model predictive control. In *2012 IEEE 51st IEEE conference on decision and control (CDC)*, pages 3851–3861. IEEE. 7

[Reble and Allgöwer, 2012] Reble, M. and Allgöwer, F. (2012). Unconstrained model predictive control and suboptimality estimates for nonlinear continuous-time systems. *Automatica*, 48(8):1812–1817. 59, 114

[Reinhardt and Johansen, 2019] Reinhardt, D. and Johansen, T. A. (2019). Nonlinear model predictive attitude control for fixed-wing unmanned aerial vehicle based on a wind frame formulation. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 503–512. IEEE. 1, 54

[Rivera and Sawodny, 2010] Rivera, G. and Sawodny, O. (2010). Flatness-based tracking control and nonlinear observer for a micro aerial quadcopter. In *Proceedings of the 2010 International Conference of Numerical Analysis and Applied Mathematics (ICNAAM)*, volume 1281, pages 386–389. AIP Publishing. 19, 24, 26

[Rousseau et al., 2019] Rousseau, G., Maniu, C. S., Tebbani, S., Babel, M., and Martin, N. (2019). Minimum-time b-spline trajectories with corridor constraints. application to cinematographic quadrotor flight plans. *Control Engineering Practice*, 89:190–203. 4, 5, 7

[Roza and Maggiore, 2012] Roza, A. and Maggiore, M. (2012). Path following controller for a quadrotor helicopter. In *2012 American Control Conference (ACC)*, pages 4655–4660. IEEE. 2, 9, 10, 19, 41

[Roza and Maggiore, 2014] Roza, A. and Maggiore, M. (2014). A class of position controllers for underactuated vtol vehicles. *IEEE Transactions on Automatic Control*, 59(9):2580–2585. 10, 21

[Rucco et al., 2015] Rucco, A., Aguiar, A. P., Fontes, F. A., Pereira, F. L., and de Sousa, J. B. (2015). A model predictive control-based architecture for cooperative path-following of multiple unmanned aerial vehicles. In *Developments in model-based optimization and control*, pages 141–160. Springer. 1, 4, 54, 156

[Rucco et al., 2016] Rucco, A., Aguiar, A. P., Pereira, F. L., and de Sousa, J. B. (2016). A predictive path-following approach for fixed-wing unmanned aerial vehicles in presence of wind disturbances. In *Robot 2015: Second Iberian Robotics Conference*, pages 623–634. Springer. 1, 2

[Saied et al., 2015] Saied, M., Lussier, B., Fantoni, I., Francis, C., Shraim, H., and Sanahuja, G. (2015). Fault diagnosis and fault-tolerant control strategy for rotor failure in an octorotor. In *2015 IEEE International Conference on Robotics and Automation (ICRA'15)*, pages 5266–5271. IEEE. 1, 128

[Scattolini, 2009] Scattolini, R. (2009). Architectures for distributed and hierarchical model predictive control–a review. *Journal of process control*, 19(5):723–731. 41, 132, 156

[Shao et al., 2018] Shao, X., Hu, Q., Shi, Y., and Jiang, B. (2018). Fault-tolerant prescribed performance attitude tracking control for spacecraft under input saturation. *IEEE Transactions on Control Systems Technology*. 9, 128, 130

[Silano et al., 2019] Silano, G., Oppido, P., and Iannelli, L. (2019). Software-in-the-loop simulation for improving flight control system design: a quadrotor case study. In *Proceedings of the 2019 International Conference on Systems, Man, and Cybernetics (SMC'19)*. IEEE. 14

[Simon et al., 2013] Simon, D., Lofberg, J., and Glad, T. (2013). Nonlinear model predictive control using feedback linearization and local inner convex constraint approximations. In *2013 European Control Conference (ECC)*, pages 2056–2061. IEEE. 89

[Singh and Fuller, 2001] Singh, L. and Fuller, J. (2001). Trajectory generation for a uav in urban terrain, using nonlinear mpc. In *Proceedings of the 2001 American Control Conference.(Cat. No. 01CH37148)*, volume 3, pages 2301–2308. IEEE. 5

[Spong, 1994] Spong, M. W. (1994). Partial feedback linearization of underactuated mechanical systems. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*, volume 1, pages 314–321. IEEE. 156

[Srinivasan et al., 2009] Srinivasan, B., Huguenin, P., and Bonvin, D. (2009). Global stabilization of an inverted pendulum–control strategy and experimental verification. *Automatica*, 45(1):265–269. 68

