Robust trajectory planning algorithms for robots with parametric uncertainties

Pascal Brault

COLLEGE DOCTORAL BRETAGNE

MATHS, TELECOMS
INFORMATIQUE, SIGNAL
SYSTEMES, ELECTRONIQUE

Université de Rennes

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES

ÉCOLE DOCTORALE N° 601
Mathématiques, Télécommunications, Informatique, Signal, Systèmes, Électronique
Spécialité : *Automatique, Productique et Robotique*

Par
Pascal BRAULT

Algorithmes de planification de trajectoires robustes pour des tâches robotiques en présence d'incertitudes paramétriques

Thèse présentée et soutenue à Rennes, le 27/04/2023
Unité de recherche : IRISA – UMR 6074

Rapporteurs avant soutenance :

Isabelle FANTONI Directrice de Recherche CNRS, LS2N, Nantes
Cristian SECCHI Full Professor in Robotics, University of Modena and Reggio Emilia, Italy

Composition du Jury :

Attention, en cas d'absence d'un des membres du Jury le jour de la soutenance, la composition du jury doit être revue pour s'assurer qu'elle est conforme et devra être répercutée sur la couverture de thèse

Président :	François CHAUMETTE	Directeur de Recherche Inria, Rennes
Examinateurs :	Isabelle FANTONI	Directrice de Recherche CNRS, LS2N, Nantes
	Cristian SECCHI	Full Professor in Robotics, University of Modena and Reggio Emilia, Italy
	Fabio MORBIDI	Maître de Conférences, Université de Picardie Jules Verne, Amiens
	François CHAUMETTE	Directeur de Recherche Inria, Rennes
Dir. de thèse :	Paolo ROBUFFO GIORDANO	Directeur de Recherche CNRS, IRISA, Rennes

ACKNOWLEDGEMENTS

My sincerest gratitude goes out to all those who have aided me throughout the contemplation and elaboration of this thesis, offering their expertise, encouragement, and constructive feedback. Without their invaluable contributions, this accomplishment would not have been possible.

First of all, I would like to thank my Ph.D. supervisor, Dr. Paolo Robuffo Giordano, for graciously accepting me into his team and for generously devoting his time, pedagogical and unwavering scientific expertise, candidness, and amiability towards me. It was an honor to work alongside him, and I am immensely grateful for all the knowledge I have gained under his guidance.

My warmest thanks to my (former) Ph.D. co-supervisor, Dr. Quentin Delamare, who mentored me during my master thesis internship, which served as the foundation for this thesis. I would like to express my heartfelt appreciation for his attentiveness to my work, his sage guidance, and his listening skills, all of which played an important role in the success of this project. His dynamic energy and unwavering support served as sources of inspiration for me, and it was a pleasure to collaborate with him.

Special thanks to Mr. Christoph Böhm, for offering to work together to combine our two approaches. Thanks to Mr. Stephan Weiss, his supervisor, for supporting us in this contribution. Christoph, I hope we will pursue interesting studies together!

I also wish to thank the Rainbow team, for the good atmosphere that prevails. Special thanks to my office colleagues. Thank you Samuel, it was a real pleasure to share this office with you during all these years. Thank you for your support, your advices, the sport/night gaming sessions, and the mental walks done together! Also thank you Cédric for the great discussions, your enthousiasm in biology, the music you shared with me, and also a huge thank you for your housewarming party! Thank you Ali for all the scientific/philosophical talks, having you by my side was a must, and I wish you every success!

Although the experiments I carried out during this work did not make it into the manuscript, they would not have been possible without the various engineers who maintain the robot rooms, drones, code, etc. Thus, I would like to thank Mr. Fabien Spindler and Mr. Joudy Nader for their big contribution to my experiments. Thanks also to Mr. Anthony Mallet for his help on

GenoM3 and TeleKyb3.

Thank you to all my friends who gave me precious moral support. Big thanks to all the friends I have met in preparation school, or after, at the ENS de Rennes. Special thanks to 'les KSOS' who are a real family to me. Thank you for the trips, the parties, the festivals, etc. This is only the beginning! Also, thanks to 'SOCISSE', which, beyond sport, are true friends.

Furthermore, I must express my gratitude to the world of sports, which has played an integral role in my intellectual growth. My heartfelt appreciation goes out to street workout/calisthenics, for igniting my imagination and inspiring me continuously throughout the years, as well as to handbalancing, which has allowed me to push my boundaries with so little resources. I firmly believe that without the influence of sports, I would not have accomplished all that I have thus far.

I extend my gratitude to the world of music, which has been a constant companion throughout my journey. From classical to techno (or tekno), from rap to hardwave, and countless other genres, music has provided me with a mental escape that I cherish deeply.

Last but not least, a tremendous thank you to my family. Without my family and the education I received from it, my journey would never have been possible. Sorry for not being very present for the past few years... My parents and siblings, even though they may not always comprehend everything I say/do, have been an unwavering source of motivation and support.

My last thank you is of course for you, Pauline. Despite life's complex highs and lows, you remain a beacon of positivity, always radiating joy. I am happy and grateful to be able to sail on this adventure alongside you.

TABLE OF CONTENTS

RÉSUMÉ EN FRANÇAIS

Au cours des siècles passés, la société a connu des changements importants, grâce à des percées majeures dans les domaines de la science et de l'industrie. Les nouvelles technologies de la mécanique et de l'électronique ont permis de substituer la main d'œuvre humaine par des machines automatisées. La standardisation de la robotique a pour vocation de porter cette évolution à un niveau supérieur, en remplaçant l'intervention humaine dans le monde physique, même dans des domaines de notre vie qui étaient autrefois considérés comme triviaux, tels que la conduite automobile, le nettoyage et l'aménagement paysager. Les robots sont d'ores et déjà exploités pour effectuer des tâches subalternes, physiquement ardues, répétitives et monotones. En outre, ils peuvent également dépasser les capacités humaines en opérant dans des environnements dangereux ou inaccessibles. D'un point de vue économique, le recours à la main-d'œuvre robotique est avantageux, car les machines peuvent fonctionner en continu.

À mesure que les robots se répandent dans nos industries et dans notre vie quotidienne, ils ont la capacité de transformer fondamentalement notre relation au travail. Toutefois, le plein potentiel de la robotique dans la société n'est pas encore atteint et requiert une recherche, une ingénierie et une innovation plus poussées, pour mieux comprendre et étendre ses applications.

L'un des défis majeurs des systèmes automatisés réside dans la nécessité de fonctionner dans des conditions réelles, donc incertaines. Étant donné que les décisions des robots sont fondées sur certains modèles, les décrivant eux-mêmes, ainsi que leur environnement, la perfection mathématique de ces modèles est inévitablement limitée lorsqu'il s'agit de décrire la réalité. Un problème courant est celui de l'incertitude paramétrique, qui peut se présenter lors de l'exécution d'une tâche de mouvement, par exemple lorsque qu'un robot mobile doit atteindre un endroit précis pour effectuer une tâche telle que la saisie d'un objet. Dans le cas où le modèle employé est valide, mais alimenté par des paramètres erronés, la réalisation de la tâche peut différer du comportement *nominal* attendu.

Pour surmonter ce problème fondamental inhérent à la robotique, la principale solution a été de concevoir des contrôleurs qui le traitent spécifiquement tout en assurant un comportement stable, même en cas de déviation des paramètres. Une première approche classique consiste à développer des contrôleurs robustes, c'est-à-dire des lois de commande statiques qui garantissent un certain degré d'insensibilité, pour une plage limitée d'incertitude paramétrique. Tou-

tefois, ces lois doivent être spécifiquement adaptées au système en question. Les méthodes fondées sur la passivité, qui peuvent également être incluses dans la commande robuste, exploitent des propriétés énergétiques structurelles du système, et invariantes aux possibles déviations des paramètres.

Une approche alternative consiste à effectuer une estimation en ligne des paramètres tout en effectuant l'action de contrôle. Cette méthode implique la transmission d'une mise à jour de la mesure à la commande, cette dernière intégrant une dynamique supplémentaire pour tenir compte des changements. Toutefois, l'estimation des paramètres en ligne n'est pas une tâche aisée, car elle nécessite d'exciter suffisamment la dynamique du robot, ce qui peut entrer en conflit avec la tâche principale de navigation. En outre, la dynamique de l'estimation peut produire des régimes transitoires indésirables, ce qui augmente les risques de compromettre la stabilité du système couplé à l'estimation.

Dans les cas mentionnés précédemment, la robustesse est souvent obtenue au détriment de la précision/performance de la tâche. En effet, un compromis est inévitable : soit on opte pour un contrôleur parfait, qui assure une grande précision de la tâche de suivi, mais qui repose sur une très bonne connaissance des paramètres du modèle (ce qui le rend très vulnérable aux perturbations), soit on préfère un contrôleur qui garantit un certain degré de robustesse face aux perturbations, mais qui ne permet souvent qu'une *stabilisation pratique*.

Un autre point de vue inverse la philosophie de conception en réponse aux incertitudes paramétriques : plutôt que de se concentrer sur des commandes sophistiquées, le couple système/contrôleur est laissé tel quel, avec ses avantages et ses inconvénients, et l'accent est mis sur l'optimisation des trajectoires, afin d'améliorer les performances des systèmes incertains. Dans cette optique, des travaux récents ont porté sur la génération de trajectoires dite *feedforward*, destinées à minimiser la *sensibilité de l'état* en 'boucle ouverte' d'un système.

Plus récemment, ces travaux ont été étendus à la planification de trajectoires ayant une sensibilité d'état minimale en 'boucle fermée'. Cette nouvelle recherche dite *control-aware* intègre directement le couplage du système/contrôleur dans l'optimisation. Ainsi, un choix adéquat de la forme de la consigne de mouvement permet d'assurer la robustesse de la navigation face aux incertitudes paramétriques. Il est probable que cette approche soit bénéfique pour de nombreuses applications futures, et nous pensons qu'elle mérite d'être développée davantage.

À cet égard, l'objectif principal de cette thèse est d'explorer les possibilités et les limites offertes par la génération de trajectoires robustes et *control-aware*, vis-à-vis des incertitudes paramétriques. Le problème de recherche est énoncé en termes plus explicites comme étant la conception (et l'implémentation) de nouveaux algorithmes de planification de trajectoire pour

les quadricoptères. La validité des méthodes logicielles présentées est toujours évaluée empiriquement par des campagnes statistiques étendues de simulations.

Structure de la thèse

Cette thèse débute par l'introduction du contexte scientifique (Chapitre 1) qui justifie les études menées dans les différentes contributions. Le Chapitre 2 vise à réaliser une revue de la littérature scientifique sur la génération de trajectoires robustes pour des systèmes à paramètres incertains. Dans un premier temps, nous examinons les différentes configurations mécaniques de robots aériens, puis nous étudions la robustesse des stratégies de contrôle des quadricoptères. Ensuite, l'intéret est porté sur l'estimation de l'état, puis des paramètres en tant que sous-problème de l'estimation d'état. Après cela, nous nous concentrons sur la génération de chemins, puis de trajectoires robotiques. Enfin, nous examinons les techniques de planification de références robustes pour des systèmes à paramètres incertains. Après avoir revu l'état de l'art, le Chapitre 3 se consacre à établir les fondamentaux mathématiques exploités tout au long du manuscrit. En particulier, un modèle dynamique de quadricoptère 3D est détaillé. En disposant de ces outils, nous sommes prêts à aborder les contributions présentées dans cette thèse.

Dans notre première contribution (Chapitre 4), nous proposons une extension des algorithmes de planification de trajectoires à sensibilité minimale, en introduisant la sensibilité de l'entrée, qui, une fois réduite, conduit à une prédiction accrue de l'entrée du système. Nous formulons ensuite un problème d'optimisation qui combine les deux sensibilités en un objectif unique. Nous avons également remplacé la représentation de la trajectoire par des courbes de Bézier, qui offrent une meilleure stabilité numérique que les polynômes ordinaires. Pour évaluer le bien-fondé de notre stratégie, nous réalisons une vaste campagne de simulations perturbées (pour simuler le comportement incertain), appliquées à des trajectoires qui ont été optimisées pour un quadricoptère planaire. Les résultats montrent que la minimisation des sensibilités de l'état et de l'entrée s'avère efficace pour réduire les erreurs dans les espaces associés. Les trajectoires obtenues forment un bon compromis entre celles qui minimisent un seul objectif.

Néanmoins, notre approche de sensibilité repose sur une hypothèse qui n'est que peu réaliste, à savoir que l'état est parfaitement connu lors de la tâche de contrôle. En effet, l'état étant une estimation calculée à partir de toutes les mesures capteurs, il est donc incertain par essence.

Motivés par ce dernier point, nous avons travaillé en collaboration avec le groupe de recherche 'Control of Networked Systems' (Université de Klagenfurt, Autriche), qui étudie la génération de trajectoires dont le suivi améliore l'estimation de l'état et/ou des paramètres du sys-

tème. Cela a permis de donner naissance à notre "control & observability-aware framework" : dans notre deuxième contribution (Chapitre 5), nous proposons une tentative de combiner les sensibilités de l'état et l'entrée en boucle fermée avec les métriques d'observabilité, pour des plans de mouvement robustes et informatifs. Intuitivement, les objectifs sont opposés. Grâce à la norme pondérée et augmentée de Chebyshev, nous avons pu inclure les deux objectifs dans un seul problème. Dans ce chapitre, la représentation des trajectoires a été remplacée par des courbes de Bézier par morceaux, car elle permet de réduire leur dimension. Notre approche multi-étapes a ensuite été évaluée pour un quadricoptère 3D : dans une étude de cas simplifiée, nous avons considéré les coefficients de portance et de traînée des hélices comme incertains. Dans la section des résultats, nous avons montré que la sensibilité et l'observabilité sont souvent en conflit, ce qui justifie notre approche. Sous les conditions décrites, les trajectoires minimisant les deux objectifs se sont révélées comme un bon compromis entre les références dites *control-aware* et celles dites *observability-aware*.

Dans notre dernière contribution (Chapitre 6), nous définissons une nouvelle norme de la sensibilité pour le coût de l'optimisation : en effet, la norme de Frobenius que nous avons exploitée dans les deux contributions précédentes, ne tire pas profit d'une éventuelle connaissance des plages d'incertitude paramétriques. Avec cette information, il est possible de construire l'ellipsoïde de la plus grande déviation dans l'espace des paramètres. Nous montrons ensuite comment exploiter les sensibilités de l'état, de l'entrée et de la sortie, pour obtenir les ellipsoïdes correspondants dans leurs espaces respectifs. À partir de cette théorie, nous exploitons maintenant les 'tubes de déviation du pire cas'. Ceux-ci sont de la plus haute importance car ils nous permettent maintenant de considérer de nouvelles contraintes, améliorant la sécurité de la tâche. De plus, cela nous donne la possibilité de supprimer la sensibilité de l'entrée du coût, et laisse maintenant toute la place aux sensibilités de l'état et de la sortie pour l'optimisation. Dans cette contribution, nous avons étendu notre modèle de quadricoptère à un modèle dont le centre de masse est décalé, ajoutant ainsi de nouvelles incertitudes à notre framework fondé sur les tubes. La section des résultats permet de vérifier le plein fonctionnement des tubes d'incertitudes. Dans le contexte de la planification des mouvements robotiques, ces derniers peuvent être exploités pour générer des références que nous assurons être dynamiquement rélisables et sans collision (elles évitent les obstacles même les pires paramètres). L'approche est validée, puisqu'elle permet d'améliorer la précision de la sortie du quadricoptère.

La piste de recherche que nous avons suivie dans cette thèse offre un aperçu intéressant du potentiel de notre approche de planification de trajectoires robustes aux incertitudes paramétriques. Bien que le concept de sensibilité existe depuis longtemps, son application à un couple

système/contrôleur en boucle fermée est récente. Nous avons demontré que le domaine de la planification de trajectoire robuste peut tirer profit de nos métriques, et des nouvelles façons de les exploiter pour résoudre des problèmes d'optimisation pertinents. Nous sommes convaincus que nous n'avons que survolé la surface de ce sujet et que des recherches ultérieures sont nécessaires pour en explorer pleinement le potentiel.

ENGLISH SUMMARY

Over the past centuries, society has undergone significant changes, thanks to major breakthroughs in science and industry. New mechanical and electronic technologies have made it possible to replace human labour with automated machines. The standardisation of robotics is intended to take this development to the next level, replacing human intervention in the physical world, even in areas of our lives that were once considered trivial, such as driving, cleaning and landscaping. Robots are already being used to perform menial, physically demanding, repetitive and monotonous tasks. Besides, they can also exceed human capabilities by operating in dangerous or inaccessible environments. From an economic standpoint, the use of robotic labour is attractive, as machines can operate continuously.

As robots become more prevalent in our industries and daily lives, they hold the potential to fundamentally transform our relationship to work. However, the full potential of robotics in society is yet to be realised and requires further research, engineering and innovation to better understand and expand its applications.

One of the major challenges of automated systems is the need to operate in real, and therefore uncertain conditions. Since robot decisions are based on models describing themselves and their environment, the mathematical perfection of these latter is inevitably limited when it comes to describing reality. A classical problem is that of parametric uncertainty, which can arise when performing a motion task, *e.g*. when a mobile robot has to reach a specific location to perform a task such as grasping an object. In the case where the model employed is valid, but fed with erroneous parameters, the task performance may differ from the expected *nominal* behaviour.

To overcome this fundamental problem inherent in robotics, the primary solution has been to design controllers that specifically address the issue, while ensuring a stable behaviour, even in the case of parameter deviations. A first, classical approach, is to develop robust controllers, *i.e*. static control laws that guarantee a certain degree of insensitivity, for a limited range of parametric uncertainty. However, these laws must be specifically adapted to the system at hand. Passivity-based methods, which can also be included in robust control, exploit structural energy properties of the system, invariant to changes in the parameters of the system.

Another approach is to perform online parameter estimation, while performing the control

action. This method involves transmitting an update of the measurement to the controller, which incorporates additional dynamics to account for the changes. However, online parameter estimation is not an easy task, as it requires sufficient excitation of the robot dynamics, which may conflict with the main navigation task. In addition, the dynamics of the estimation can produce undesirable transients, which increases the risk of compromising the stability of the coupled system/estimation.

In the aforementioned cases, robustness is often achieved at the expense of the accuracy or performance of the task. Indeed, a trade-off is unavoidable: either one opts for a perfect controller, which ensures a high accuracy of the tracking task, but which relies on a very good knowledge of the model parameters (thus making it very vulnerable to disturbances), or one prefers a controller that guarantees a certain degree of robustness against disturbances, but which often only allows a so-called *practical stabilisation*.

Another point of view reverses the main design philosophy in response to parametric uncertainties: rather than focusing on the design sophisticated controls, the system/controller pair is left as it is, with its advantages and disadvantages, and the focus is on optimising trajectories to improve the performance of systems with uncertain parameters. In this perspective, recent work has focused on the generation of so-called *feed-forward* trajectories, adequately shaped to minimise the 'open-loop' state sensitivity of a system.

More recently, these works has been extended to the planning of trajectories with minimal state sensitivity in 'closed-loop'. This new research, known as *control-aware*, directly integrates the system/controller coupling into the optimisation. In this way, an adequate choice of the shape of the motion setpoint ensures the robustness of the navigation task, even under parametric uncertainties. This approach is likely to be beneficial for many future applications, and we believe it deserves further development.

In this respect, the main objective of this thesis is to explore the possibilities and limitations offered by robust, *control-aware* trajectory planning with respect to parametric uncertainties. The research problem is stated in more explicit terms as the design (and implementation) of novel trajectory planning frameworks for quadrotors. The validity of the presented software methods is always empirically assessed through extensive statistical simulation campaigns.

Thesis structure

This thesis starts with the introduction of the scientific background (Chapter 1) which justifies the studies carried out in the different contributions. Chapter 2 aims to review the scientific

literature on the generation of robust trajectories for systems with uncertain parameters. First, we examine the different mechanical configurations of aerial robots, and then we study the robustness of control strategies for quadrotors. Next, we focus on state estimation, and parameter estimation as a sub-problem of estimation. After that, we study the generation of paths and then robotic trajectories. Finally, we review robust reference planning techniques for systems with uncertain parameters. After reviewing the state of the art, Chapter 3 focuses on establishing the mathematical fundamentals exploited throughout the manuscript. In particular, a dynamic model of a 3D quadrotor is detailed. With these tools at our disposal, we are ready to present the contributions of this thesis.

In our first contribution (Chapter 4), we propose an extension of the minimum sensitivity trajectory planning framework, introducing the closed-loop input sensitivity, which, when reduced, leads to an increased prediction of the system input. We then formulate an optimisation problem that combines the two sensitivities into a single objective. We also replace the trajectory representation with Bézier curves, which offer better numerical stability than plain polynomials. To evaluate the appropriateness of our strategy, we perform an extensive campaign of perturbed simulations (to simulate the uncertain behaviour), applied to trajectories that have been optimised for a planar quadrotor. The results show that minimising the state and input sensitivities is effective in reducing the errors in the associated spaces. The resulting trajectories are a good trade-off between those that minimise a single objective.

Nevertheless, our sensitivity method relies on the unrealistic assumption that the state is perfectly known during the control task. Indeed, as the state is an estimate that gets computed from all sensor measurements, it is inherently uncertain.

Motivated by this last point, we have worked in collaboration with the research group 'Control of Networked Systems' (University of Klagenfurt, Austria), which studies the generation of trajectories whose tracking improves the estimation of the system state and/or parameters. This led to the "control & observability-aware framework": in our second contribution (Chapter 5), we propose an attempt to combine the state and input closed-loop sensitivities with observability metrics, for robust and informative motion plans. Intuitively, the objectives are opposite. Thanks to the augmented weighted Chebyshev method, we were able to include both objectives in a single cost. In this chapter, the representation of trajectories was replaced by piece-wise Bézier curves, as it allows to reduce their dimension. Our multi-step approach was then evaluated for a 3D quadrotor: in a simplified case study, we considered the thrust and drag coefficients of the propellers as uncertain. In the results section, we have shown that sensitivity and observability are often in conflict, which justifies our approach. Under the conditions described, the

trajectories minimising both objectives proved to be a good compromise between the so-called *control-aware* and *observability-aware* references.