[Stoican et al., 2016] Stoican, F., Ivănuçcă, V.-M., Prodan, I., and Popescu, D. (2016). Obstacle avoidance via b-spline parametrizations of flat trajectories. In *Proceedings of the 24th Mediterranean Conference on Control and Automation (MED'16)*, 10.1109/MED.2016.7536053, pages 1002–1007, Athens, Greece. MCA, IEEE. 156

[Stoican et al., 2019] Stoican, F., Prodan, I., Grøtli, E. I., and Nguyen, N. T. (2019). Inspection trajectory planning for 3d structures under a mixed-integer framework. In *Proceedings of the 2019 IEEE International Conference on Control & Automation (ICCA'19)*, pages 1349–1354. IEEE. 3, 20, 36

[Stoican et al., 2015] Stoican, F., Prodan, I., and Popescu, D. (2015). Flat trajectory generation for way-points relaxations and obstacle avoidance. In *Proceedings of the 23th Mediterranean Conference on Control and Automation (MED'15)*, pages 695–700. MCA, IEEE. 5, 7, 19, 24, 29, 30, 33

[Stoican et al., 2017] Stoican, F., Prodan, I., Popescu, D., and Ichim, L. (2017). Constrained trajectory generation for uav systems using a b-spline parametrization. In *Proceedings of the 25th Mediterranean Conference on Control and Automation (MED'17)*, pages 613–618, Valletta, Malta. MCA, IEEE. 4, 5, 6, 7, 19, 24, 29, 32, 33

[Stoican and Prodan and Grøtli and Nguyen, 2019] Stoican and Prodan and Grøtli and Nguyen (2019). Inspection trajectory planning for 3d structures under a mixed-integer framework. In *Proceedings of the 2019 IEEE International Conference on Control & Automation (ICCA'19)*, pages 1349–1354. IEEE. 14

[Sun et al., 2018] Sun, S., Sijbers, L., Wang, X., and de Visser, C. (2018). High-speed flight of quadrotor despite loss of single rotor. *IEEE Robotics and Automation Letters*, 3(4):3201–3207. 9, 12, 128

[Suryawan, 2011] Suryawan, F. (2011). *Constrained Trajectory Generation and Fault Tolerant Control Based on Differential Flatness and B-splines*. PhD thesis, School of Electrical Engineering and Computer Science, The University of Newcastle, Australia. 30

[Suryawan et al., 2011] Suryawan, F., De Doná, J., and Seron, M. (2011). Minimum-time trajectory generation for constrained linear systems using flatness and b-splines. *International Journal of Control*, 84(9):1565–1585. 7

[Tayebi and McGilvray, 2006] Tayebi, A. and McGilvray, S. (2006). Attitude stabilization of a vtol quadrotor aircraft. *IEEE Transactions on control systems technology*, 14(3):562–571. 53

[Teel, 1992] Teel, A. R. (1992). Global stabilization and restricted tracking for multiple integrators with bounded controls. *Systems & control letters*, 18(3):165–171. 20, 42, 44, 45, 159

[Uebel et al., 1992] Uebel, M., Minis, I., and Cleary, K. (1992). Improved computed torque control for industrial robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 528–533. IEEE. 54, 59

[Valencia-Palomo and Rossiter, 2010] Valencia-Palomo, G. and Rossiter, J. (2010). Using laguerre functions to improve efficiency of multi-parametric predictive control. In *Proceedings of the 2010 American Control Conference*, pages 4731–4736. IEEE. 6

[Wächter and Biegler, 2006] Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57. xii, 54, 70, 78, 80, 123, 147

[Wang et al., 2017] Wang, Y., Ramirez-Jaime, A., Xu, F., and Puig, V. (2017). Nonlinear model predictive control with constraint satisfactions for a quadcopter. In *Journal of Physics: Conference Series*, volume 783, page 012025. IOP Publishing. 9

[Wildhagen et al., 2019] Wildhagen, S., Müller, M. A., and Allgöwer, F. (2019). Economic mpc using a cyclic horizon with application to networked control systems. *arXiv preprint arXiv:1902.08132*. 102

[Wu and Lu, 2004] Wu, F. and Lu, B. (2004). Anti-windup control design for exponentially unstable lti systems with actuator saturation. *Systems & Control Letters*, 52(3):305–322. 20

[Xu, 2017] Xu, B. (2017). Composite learning finite-time control with application to quadrotors. *IEEE Transactions on systems, man, and cybernetics: systems*, 48(10):1806–1815. 39, 40, 42

[Yang and Lum, 2003] Yang, G.-H. and Lum, K.-Y. (2003). Fault-tolerant flight tracking control with stuck faults. In *Proceedings of the 2003 American Control Conference*, volume 1, pages 521–526. IEEE. 12, 128, 130