In our last contribution (Chapter 6), we define a new sensitivity norm for the cost of optimisation: indeed, the Frobenius matrix norm that we exploited in the two previous contributions, does not take advantage of a possible knowledge of the parametric uncertainty ranges. With this information, it is possible to construct the ellipsoid of the largest deviation in the parameter space. We then show how to exploit the sensitivities of the state, input and output, to obtain the corresponding ellipsoids in their respective spaces. From this theory, we now exploit the 'worst case deviation tubes'. These are of utmost importance as they now allow us to consider new constraints, improving the safety of the task. In addition, it gives us the possibility to remove the input sensitivity of the cost, and now leaves all room for the state and output sensitivities to be optimised. In this contribution, we have extended our quadrotor model to a new one with a shifted centre of mass, thus adding new uncertainties to our tube-based framework. The results section verifies the full functionality of the uncertainty tubes. In the context of robotic motion planning, they can be exploited to generate reference motions that we ensure dynamically feasible and collision-free (they avoid obstacles even with the worst parameters). The approach is validated, since it improves the accuracy of the quadrotor output.

The research track we have pursued in this thesis provides an interesting insight into the potential of our approach to robust trajectory planning under parametric uncertainties. Although the concept of sensitivity has been around for a long time, its application to a closed-loop system/controller pair is recent. We have shown that the field of robust motion planning can benefit from our metrics, and from new ways of exploiting them to solve relevant optimisation problems. We believe that we have only scratched the surface of this topic and that further research is needed to fully explore its potential.

LIST OF FIGURES

LIST OF ACRONYMS

ACO Ant Colony Optimisation.

AI Artificial Intelligence.

APF Artificial Potential Fields.

APhI Aerial Physical Interaction.

AR Aerial Robot.

AU Actuation Unit.

AV Aerial Vehicle.

BLDC brush-less Direct Current.

CCW Counter-Clockwise.

CFD Computational Fluid Dynamics.

CHOMP Co-variant Hamiltonian Optimisation for Motion Planning.

COBYLA constrained optimisation by linear approximations.

CoG Centre of Gravity.

CoM Centre of Mass.

COP Control & Observability-aware Planning.

CW Clockwise.

DFL Dynamic Feedback Linearisation.

DoF Degree of Freedom.

EA Evolutionary Algorithms.

EELOG Expanded Empirical Local Observability Gramian.

EKF Extended Kalman Filter.

ESC Electronic Speed Controller.

ESDF Euclidean Signed Distance Field.

FCU Flight Control Unit.

FL Fuzzy Logic.

FPV First Person View.

GA Genetic Algorithms.

GPS Global Positioning System.

H_∞ H-infinity.

IEKF Iterative Error-state Kalman Filter.

IMU Inertial Measurement Unit.

INDI Incremental Non-linear Dynamic Inversion.

INS Inertial Navigation System.

KF Kalman Filter.

LiPo Lithium Polymer.

LQR Linear Quadratic Regulation.

LSP Linear Scalarisation Problem.

MoCap Motion Capture.

MOOP Multi-Objective Optimisation Problem.

MPC Model Predictive Control.

NLOPT non-linear optimisation.

NN Neural Network.

ODE Ordinary Differential Equation.

OKF Optimal Kalman Filter.

PID Proportional Integral Derivative.

POMPD Partially Observable Markov Decision Process.

PRM Probabilistic Road-map Method.

RRT Rapidly-exploring Random Trees.

s.t. subject to.

S/I-S State and Input Sensitivities.

SMC Sliding Mode Controller.

SNR Signal-to-Noise Ratio.

SOOP Single-Objective Optimisation Problem.

std standard deviation.

TO Trajectory Optimisation.

TSP Travelling Salesman Problem.

UAV Unmanned Aerial Vehicle.

UKF Unscented Kalman Filter.

VTOL Vertical Take-Off and Landing.

w.r.t. with respect to.

Part I

Preliminaries

CHAPTER 1

INTRODUCTION

In the past centuries, society has undergone significant changes, due to major breakthroughs in science and industry. New technologies in mechanics and electronics have allowed to increasingly replace human labour with automated machines. In recent decades, this trend has accelerated even further, as the democratisation of computers and the internet has led to many tasks and decisions previously performed by humans being delegated to automated systems. The standardisation of robotics is set to take this evolution to the next level, replacing human intervention in the physical world, even in areas of our lives that were once considered mundane, such as driving, cleaning and landscaping. Robots are already being utilised, to perform tasks that are menial, physically demanding and repetitive. Furthermore, they may also surpass human capabilities by operating in dangerous or inaccessible environments, such as the ocean depths or space. From an economic standpoint, using robotic labour is attractive, as machines can operate continuously. However, the interest in robotics goes beyond just economic considerations, as seen by the growing use of robotics in the healthcare sector, for whom reliability and precision are beneficial.

As robots become more prevalent in our industries and daily lives, they hold the potential to fundamentally change our relationship with work. However, the full potential of robotics in society is yet to be reached and requires further research, engineering, and innovation to fully understand and expand its applications.

1.1 Aerial robotics

The term *aerial robotics*, see *e.g.*, [Feron et al., 2008], has been popularised as a way to describe a class of automated machines with flying capabilities. This field of technology mainly aims at developing a broad range of systems, Unmanned Aerial Vehicles (UAVs), more commonly known as drones, that have the ability to manoeuvre in 3-dimensional space, without human pilots on board. UAVs are conceived for different levels of autonomy: some may operate fully under remote control by a human, when others have specific sensors and functions for

piloting aid. The high levels of autonomy are reached with autopilot, computer control, or even completely unsupervised flights with no human intervention. It is now a very common topic in the industry as well as in the research community.

The history of aerial robotics can be traced back to the early 20th century, and is closely tied to the history of flight itself. Likely, the high rate of deaths during early manned flight tests convinced engineers that there was a need to explore the potential of unmanned aircrafts, even before specific applications were identified. For this reason in particular, pioneers have conducted the early unmanned flights at the beginning of the 20th century. Primarily, these latter were performed to test wing designs, control surfaces, and in general, assess the capabilities of their aircrafts. The early unmanned flights were important milestones in the development of UAVs, as they allowed engineers to control the systems, and improve their designs before putting human pilots on board. As a result, the first powered unmanned flight was achieved, and has since sparked significant interest in the field of aerial robotics.

After the first trials, the following developments of UAVs have enabled to use them for military purposes, see [Keane et al., 2013]. During World War I, they have not been widely exploited, as the technology was not yet advanced enough to support their deployment on the battlefield. However, there were some early attempts to take advantage of these systems for reconnaissance, artillery spotting. The efforts were mostly unsuccessful, due to the limitations of the technology at time. Two decades after, during World War II, UAVs began to be more and more exploited, for reconnaissance missions, target acquisition, and as decoys for air defence systems. Some drones were even used in combat: for instance, pulsejet-powered cruise missiles were launched from a ramp, and guided by autopilot. In general at these times, the technology was still in its infancy. Limitations such as short range and lack of autonomy made them less effective than manned aircrafts. However, their successful use in reconnaissance missions laid the foundation for further development and advancements in the field. In the post-World War II period, UAVs continued to be developed and utilised for military purposes, with an emphasis on reconnaissance and surveillance.

During the later part of the 20th century, the technological improvements in many fields, *e.g.* mechanics, electronics and automation, have significantly improved the capabilities and performance of Aerial Robots (ARs). Among the main advancements that have been beneficial to aerial robotics, we can cite:

— the use of lightweight materials such as carbon fibre has allowed to reduce the total weight of the designs;

— advanced aerodynamic designs, especially about the propellers, have led to the develop-

ment of small, more manoeuvrable systems;

— more performing, efficient motors, such as brush-less Direct Current (BLDC) motors, which have been made viable by advances in permanent magnet materials;
— the manufacturing of batteries with higher energy density, such as Lithium Polymer (LiPo) technology, that have been made more powerful, reliable, versatile and long-lasting, allowed to increase flight times, thus improving the duration of drone missions;
— the integration of Global Positioning Systems (GPSs), Inertial Navigation Systems (INSs) and computer vision have greatly enhanced the accuracy of location, hence of control, allowing for precise and efficient operations;
— the development of advanced, small integrated sensors such as cameras, lidars, radars, and other imaging devices have enabled to gather detailed and accurate data;
— the progress made in telecommunications has been a key factor in the expansion of UAVs capabilities: the ability to transmit and receive data in real-time over extended distances provides the ability to execute more complex tasks than before.

The aspects listed above have made it possible to improve the overall performance while reducing the costs of systems, thus arising the interest in potential civilian applications, and democratising the use of ARs beyond the traditional military roles. Subsequently in the past decades, many Aerial Vehicles (AVs) have been developed, including multi-rotor UAVs. Owing to the many implementations, it is now common to find a large range of commercial products, for entertainment purposes and many professional applications. For the sake of understanding why the society can benefit from the progress in the aerial robotic field, we now cite typical usages of multi-rotor UAVs.

First, we can mention surveillance and reconnaissance, one of the major application of multi-rotors. Equipped with cameras and other sensors, they capture images and valuable data to provide the desired information. Therefore, they are useful for monitoring large areas, and track movement of people, vehicles. Their small size and manoeuvrability allow them to navigate easily in areas that are not accessible by manned aircrafts. Hence, the integration of ARs in surveillance and similar operations has expanded the capabilities, in a very cost-effective way.

A similar application is external asset inspection and maintenance, which is performed on many infrastructures, including power plants, bridges, power lines. UAVs equipped with cameras remotely collect image-based data, and can also take advantage of specific advanced sensors to detect and identify various types of defects, *e.g.* corrosion and structural issues, that may be harmful to the integrity of facilities. Thanks to multi-rotors, the monitoring is safer, since inspection teams can avoid potential danger by staying on the ground, and is also greatly sped

up, thus ultimately much cheaper.

AVs are also employed to help in the construction industry, by conducting extensive imaging of the sites. More precisely, specialised onboard sensors can be used to generate detailed 3-dimensional representations of the current structures, which facilitates the planning and execution of projects, with improved precision and efficiency.

Search and rescue operations greatly benefit from the use of drones, since they can be employed to quickly search for missing individuals in challenging areas, such as mountainous and forested regions, that would be difficult to access by ground-based search teams. In the case of disasters, they provide real-time situational awareness and assess the extent of damages, which allow quicker, informed decisions from the emergency teams. Potentially, with drones, more lives are saved during this kind of operations.

Many other applications exist, among which we can list:

— environmental monitoring and meteorological research, where AVs gather information, monitor wildlife, help to study the environment, which for instance, led scientists to a greater understanding of the Earth's atmosphere and surface;
— agriculture, where drones equipped with cameras, tanks and sprayers can be quickly deployed over fields to carry out the appropriate watering, give visual feedback to the farmers, with additional data on crop health, pest level assessment: with all these tasks performed at once, the savings in time and money are significant;
— film and photography, where multi-rotors have become a popular tool, since they open up to stunning possibilities in terms of shooting: now, one can use them to produce high-quality immersive videos, imitating bird's-eye perspective, as popular these days with First Person View (FPV) recordings;
— delivery of small packages, especially where traditional delivery methods are hindered by factors such as terrain, traffic, or security concerns: here, they have proven to be a valuable asset for delivering essentials, *e.g.* medical supplies or tools, respectively in disaster areas or for construction/maintenance operations.

One can note that in all of the applications discussed thus far (apart from delivery), the UAVs are mostly exploited as remote sensors. The main interest in these is to gather data, without actively interacting with the environment. Besides free-flying, a recent trend is also in Aerial Physical Interaction (APhI), which opens the door to a wider range of new and exciting applications, expanding the already existing capabilities of multi-rotor UAVs. We mention some topics of current interest in the literature, such as:

— tool/package delivery for maintenance operations, see [Suarez et al., 2022]: please note

that as opposed to the previously mentioned delivery application, here, high proximity with workers is considered;

— contact-based inspection, see *e.g.* [Alexis et al., 2016], as an extension of the previously existing contact-free inspection tasks;

— assembly and construction of structures, *e.g.* in [Jimenez-Cano et al., 2013], and extended to co-working of ARs in [Muscio et al., 2017];

— human-robot interaction for co-working/manufacturing, see [Afifi et al., 2022; de-Dios et al., 2020].

Of course, the subject is very vast, and we thus propose the reader to refer to [Ollero and Siciliano, 2019; Ollero, Tognon, et al., 2021; Tognon and Franchi, 2020] as well as the references therein, for more in-depth explanations on aerial manipulation, and APhI in general.

In any case, this emerging trend is not the subject of this thesis, as we limit ourselves to contact-free flights. We mention this new concept because it allows us to illustrate the problems that we address in this thesis with concrete examples, *e.g.* the ones in Fig. 1.1, discussed later.

1.2 Scientific context and main objectives

Adopting a general viewpoint, one of the major challenges for automated systems is the need to operate in real (thus uncertain) conditions. As robot decisions are based on some models, of themselves, of their environment, the mathematical perfection of these models is unavoidably limiting for describing reality. To clarify, there is an infinitesimal chance that a mathematical model takes into account all phenomena/effects that affect the behaviour of a robot, since it is almost certain that some of them have not yet been identified. A classical example of what we stated here-above is that of parametric uncertainty, in the context of execution of a motion task, *e.g.* a mobile robot needs to reach a specific location, to perform a task such as grasping an object. In the case of a valid model, but with wrong parameters feeding it, the realisation of the task might differ from the *nominal* behaviour, which occurs when the accurate parameter values are implanted inside the *closed-loop* system.

More specifically, systems such as multi-rotor UAVs are subject to uncertainties, in the models and their parameters. A first example of that is the thrust exerted by the propellers, whose model is derived from fluid dynamics, a disciplinary field for which the knowledge is still limited, see, *e.g.* [Walters et al., 2002]. The intricacies of the turbulences in a fluid and many other phenomena, often highly non-linear, are very hard to model properly. One can also easily understand that even with an exhaustive description of the phenomena, the complexity of

the models would explode, which is not desirable at all.

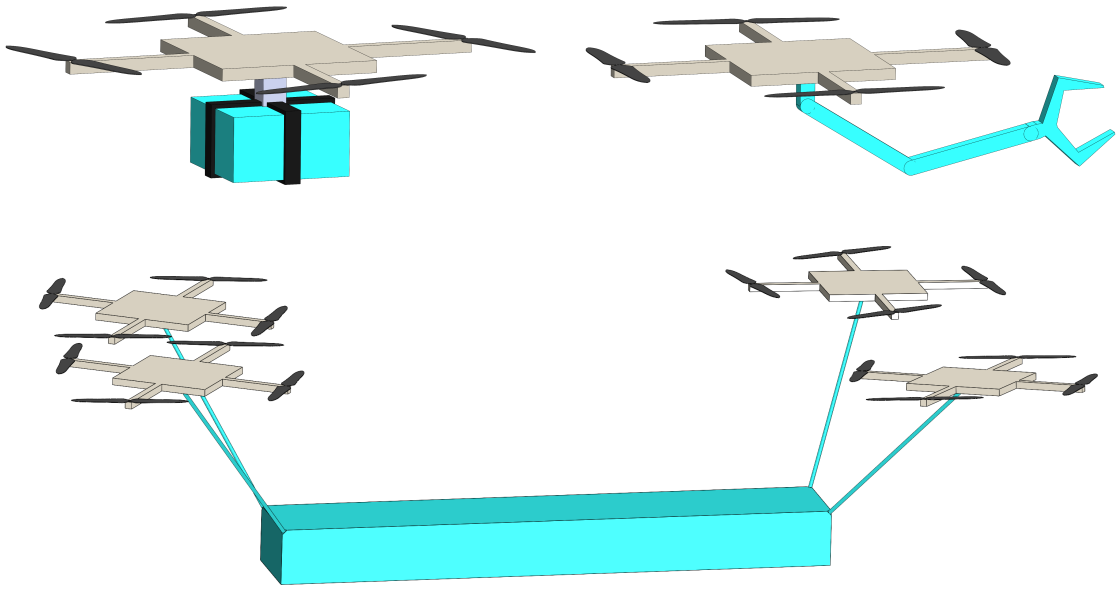

Figure 1.1 – Schemes of recent/future possible applications with Vertical Take-Off and Landing (VTOL) UAVs. At the top, a quadrotor carrying a package for delivery (left), and a quadrotor with an articulated gripper (right). At the bottom, four drones carry a large object attached by cables. For these three cases, we highlight (in cyan) the additional elements that may lead to parametric uncertainties.

Other possible examples of uncertainties in the model parameters of an AR could be, *e.g.* length properties, the total mass, inertia, or position of Centre of Mass (CoM). This can happen especially when carrying a package, tool, or any other object of uncertain dimensions (thus inertia). Even with a good knowledge of the parameters of this object, the quality of the grip itself may also be at stake, and have a significant impact on the measurement/estimation accuracy of the parameters involved in the system dynamics. Therefore, the challenges related to parametric uncertainties have to be studied for multiple applications, as depicted in Fig. 1.1. At the top left, one can observe a quadrotor carrying a package for delivery. In this context, the uncertainties may occur on: the mass of the object to be delivered, its inertia (especially if the package has moving parts inside of it), the position of its CoM, the total air drag coefficients of the structure, *etc*. At the top right, one can observe an hexarotor with an articulated gripper. For this system, the uncertainties occur on the same parameters than with the previous example, but here the parameters of the complete structure can change if the gripper articulation angles changes. Again, *e.g.* if a tool is gripped, the grasping quality might cause uncertainties. The last example (at the bottom of Fig. 1.1) depicts four co-working quadrotors that move a large object using cables. For this last example, the idea is the same than for the previous one: uncertainties

can arise from the object to be moved itself, or from the relative positioning of the quadrotors with respect to (w.r.t.) the other elements. Note that for the preceding examples, the parameters are subject to variations w.r.t. time: in the scope of this thesis, even if we consider parameters that may deviate from their *nominal* values, we always make the (strong) assumption that the parameters are constant w.r.t. time (which, of course, may be wrong for certain parameters, *e.g.* the thrust coefficient of the propeller might change if there is a high shift in temperature).

To tackle this fundamental problem of robotics, the primary solution has been to design controllers that specifically address the issue (for the system at hand), and ensure a stable behaviour, even in the presence of parametric uncertainties.

A first classical approach is that of designing robust controllers, which are typically static control laws that ensure a certain degree of insensitivity to a bounded range of parametric uncertainty, *e.g.* H-infinity loop shaping, or sliding mode controller. However, they need to be specifically tailored to the system at hand, by exploiting the uncertainty upper bound. Passivity-based methods, which can also be included in robust control, exploit structural energy properties of the system, invariant to changes in the parameters of the system.

Another way is to estimate the parameters online, while performing the control action. An update of the measurement is fed to a controller, which includes additional dynamics for considering the changes. But estimating parameters online is not easy, since it requires that the system dynamics during the trajectory tracking are exciting enough, which might be in conflict to the main task. Besides, the dynamics of the estimation might produce unwanted transitional regimes that might compromise the stability of the coupled system/estimation, and therefore hinder performance.

In the previously mentioned cases, robustness is often attained at the expense of the accuracy/performance of the task. Indeed, a trade-off is always present: either we take a perfect controller, that is able to perform the tracking task with utmost accuracy, but which relies on good knowledge in the model parameters (and is subsequently very sensitive to perturbations), or we rather make the choice of one that guarantees a degree of robustness even in the presence of uncertainties, but which will not be able to perform the tracking very well, and often only ensures a *practical stabilisation*.

Another point of view, which reverses the main design philosophy when seeking to counter parametric uncertainties: instead of designing sophisticated controllers, the system/controller pair is left as it is, with its advantages and drawbacks, and the goal is rather to focus on trajectory optimisation, to improve the performance of a control task for uncertain systems. For instance, recent works have put efforts in the generation of feed-forward/planned trajectories meant to

minimise the *state sensitivity* of a system, but with the main limitation of remaining in *open-loop*.

More recently, these works have been extended, to plan trajectories that have minimal *closed-loop* state sensitivity. This new *control-aware* research takes into account the coupling of the system/controller directly in the optimisation. In that way, the robustness is ensured by choosing an adequate shape of the reference motion. We believe that this approach deserves to be further developed, as it could prove useful for many future applications.

In that respect, the main objective of this thesis, see Fig. 1.2, is to explore the possibilities and limitations offered by robust, *control-aware* trajectory planning with respect to (w.r.t.) parametric uncertainties. The research problem is stated in more explicit terms as the conception (and implementation) of novel trajectory planning frameworks, for ARs, and more precisely quadrotor UAVs. The soundness of the presented software methods is always empirically assessed through extensive statistical campaigns of simulations.

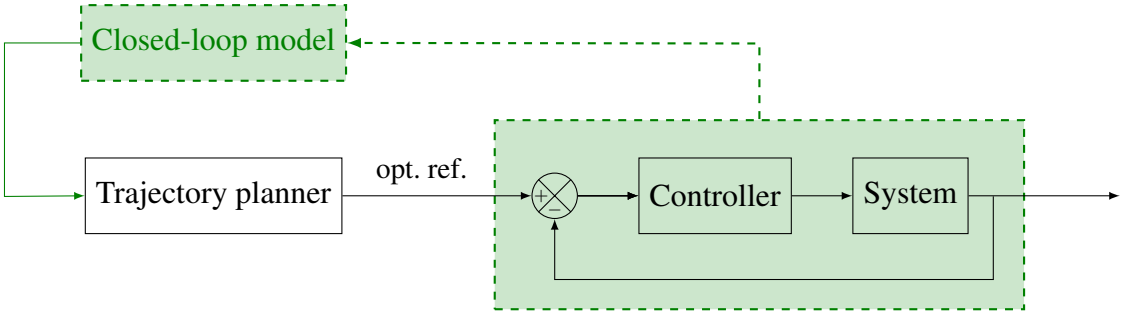

Figure 1.2 – Scheme of the main design philosophy of the robust, control-aware trajectory planning framework that we develop in this thesis: we model the closed-loop system/controller pair, and use this information to generate trajectories to be fed to the real closed-loop system. Here, the system has parameters (unknown, because of measure uncertainty), and the *nominal* parameters, which are implanted inside the closed-loop model (measured value) might differ from reality, thus justifying our approach.

1.3 Thesis outline

This manuscript is organised in three main parts. To begin with, Part I (Chapters 1 and 3) introduces the subject, and the concepts which are used throughout the thesis. The overview of the current state of the literature of Chapter 2 allows us to enter in depth into the main topics of this manuscript. First, a short study of existing multi-rotor ARs designs is proposed. Then, the rest of the literature review focuses mostly on quadrotor UAVs. Hence, we present different types

of control strategies for these latter, with an emphasis on the robustness of each w.r.t. model parametric uncertainties. Next, we explore the state/parameter estimation trend, that seeks to properly estimate various quantities of a system, so as to improve the performance of the full feedback control system. The chapter continues with a focus on trajectory planning for AVs, which is subsequently completed by our main topic of interest, *i.e*. robust trajectory planning under model parameter uncertainties. By addressing this last point, we introduce for the first time the concept of *closed-loop state sensitivity*, which is further used in the contributions. We finish the preliminaries with the presentation of the main mathematical conventions and tools for modelling the multi-rotor AVs and their controllers, in the subsequent chapters. In particular, we detail the procedure to obtain the dynamics of a standard quadrotor therein.