[Yu et al., 2013] Yu, S., Maier, C., Chen, H., and Allgöwer, F. (2013). Tube mpc scheme based on robust control invariant set with application to lipschitz nonlinear systems. *Systems & Control Letters*, 62(2):194–200. 155

[Yu et al., 2015] Yu, Y., Yang, S., Wang, M., Li, C., and Li, Z. (2015). High performance full attitude control of a quadrotor on so (3). In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1698–1703. IEEE. 11

[Zagaris et al., 2004] Zagaris, A., Kaper, H. G., and Kaper, T. J. (2004). Fast and slow dynamics for the computational singular perturbation method. *Multiscale Modeling & Simulation*, 2(4):613–638. 3

[Zanelli et al., 2018] Zanelli, A., Horn, G., Frison, G., and Diehl, M. (2018). Nonlinear model predictive control of a human-sized quadrotor. In *2018 European Control Conference (ECC)*, pages 1542–1547. IEEE. 10, 12, 15, 19, 80, 82, 124, 151

[Zhao and Go, 2014] Zhao, W. and Go, T. H. (2014). Quadcopter formation flight control combining mpc and robust feedback linearization. *Journal of the Franklin Institute*, 351(3):1335–1355. 19, 21, 22, 39, 40, 42, 43

[Ziegler, 2012] Ziegler, G. M. (2012). *Lectures on polytopes*, volume 152. Springer Science & Business Media. 32

Abstract

The goal of this thesis is to propose reliable control laws for the motion planning of a multicopter system under constraints and unexpected events (e.g., actuator faults). A hierarchical control architecture which decouples the scheme into position and attitude control is proposed. At the high level the position controller calculates the position error and provides the desired thrust and angles to the attitude controller at the low level to stabilize the system around the desired angles. The scheme's reliability (i.e., ensuring feasibility, stability and constraint validation) is done through a coherent merging of differential flatness, feedback linearization and Nonlinear Model Predictive Control (NMPC). Hence, the main thesis contributions lie in:
i) The analysis and design of bounds which characterize the various inputs and states of the system (angle position and velocity, torques, etc.). These are subsequently applied for constrained trajectory design (which combines differential flatness and feedback linearization through the use of B-spline parametrizations).
ii) Designs which exploit the "computed-torque control law" as local control within an NMPC with recursive feasibility guarantees. We show that avoiding the standard linearizations employed for nonlinear dynamics improves performance (in the sense of reducing the prediction horizon, enlarging the terminal region and reducing the problem's complexity). Further advances relax the requirement of set invariance and even discard the need for terminal stabilizing constraints. Generalizations for similar feedback linearizable systems are discussed.
iii) A hierarchical optimization-based FTC (Fault Tolerant Control) scheme to counteract a stuck rotor fault. This is done through control reconfiguration at both high and low levels, coupled with a fault diagnosis mechanism capable of handling fault detection, isolation and estimation. The results are validated over extensive simulations and laboratory experiments involving a nano-quadcopter.

Résumé

Le but de cette thèse est de proposer des lois de commande fiables pour la planification du mouvement d'un système multicoptère sous contraintes et événements inattendus (par exemple, des défauts d'actionneur). Une architecture hiérarchique qui sépare la commande en position et attitude est proposée. Au niveau haut, l'erreur de position est calculé et la poussée, ainsi que les angles désirés qui en découlent sont fournis au contrôleur d'attitude de niveau bas qui stabilise le système autour des angles désirés. La fiabilité du système est assurée par une combinaison cohérente des commandes par de platitude différentielle, par linéarisation et prédictive nonlinéaire (NMPC). Les principales contributions de la thèse sont les suivantes:
i) La caractérisation des contraintes sur les entrées et états du système (position et vitesse de l'angle, etc). Celles-ci sont ensuite appliquées pour la conception de trajectoires contraintes (combinant la platitude differentielle et la linéarisation par feedback via l'utilisation d'une paramétrisation des trajectoires poursuivies par B-splines.
ii) Des conceptions de commande de type NMPC pour un système obtenu après linéarisation par feedback (Computed Torque Control) avec des garanties de faisabilité récursives. Nous montrons en particlier qu'éviter les linéarisations usuelles de la dynamique améliore les performances (i.e. réduit l'horizon de prédiction, élargit la région terminale et réduit la complexité du problème). Des améliorations proposées permettent ensuite d'assouplir l'exigence d'invariance d'ensemble et éliminent la nécessité de contraintes de stabilisation terminales. Des généralisations pour des systèmes linéarisables similaires sont discutées.
iii) Un schéma hiérarchique de commande tolérante aux pannes de rotor (rotor bloqué).
Les résultats sont validés par des simulations et des expériences de laboratoire impliquant un nano-quadricoptère.