Equipped with the tools of the modelling chapter, we next delve into the core of this thesis, *i.e*. Part II (Chapters 4 and 6), and present an in-depth presentation of the contributions, *i.e*. the evolution of our robust trajectory generation framework. After a generic formulation of the Trajectory Optimisation (TO) problem, Chapter 4 defines the newly introduced *closed-loop input sensitivity*, that complements the already existing closed-loop state sensitivity. We then exploit these latter to generate the objective cost for the optimisation problem, which now considers actuator limitations, as non-linear constraints. Lastly, we validate the framework with an extensive statistical set of simulations, applied to a planar quadrotor combined with a Dynamic Feedback Linearisation (DFL) controller.

Chapter 5 seeks to combine *control-aware* trajectories, *i.e*. the ones that are obtained by minimising the closed-loop state and input sensitivities, and *observability-aware* trajectories, *i.e*. the ones that facilitate the most the state/parameter estimation. We formulate an optimisation cost that incorporates the related metrics into a single index to be reduced. Then, we analyse the soundness of the proposed method for a space quadrotor with a geometric tracking controller. In particular, we discuss the cost behaviour regarding the two (seemingly opposite) objectives and their combination. To finish, we validate the procedure by showing positive results in terms of target reach precision, and parameter estimation performance.

In our last contribution, Chapter 6, we formulate an upgraded optimisation problem for robust trajectory planning, thanks to a new norm definition for the closed-loop state/output sensitivities. With this new methodology, and the knowledge of maximum parameter deviations, one can now construct the worst case tubes in the state/input/output spaces. Thanks to this new approach, we removed the closed-loop input sensitivity from the cost, and thus, we rather use it to formulate new actuator constraints, with the uncertainty tubes. As for the two preceding contributions, we validate the procedure with an extensive statistical campaign of simulations.

For this last contribution, we consider an upgraded model of the space quadrotor with a possible shift in the Centre of Gravity (CoG), compared to Chapter 5, and keep the same geometric tracking controller.

Finally, in Part III, we propose an overall conclusion of the contributions. It summarises the previously mentioned achievements and their limitations. Furthermore, we take a step back, and give a general conclusive word on the applicability of our frameworks, including an overview of the remaining challenges to be addressed, and perspectives opened by this thesis.

1.4 Thesis contributions

The work conducted throughout this thesis led to publications in international peer-reviewed conferences.

Pascal Brault, Quentin Delamare, et al. [2021], « Robust Trajectory Planning with Parametric Uncertainties », *in*: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11095–11101

Christoph Böhm, Pascal Brault, et al. [2022], « COP: Control & Observability-aware Planning », *in*: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3364–3370

Pascal Brault and Paolo Robuffo Giordano [2023], « Tube-Based Trajectory Optimisation for Robots with Parametric Uncertainty », *in*: In preparation for the IEEE Robotics and Automation Letters (RA-L)

CHAPTER 2

ROBUST TRAJECTORY PLANNING FOR UNMANNED AERIAL VEHICLES: A LITERATURE REVIEW

2.1 Introduction

This chapter aims to give an overview of the state of the art in relation to robust trajectory planning for UAVs. To begin with, we present several designs of multi-rotor ARs, with their capabilities and limitations. After reviewing several possible control strategies for such systems, we highlight some common approaches for state and parameters estimation. We finally focus on the main subject of this thesis, by first investigating customary methodologies of trajectory generation for UAVs, and then focusing on advanced techniques for robust trajectory planning.

2.2 Multi-rotor aerial robot designs

The versatility of multi-rotor ARs is achieved through the multitude of mechanical configurations available. Each design group possesses unique properties and capabilities, and therefore the choice of configuration may depend on the desired application. This topic is largely covered in the input allocation-based taxonomy of [Hamandi et al., 2021]. Through the whole manuscript, particular attention is given to AVs capable of VTOL, see [Y. Zhou et al., 2020]. Thanks to their ability to hover and take-off/land in a variety of environments, these latter offer a wider range of applications compared to other AVs: on the one hand, they can navigate in indoor environments such as offices, warehouses and other facilities, and on the other hand they are able to operate outdoor in adverse areas like forests or cluttered urban environments. The main components required for UAV actuation are first outlined, followed by a non-exhaustive examination of various designs and their key features.

2.2.1 Actuation unit components

A typical Actuation Unit (AU) for an AR consists of a motor and a propeller.

The vast majority of UAV designs utilise BLDC motors, see *e.g.*, [Yedamale, 2003], due to their advantageous properties, such as high torque-to-weight ratio, improved efficiency, reduced noise, longer lifespan (since they have no brushes), *etc*. Proper selection of these electromechanical conversion machines is achieved through characterisation, see [Gabriel et al., 2011], and allows to design systems with high efficiency, well-suited for the desired tasks.

Downstream of the engine, the mechanical rotational energy is converted into aerodynamic thrust by the propeller, which features several blades arranged in a helical spiral shape. The blades generate thrust on a fluid medium, such as air, as they rotate, creating a pressure differential between their upward and downward surfaces through Bernoulli's principle. Recent research in the field of propeller design for quadrotors can be found in [Patel et al., 2017], which covers optimal design and stability features. The role of propeller aerodynamics in UAV models is explored in [Bristeau et al., 2009], taking into account blade flexibility and the effects of disturbances on the CoM during forward flight. The aerodynamic characteristics of an AR can be further evaluated through Computational Fluid Dynamics (CFD), as demonstrated in [Christodoulou et al., 2019], leading to the identification of optimal propeller shapes for quadrotor UAVs.

Based on these studies, one can make informed decisions when selecting components for an AU, depending on the application. Note that this selection should be made taking into account the overall design of the UAV. In the following subsection, we will therefore present various drone configurations, emphasising their characteristics, depending on the number of AUs and their arrangement (in space).

2.2.2 Collinear actuation unit axes

UAVs with fewer than four AUs are inherently unstable, and thus receive limited coverage in literature. Nevertheless, studying such configurations is still valuable, as their control theories can be applied to naturally stable structures, in the event of one, or multiple AUs failures. [Zhang et al., 2016] presents a unirotor's dynamical model along with a relaxed hover solution. By nature, the vehicle can substantially only remain in a single position while its attitude is non-zero, due to the propeller's drag force. Typically, unirotors are only controllable in three translational Degrees of Freedom (DoFs) and two attitude DoFs. They are classified as collinear and coplanar since they are composed of a single actuation unit. [Sun, Wang, et al., 2020] also

introduces an Incremental Non-linear Dynamic Inversion (INDI) control strategy for a quadrotor, with the loss of two opposing AUs, resulting in a similar flawed structure as a collinear birotor. This configuration can only reach a relaxed hover solution, hovering in position while spinning along the yaw axis. Similarly, collinear trirotors face hovering limitations due to unbalanced torques, caused by an odd number of propellers, making a steady position and zero yaw velocity incompatible, as seen in [Kataoka et al., 2011].

UAVs with four collinear AUs axes, commonly referred to as quadrotors, are the most prevalent design in the field. In fact, having at least four AUs is the minimum required condition to be able to achieve complete hovering stability, *i.e.* all forces and torques applied to the structure can balance, leading to motionlessness. With less than that, tilting an AU would be required to gain this ability. In that respect, the simple mechanics and decoupled force and torque spaces of collinear quadrotors have made them a popular choice for study. The literature on collinear quadrotors, including works such as [Bouabdallah, Noth, et al., 2004; Bouabdallah and Siegwart, 2005; Mahony et al., 2012; Pounds, Mahony, and Corke, 2010; Pounds, Mahony, Hynes, et al., 2002], provides an in-depth understanding of the modelling and control problems. In addition to the simple mechanics, as described in [Mistler et al., 2001], *differential flatness* is a noteworthy property of quadrotors. In systems theory, *flatness* is a property that extends the notion of controllability from linear systems to non-linear dynamical systems. By definition, a system is considered *flat* if it has a *flat output*, from which one can explicitly express all states and inputs, in terms of the flat output, and a finite number of its derivatives. Quadrotors with collinear and coplanar AUs, evenly spaced around the geometric centre of the structure feature this nice property, which enables the use of DFL control laws, for very effective trajectory tracking.

Figure 2.1 – Designs of UAVs with collinear and coplanar AUs axes: a quadrotor (left), an hexarotor (middle) and an octorotor (right). Source: MikroKopter.

The properties of collinear quadrotors are shared with platforms that include six or even eight collinear AUs, hence these are also widely documented in the literature. As for quadrotors, the vast majority of the designs is with coplanar propellers, regularly spaced around the

geometric centre of the structure, and with maximum possible symmetry, as can be observed in the designs of Fig. 2.1. These solutions are favoured for their mechanical simplicity, as well as the ease of creating control laws. The collinearity of the propellers allows for maximum energy efficiency, as the thrust from each AU is directed towards the desired motion. Compared to quadrotors, collinear hexarotors and octorotors have increased payload capacity, and can overcome rotor faulty: *e.g.*, [Baskaya et al., 2021] studies INDI on a hexarotor, thus capable of static hovering, in the presence of rotor failure; [Marks et al., 2012] presents a redistributed pseudo inverse method of control reallocation for fault tolerant control of an octorotor. The combination of these strategies and the redundancy of the AUs on these platforms enhances their resilience, which is crucial for safety, especially when operating in close proximity to humans. Other designs with an odd number of AUs, such as pentarotors and heptarotors, lack symmetry and are by nature much harder to control, resulting in their limited presence in the literature.

Beyond the interesting properties offered by the collinearity of the AUs, some weaknesses arise directly from the design. The major issue is that the total thrust can only be produced in one fixed direction w.r.t. the body frame, causing a strong coupling between position and orientation control. This results in an *under-actuated* design, meaning that it is not possible to fully control the AV state. For example, starting from a hovering position, the UAV must rotate to orient the thrust toward a desired lateral direction. Conversely, the platform cannot tilt without also moving. In fact, the only stable hovering orientation is with the AUs axis aligned with the gravity acceleration vector. Furthermore, the yaw control is limited because the propellers are coplanar. In configurations with an even number of rotors, one out of two propeller rotates Clockwise (CW) and the other Counter-Clockwise (CCW): thus, the total torque around the yaw axis arises from the difference in drag torques between the even propellers and odd propellers.

One can quickly deduce that this class of designs is inadequate for various challenges, including APhI. In this context, we recall the reader that the AV must be capable of physically engaging with its surroundings, by, *e.g.*, exerting forces to objects in any desired direction, and possibly while statically hovering.

2.2.3 Tilted actuation unit axes

To address the limitations of collinear designs, some studies have explored the capabilities of structures with tilted AUs. These latter designs allow for a combination of individual thrust vectors, each generated by a single propeller's spin, resulting in the ability to exert a non-fixed thrust orientation in the AR's body frame. This versatility provides partial or full decoupling of position and orientation control. As depicted in Fig. 2.2, the total thrust exerted by a tilted

hexarotor UAV is compared to that of a collinear design, in different propeller rotation rate configurations. On the top row, where propellers rotate at the same speeds, the total thrust vector (in red) is aligned to the vertical axis of the body frame in both designs, with slightly lower force applied to the tilted UAV: this highlights the reduced energy efficiency of the tilted designs, when oriented towards a desired direction of motion. On the bottom row with different propeller angular rates, the total thrust orientation changes in the tilted design, allowing for lateral movement while maintaining standard hovering orientation, while for the collinear design, it remains vertical: this emphasise the under-actuation of collinear designs, for which a change in the body's orientation is required to exert the total thrust in a different direction than when hovering.

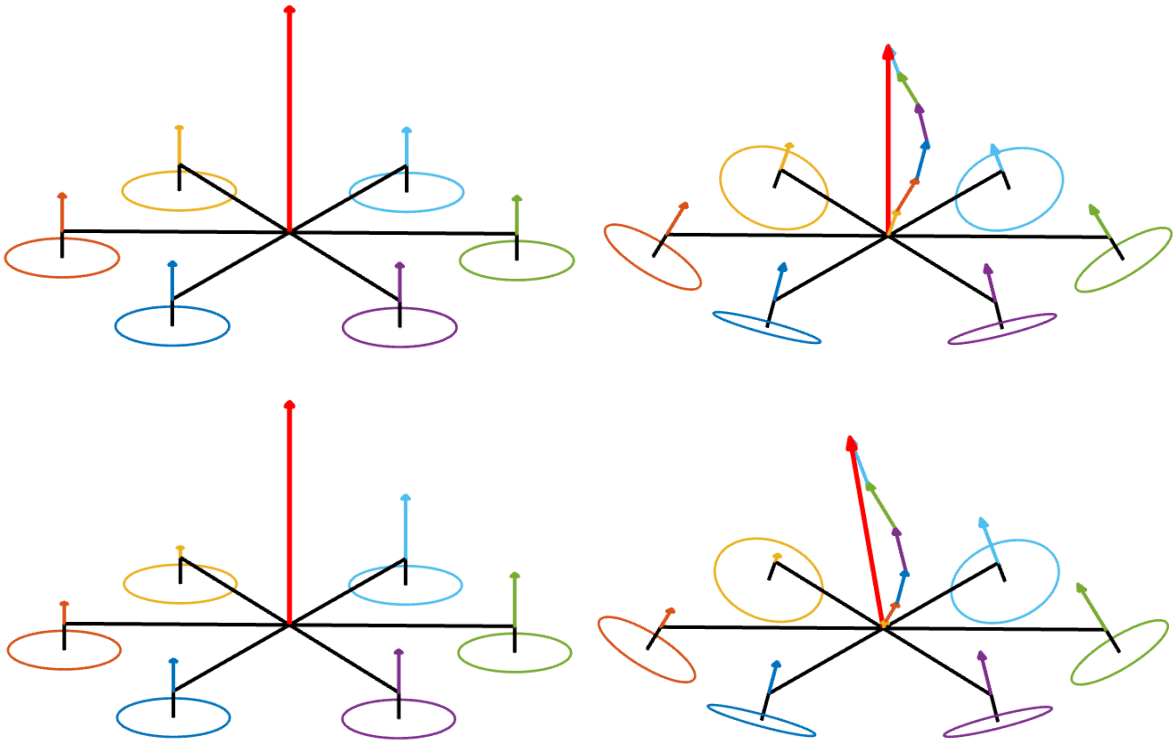

Figure 2.2 – Comparison of a collinear and coplanar AUs hexarotor (left) with a tilted AUs hexarotor (right). The total exerted thrust force vector (in red) is visible for two configurations of the propellers rotation speeds, respectively at the top and bottom rows.

Similarly to birotors and trirotors with collinear and coplanar propellers, designs for flying robots with two or three tilting AUs have been proposed, but controlling the complex dynamics is challenging, and static hovering is difficult to achieve, if achievable. Therefore, this subsection focuses primarily on systems with four or more AUs, that inherently possess the ability to

perform static hovering.

The article in [Zheng et al., 2020] describes a unique design for a quadrotor, where the AUs can be controlled in unison, through a parallel mechanism powered by two linear actuators. This prototype has the ability to hover, while tilting up to $30°$ due to its simple control allocator, which decouples the platform's position and orientation. Additionally, the prototype is capable of tracking lateral position commands without tilting, through thrust vectoring. However, it should be noted that this design has limited yaw control, because the AUs's tilting is always synchronised. Other studies have been conducted to enable quadrotors to fully track a 6-DoFs arbitrary trajectory. For example, in [Ryll, Bülthoff, et al., 2014], a novel over-actuated design of a quadrotor UAV was presented. This design enables the AUs to radially tilt independently, with four additional actuators, allowing for improved interaction with objects, compared to collinear and coplanar quadrotors. Despite this advantage, the design comes at the expense of increased complexity in dynamics and control, as well as increased total mass and inertia, which is inherently a limitation. The properties of robots with four tangentially tilting rotors have also been explored, *e.g.*, in [Badr et al., 2016; Devlin et al., 2018], which enables some decoupling between translational and rotational dynamics. In [Şenkul et al., 2014], a quadrotor was tested where all AUs could be inclined independently, both radially and tangentially. Equipped with a cascaded Proportional Integral Derivative (PID) controller, this AR was able to accurately track a 6-DoFs trajectory.

Without dynamic tilting, having six AUs is the minimum required condition for a UAV to allow full actuation. However, as states in subsection Subsection 2.2.2, it is not possible to obtain this feature with collinear and coplanar propellers. As well explained in [Hamandi et al., 2021], the tilting angles of each AU have to be carefully chosen in order to achieve full actuation, *i.e.* good force and torque decoupling, without sacrificing too much energy efficiency. Tilted hexarotor designs, as explored in [Michieletto et al., 2017; Rajappa et al., 2015; Ryll, Muscio, et al., 2017; Tognon, Chávez, et al., 2019], demonstrate excellent capabilities for APhI tasks. Indeed, to a certain extent, such systems are able to apply forces on objects for various purposes. Without a doubt, applying precise forces with a flying robot is very challenging, due to the nature of thrust generation: usually, we only control the angular velocity of the propeller, which, as explained is Subsection 2.2.1, converts the mechanical rotational energy into aerodynamic thrust force. The complexity of aerodynamics and associated disturbances limit the accuracy of force application, and thus the interaction.

While other tilted designs, such as pentarotors, heptarotors, and octorotors have been explored, the complexity of their models has resulted in increased difficulty in control. As a result,

the flagship designs remain dominant in the literature.

2.2.4 Summary on multi-rotor designs

In this section, we have shed light on several possible designs of multi-rotors. With each configuration possessing its own unique attributes and varying levels of intricacy, it becomes evident that the development of control algorithms is crucial, to allow AVs to operate autonomously and safely.

Although the concepts discussed in this thesis can be applied to any multi-rotor ARs, through the entire manuscript, the studies are primarily restricted to the classical quadrotors, with collinear and coplanar AUs, evenly spread around the geometric centre.

2.3 Robust control techniques for quadrotors

A key challenge in robotics is to create autonomous systems that perform well in complex environments. This requires implementing algorithms that allow for accurate and safe navigation. To achieve this, a feedback control system is often used, which involves monitoring the system's actual output using sensors, and then using a control law to compute inputs that bring the real output closer to a desired reference signal. The machinery, as depicted in Fig. 2.3, compares the actual output with the reference signal and calculates the necessary inputs to minimise the tracking error. When designed and configured correctly, a feedback control system can effectively track desired trajectories.

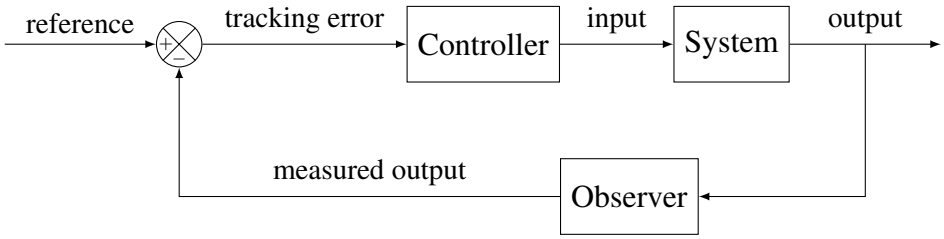

Figure 2.3 – Classical scheme of a feedback control system.

In Subsection 2.2.2, we discussed the main features of conventional UAV designs, particularly quadrotors with collinear and coplanar AU axes, evenly positioned around its geometric centre (as seen in the left of Fig. 2.1). This design is under-actuated, as the collinearity of the AU axes limits the total thrust force to a single direction w.r.t. the body frame. From a more mathematical point of view, the under-actuation of a quadrotor arises from the four AUs being

strictly inferior to the six DoFs of the UAV's pose in space. As a consequence, it is not possible to fully control the AR's state. This shortage induces challenges for stabilisation, trajectory tracking, and disturbances rejection.

The coupling between the position and orientation of a standard quadrotor UAV makes its dynamics highly non-linear. To address this, and improve the global performance of systems, researchers have designed controllers with varying levels of complexity, see *e.g.*, the reviews of [Han et al., 2018; Mo et al., 2019; Nguyen et al., 2020; Zulu et al., 2016] and the references therein. In this section, we review the main controllers that have been developed over the past decades, categorised in three classes of algorithms: linear control, non-linear control and adaptive/intelligent control.

As outlined in Section 1.2, the models of multi-rotor UAVs are based on the complex theories of aerodynamics and fluid mechanics. The possible uncertainties in the model parameters pose challenges for precise and reliable control. In the context of this thesis, for each class of control strategies, we compare the level of achievable robustness.

2.3.1 Linear flight controllers

Linear control systems are widely known for their simplicity and versatility. These control strategies have been effectively utilised in several applications, including the quadrotor UAVs case, where they have been shown to perform well in maintaining stable hovering. In this subsection, we will review some of these linear control techniques in more detail.

2.3.1.1 Proportional integral derivative controller

The PID controller, see [Johnson et al., 2005; Visioli, 2006], is probably one of the most popular algorithm. Due to its simplicity, it has been widely used to control mechatronic systems: it is easy to implement, the gain parameters can be adjusted without difficulty, and the tracking performance is consistent. The PID algorithm takes into account past, current, and future error states to reduce control error, and does not require a good mathematical model of the system being controlled.

However, when implementing a PID controller on a quadrotor, additional challenges arise, due to the non-linear dynamics, caused by the under-actuation of the AR. Despite this issue, many works have contributed to the development of this control strategy, see *e.g.* [Atheer et al., 2010; Bouabdallah, Murrieri, et al., 2005; Bouabdallah, Noth, et al., 2004; J. Li et al., 2011; Salih et al., 2010], and have demonstrated its ability to control the orientation angles, even in

the presence of small perturbations. The results lead one to believe that even if a quadrotor is inherently unstable, PIDs controllers are reliable enough for applications where the platform is sufficiently close to hovering (*i.e.*, the pitch and roll angles remain small enough).

In more recent developments [Hoffmann et al., 2007; H. Yang et al., 2017] in PID control, several issues have been addressed, *e.g.* when operating at higher speeds (*i.e.* when the flight regime cannot be approximated as near hovering), or in the presence of wind disturbances.

2.3.1.2 Linear quadratic regulation controller

As the field of optimal control is concerned with operating a dynamic system at minimum cost, linear quadratic problems have been described as the association of a system's linear dynamics and a quadratic cost function. It has been shown that the solution for this type of problem is provided by the Linear Quadratic Regulation (LQR) controller. Essentially, the algorithm helps automating the search of an optimal controller, set by minimisation of the cost function, which is often a function of measurement's deviations from their desired values.

This technique has been effectively applied and tested for quadrotors in various studies, including [Bouabdallah, Noth, et al., 2004; Cowling et al., 2007] and more recently in [Argentim et al., 2013; Khatoon et al., 2014; Martins et al., 2019; Okyere et al., 2019]. It has been validated through simulation and experiments, and has shown great capabilities for trajectory tracking, even under external disturbances.

Note that for this kind of controller, a good model is needed, because otherwise the optimisation might output incorrect gains, and the system will not be able to deliver its best closed-loop performance. This is the case for aggressive trajectories, for which the drone has to tilt a lot in order to produce a high horizontal thrust, *i.e.* the near hovering linearised model is not a good approximate of the UAV's behaviour.

2.3.1.3 H_∞ controller

In the context of robust tracking w.r.t. model parameter uncertainties, specific algorithms have been developed to address the issue. In particular, the H-infinity (H_∞) approach is widely used for linear, time-discretised systems, and aims to provide controllers that have minimal sensitivity, see *e.g.*, [Vinnicombe, 2001].

Such algorithms are common in practice, because they guarantee the basic control performance requirements, plus they are able to solve the modelling error. This optimal control method seeks to find a stable classical dynamic feedback loop, that minimises the H_∞ norm of

the system between unwanted signals and output errors, thus minimising the worst-case scenario effects of undesired signals, *i.e*. over all possible inputs. Therefore, it is considered a robust control technique. H_∞ controllers have been successfully implemented in various studies, *e.g*. [Araar et al., 2014; Hamza et al., 2022; Kang et al., 2015], and further validated by the simulations and experiments therein.

However, it is important to note that this approach requires a high level of mathematical understanding, and incurs significant computational costs.

2.3.1.4 Summary on linear controllers

The linear control policies discussed in Subsection 2.3.1 have successfully shown their ability to stabilise a quadrotor UAV and perform the tracking of a reference output trajectory. In terms of robustness against deviations in the model parameters, they can be ordered from least to most robust:

— the LQR controller is optimal in terms of tracking performance when the model parameters are known with utmost accuracy, but it is obtain through an optimisation procedure that exploits the system's dynamical model, and is thus not very robust against parametric uncertainties;

— the PID control policy does not require a good model of the system, and provides an average robustness;

— as H_∞ strategy is tailored to be robust, and places itself far ahead of the other two.

Nonetheless, linear controllers require the dynamics of the AR to be linearised, meaning that they can only perform really well when the approximation is correct, *i.e*. when the system is near hovering. Consequently, non-linear techniques are further examined to overcome this limitation.

2.3.2 Non-linear flight controllers

Non-linear control theory tackles complex systems that exhibit non-linear and/or time-varying behaviours. Unlike in linear control theory, non-linear controllers can handle a broader range of systems, which do not abide by the superposition principle. In reality, all systems possess non-linear characteristics, and their dynamics are thus often governed by sets of non-linear differential equations. Classical quadrotors, due to their under-actuation, have highly non-linear dynamics, and thus non-linear flight controllers are warranted for better performance. In this subsection, we review some of the prominent non-linear algorithms that have been developed for quadrotor UAVs.

2.3.2.1 Dynamic feedback linearisation controller

As explained previously in Subsection 2.2.2, differential flatness is a property that allows non-linear dynamical systems to be controlled in a manner similar to linear systems. Classical quadrotors feature this interesting property, thus all states and inputs can be expressed in terms of a suitable flat output (*e.g.*, the Cartesian space position and the yaw angle) and a finite number of its derivatives, as described in [Murray et al., 1995; Van Nieuwstadt et al., 1998]. This enables the creation of a DFL control policy, which allows to generate smooth trajectories in the space of the flat output, to be tracked by the under-actuated AR.

Therefore in [D. Lee et al., 2009; Mellinger and Kumar, 2011], the flatness property is exploited to transform the non-linear dynamics into a linear equivalent. Then, based on the inverted dynamics, a DFL control law is proposed, to ensure the global exponential attractiveness of the feedback control loop. While these studies have validated the potential of the strategy to track references with utmost accuracy, they have also shown that the DFL algorithm is sensitive to sensor noise because high degree of derivation needed to perform the linearisation, and it has also very limited robustness against model parameter uncertainties.

2.3.2.2 Back-stepping controller

The back-stepping approach is a non-linear control method that has been explored for a specific class of systems, as reported in studies [Fantoni et al., 2002; Sepulchre et al., 2012]. This recursive method works by designing the control law for the innermost subsystem of a robot, and then successively designing controllers for outer subsystems, until the external control is reached.

If a quadrotor's model is written in an appropriate form, *e.g.* divided into three subsystems – an under-actuated subsystem, a fully actuated subsystem, and a propeller subsystem – this technique can be applied, and a synthesis of the complete strategy can be described in several steps, to ensure and asymptotically stable feedback control system.

Simulation studies, *e.g.* [Bouabdallah and Siegwart, 2005; Madani et al., 2006] have demonstrated good performance in tracking the Cartesian position and heading angle of a quadrotor UAV. To summarise, the back-stepping method is fast-converging, requires low computational resources, and can handle disturbances effectively. However, as the construction of this algorithm is highly model-dependent, its robustness against parametric uncertainties is limited.

2.3.2.3 Geometric tracking controller

As the quadrotor UAV is an under-actuated system, it must first orient itself in a desired direction before it can move towards it. Therefore, it is crucial for the orientation coding to be as efficient as possible for control. For instance, Euler angles, that consist of pitch, roll, and yaw angles, are often used to mathematically represent the orientation of a body in space, w.r.t. a fixed frame. However, this three-gimbal solution is subject to a loss of degree of freedom when the axes of two of the gimbals align, causing the system to be unable to rotate about one axis.

Recognising this well known issue, a geometric non-linear controller for a quadrotor was proposed in [T. Lee, Leok, et al., 2010b]. This design expresses the system's dynamics on the configuration manifold of the special Euclidean group $\mathrm{SE}(3)$. The controller is capable of stabilising six DoFs using the four thrust inputs, while tracking the drone's linear position and heading angle. The controller exhibits exponential stability, as long as the initial attitude error is small. This design has shown excellent results in tracking fast and acrobatic manoeuvres for a quadrotor, as demonstrated in [T. Lee, Leok, et al., 2010a].

More recently, the controller has been advanced to a more robust version. In [T. Lee, Leok, et al., 2013], the authors demonstrated successful completion of complex flight manoeuvres in numerical simulations, by switching between three control laws, with the assumption of bounded and additive uncertainties in the quadrotor's dynamical model. In [Gamagedara et al., 2019], the geometric controller was updated, to take into account the mechanism of moment generation in control law design, as the dynamics of the attitude angles were assumed to be under-actuated. The improved control method in this version exhibits improved reliability and consistency, with a longer response time around the yaw axis, but maintained close to zero errors in critical pitch and roll angles for linear position tracking, resulting in superior position tracking performance compared to the original [T. Lee, Leok, et al., 2010b] controller.

2.3.2.4 Modern predictive control

In the field of aerial robotics, the study of optimal control has been a longstanding focus. This branch of control theory aims at minimising a specified cost function within a defined domain. Model Predictive Control (MPC), as described in [Camacho et al., 2013], is a sophisticated form of optimal control, that leverages the history of control inputs, to predict the future actions of a robot.

Unlike LQR, MPC optimises the current control inputs over a finite horizon at successive sampling instants, instead of generating a single optimal input over the entire time horizon.

Although LQR has better stability properties compared to MPC, it lacks performance when operating outside of the linearised model. MPC, on the other hand, can handle non-linear systems, as described in [Allgöwer et al., 2012], by considering the system dynamics and various constraints and objectives, through local linearisation of the system's non-linear dynamics, and solving the optimisation problem at each sampling time over the receding horizon. While stability proofs of non-linear MPC have not been fully established, only being demonstrated for unconstrained problems, the efficiency and robustness of this control approach have been empirically demonstrated for non-linear systems.

In [Sun, Romero, et al., 2022], a comprehensive comparison between non-linear MPC and DFL control was conducted, showing that MPC outperforms DFL in terms of robustness against model uncertainties. In [Kamel, Burri, et al., 2017], the authors compare linear and non-linear MPC for AVs, concluding that the non-linear version offers several advantages, including improved disturbance handling and tracking accuracy, as well as faster computation speed.

2.3.2.5 Sliding mode controller

The Sliding Mode Controller (SMC) theory, described in [Edwards et al., 1998; Young et al., 1999], employs an algorithm that adjusts the dynamics of a non-linear system, and guides the system states towards a specific surface, known as the sliding surface, in the state space. The feedback control law is a discontinuous function of time, and alternates between various continuous structures, based on the current state measurements. To design this controller, it is necessary to choose an appropriate sliding surface, that meets the desired behaviour specifications, and then select the control law to attract the system state to the switching surface.

Sliding Mode Controllers (SMCs) offer several advantages, including the ability to customise system behaviour through the design of the sliding function, and insensitivity to uncertainties such as model parameter variations, external disturbances, and bounded non-linearities. Researchers [Bensalah et al., 2019; Bouabdallah and Siegwart, 2005; Efe, 2007; Eltayeb et al., 2020; Labbadi et al., 2019] have successfully implemented and tested the full closed-loop SMC for quadrotor control, showing its ability to stabilise the UAV. The proposed control strategy has also outperformed a DFL controller, in the presence of parametric uncertainties.

2.3.2.6 Summary on non-linear controllers

In this subsection, we have reviewed several non-linear control strategies, prominent in the UAV community. Although each has its own characteristics, all are capable of performing track-

ing tasks correctly. Regarding their robustness against deviations in the model parameters, we can rank them as follows:

— the DFL controller offers superior tracking performance without parametric uncertainties, however it clearly lacks robustness as it strongly relies on the model of the robot;
— the back-stepping control policy, which requires low computational resources, also relies on the model, thus it is not very robust;
— the geometric tracking controller has an average robustness;
— MPC is quite robust, since it leverages the measurements and the previous predictions to do the optimisation over a time window, thus adapting efficiently;
— the SMC offers great robustness against uncertainties in the model parameters, but requires a more complex design and adjustment to avoid chattering.

In essence, these algorithms can be improved to address various issues: recent versions of the geometric tracking controller have made it more robust, or MPC can be adapted to be more robust, *e.g.* by adding unmeasured noise in the process, *etc*. Nonetheless, it would be interesting to design controllers that could adapt, and handle a wide range of challenges, without prior knowledge. In the next subsection, we examine the capabilities of various intelligent control policies.

2.3.3 Intelligent flight controllers

Intelligent control algorithms are developed by emulating specific behaviours found in biological systems. The characteristics include adaptation, learning, planning under possible large uncertainties, and handling large amounts of data. The recent advancements in computing technologies have allowed to evaluate the effectiveness of these advanced techniques.

2.3.3.1 Fuzzy logic controller

In control theory, a Fuzzy Logic (FL) controller, as described in [Passino et al., 1998], operates by utilising mathematical systems, with logical variables that can take continuous values between 0 and 1. Unlike conventional control methods, FL can handle non-linear systems and uncertainty in data, while being highly modular. In a FL controller, measured errors are mapped by fuzzy sets, in order to create linguistic representations, that can be understood by humans. A set of if-then rules then guides an inference engine, to simulate human decisions from the fuzzy variables, and output further fuzzy variables, which are eventually defuzzified, as system inputs.

A case study in [Santos et al., 2010] used FL for a quadrotor UAV control system, with four strategies designed to track height, pitch, roll, and yaw references. An associated PID-like controller was then defined for each possible motion, including a detailed height control law. Simulated flight results were promising, with the authors highlighting the importance of parameter tuning, and the need for future experiments, to demonstrate the robustness and efficiency of the control strategy.

In another study, [Coza et al., 2011] proposed an adaptive FL controller, for robust quadrotor stabilisation, in the presence of wind disturbances. In comparison to SMC, which can result in chattering of the control signal, the FL controller showed great stability in simulations.

2.3.3.2 Neural networks controller

Artificial Neural Networks (NNs), see *e.g.* [Abdi et al., 1999; C. M. Bishop, 1994], are widely used to control systems nowadays, especially non-linear ones. In fact, the goal of a NN is to replicate the abilities of the human brain, among which adaptability is the most sought after. Thereby, NNs are originally inspired by the behaviour of a biological central nervous system, in order to introduce the desired brain functionalities. Typically, a NN is a mathematical function that maps a given set of inputs to a desired set of outputs. This tool is specifically designed and then trained for a specific application, using a database of processing examples, which contain a known input and the associated desired output. The training process is usually conducted by successively computing the error between the output given by the NN for a specific input, and the desired target output. At each iteration, and through the whole database, the network adjusts its internal weights, according to the learning rule and the computed output error. If the database is sufficiently large and includes enough variation in the inputs, the NN will tend to produce outputs which are increasingly close to the target outputs. The great advantage of Artificial Intelligence (AI) is that is can learn without the need of explicit programming, but rather by providing database of examples associated with the target result. Then, a simple gradient descent allows to adjust the parameters of the network, which optimises it for the sought application.

Of course, as NNs have been very popular in recent decades, works such as [Dierks et al., 2009; T. Lee and Y. Kim, 2001] have implemented feedback control laws based on these latter, in order to test their soundness for quadrotor UAVs. Even in the presence of unmodeled dynamics and unknown bounded disturbances, the controllers have successfully tracked all six DoFs. The virtual control structures allowed to obtain the desired translational velocities with the correct attitude tracking of the UAV. Simulation results confirm the theoretical conjectures. In [T. Lee and Y. Kim, 2001], the stability is analysed with Lyapunov theory. However, the

major drawback of NNs is the high computational power required for training, with a suitable training database, for the desired application.

2.3.4 Summary on flight controllers

It is clear that no single linear, non-linear, or intelligent control algorithm for quadrotors can outperform the others, on all criteria. Researchers have attempted to overcome this challenge by developing hybrid controllers, which combine the principles of several control policies. Although some of these attempts have shown improved versatility and robustness, each improvement from hybridisation also has a drawback, inherent in its design. As a result, even with hybrid controllers, it is impossible for one controller to outperform all others. There are too many performance characteristics to consider, such as robustness, speed, precision, ease of tuning, adaptability, and computational cost, that a trade-off must always be made. The optimal controller will depend on the specific application. Switching between controllers during flight is possible, but it requires designing and implementing multiple controllers, as well as defining the switching conditions.

Once a controller is designed or chosen, it can deliver a closed-loop performance that reflects its characteristics. On the first hand, for many controllers, the control inputs are determined using a mathematical model of the quadrotor's dynamics, which relies on accurate parameter measurements. On the second hand, as one can see in Fig. 2.3, the tracking error is also influenced by the quality of the state measurement. Thus, regardless of the selected controller, the overall performance of the feedback control system depends on the accuracy of the state observer, and the measurement of model parameters.

2.4 State and parameter estimation

For robots to act autonomously, they must have the ability to determine their location at a given instant, *e.g.*, when reaching a target to carry a task. This understanding is achieved through perception: as for any living organism, it consists of acquiring information through sensors. In robotics, sensors are detection devices that convert physical inputs into electrical signals, that can be further processed by the robot's control system. However, the conversion process is not perfect and can introduce errors, such as drift and measurement noise. This means that the sensor readings are not always accurate representations of the actual physical data.

Moreover, as explained in Section 2.3, as parameters of the dynamic model are often im-

planted inside the control algorithms, they must be accurately estimated. The parameters estimates are made through measurements, which can range from simple methods, such as weighing the robot, to more complex experiments, such as measuring aerodynamic thrust coefficients in a wind tunnel. The quality of these measurements directly impacts the accuracy of the parameters used in the control algorithm, with better quality measurements resulting in lower parametric uncertainty, thus better performance of the feedback control system.

After a brief overview of traditional state estimation techniques for ARs, we will explore methods that enable real-time parameter estimation.

2.4.1 State observers for aerial robots

A feedback control system, as illustrated in Fig. 2.3, requires measurement tools, to compare the actual output of a system to the desired output. As already highlighted, through the conversion process of sensors, drift and noise errors may be introduced in the measurements. To address this issue, researchers have developed state estimation techniques that integrate data from the various sensors available for the robot, to produce an estimate of the varying quantities of interest, namely the system's state. Typically, most state observers exploit the system's dynamic model, known control inputs, and multiple sensor measurements, to compute an estimation of a state, that is likely to be more precise than the estimate obtained by using the single measurement alone. Additionally, in most cases, the state of the system is larger than the number of directly measured variables. However, if designed correctly, state observers can still reconstruct and estimate the full state.

In control and estimation theory, Kalman Filter (KF), see [Welch et al., 1995], also referred to as linear quadratic estimation, is a widely used algorithm. The filter produces more accurate estimates of desired variables by exploiting a combination of system dynamics, measurement data and sensor fusion, to mitigate uncertainties caused by measurement drifts and noise. As shown in Fig. 2.4, the estimator calculates a weighted average of the current prediction and the new measurement at each time sample, to update its estimate. The Kalman gain is chosen to minimise the *a posteriori* covariance of the estimator. A study in [Wendel et al., 2006] focused on the state estimation of a quadrotor UAV using two different KFs, with the primary sensor being an Inertial Measurement Unit (IMU), as the INS, and also a secondary sensor being a small GPS receiver. The aim was to achieve sufficient attitude accuracy for stability under various operating conditions, including GPS outages (when the GPS was unavailable, a switch was made between the two KFs). The study showed that, even in the absence of GPS measurements, the attitude estimation was precise enough.

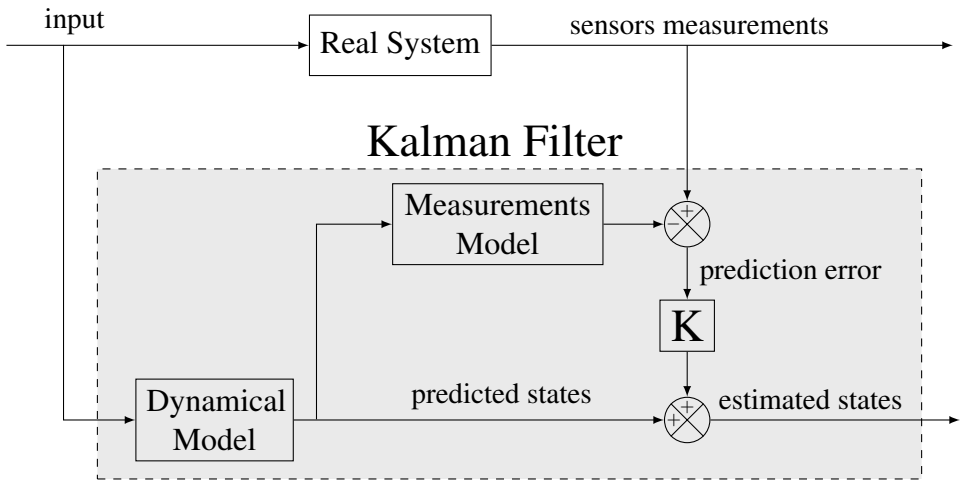

Figure 2.4 – Architecture of a Kalman filter.

In [Xiong et al., 2015], an Optimal Kalman Filter (OKF) method was developed, to assess the accuracy of full state vector estimation for a quadrotor UAV, in the presence of white Gaussian process and measurement noise. The dynamics of the quadrotor were modelled as a discrete time system through linear extrapolation. The performance of the full feedback control system was tested through a series of simulations, which showed that the method effectively reduced variance and improved estimation accuracy.

The original KF estimation algorithm can be adapted to produce an Extended Kalman Filter (EKF), which incorporates the non-linear dynamics of the system into the estimation process. As for the KF, the EKF is divided in two main components, the prediction and the correction steps. The prediction step consists of propagating the states forward using the non-linear system's model, as well as the measurements model. Then, the correction step refines the estimates, using the current available measurements and the predicted states. The performance of the EKF has been evaluated in various studies, such as [Ascorti, 2013; Raja, 2017]. The filter was tested using standard IMU sensors, consisting of accelerometer, gyroscope, and magnetometer. Despite the presence of biases and faults in the IMU sensors, the EKF was able to accurately estimate the states of a quadrotor in real-time.

The estimation process is crucial, especially when performing aggressive manoeuvres, in situations where onboard sensors can partially fail or in GPS-denied environments. [Goodarzi et al., 2017] has improved upon previous works on the control of quadrotors [T. Lee, Leok, et al., 2010a, 2010b, 2013], by formulating an EKF on the special Euclidean group $\mathrm{SE}(3)$. The proposed filter showed desirable properties in numerical simulations, including accurate estimation of translational velocity despite large initial error and high noise measurements, and

effective operation without GPS data. Real experiments further validated the proposed method's capabilities, with angular velocity and attitude measured from an IMU, and position from a Motion Capture (MoCap) system. The results showed that the EKF provided precise estimation, in multiple scenarios.

As already explained several times, quadrotors are subject to aerodynamic forces, uncertainties, and external disturbances. In that context, sliding mode observers are commonly used for state estimation, and evaluation of external disturbances, such as wind and noise. As noted in [Benallegue et al., 2008], these observers offer several benefits, including immunity to unknown inputs, the ability to identify inputs, and finite-time convergence to the state vector. This study combined a DFL controller with a high-order sliding mode observer, to control a the UAV. The aim was to avoid complex non-linear control solutions. The observer was used to counteract the non-linear effects of external disturbances, and simulations showed that accurate inner estimations of these disturbances enabled the development of a stable and robust feedback control system. The observer design successfully reconstructed unmeasured states.

Designing experiments that yield accurate state estimation requires careful consideration of the control inputs that will drive the system. An effective approach is to generate specific trajectories that maximise the observability of the system's states. In this regard, the non-linear observability metric provides insight into the observability of a system, and identifies control inputs that can improve the accuracy of the estimates. The authors of [Hausman, J. Preiss, et al., 2017] propose a framework for trajectory optimisation, that facilitates self-calibration of UAVs. This framework, which is not tied to any specific state estimation method, focuses on generating motions that render the state more observable. The authors use a Taylor expansion to approximate the local observability Gramian, by integrating over the trajectory, and then use this metric to define the cost function in the trajectory optimisation problem. A simulation study involving a quadrotor UAV equipped with an EKF IMU/GPS state estimator demonstrates the effectiveness of the observability-aware trajectory generation method, which is shown to outperform standard methods such as minimum snap [Mellinger and Kumar, 2011].

Afterwards in [J. A. Preiss et al., 2018], the Expanded Empirical Local Observability Gramian (EELOG) is defined as a suitable metric for evaluating the observability quality. As outlined in [S. M. Weiss, 2012], a state is considered well observable if a marginal change in the state results in a significant change in the system's output, while it is poorly observable if a large change in the state causes only a small change in the output. The theory presented in [J. A. Preiss et al., 2018] is supported by extensive simulations on a quadrotor UAV, that tend to demonstrate the correlation between the minimised EELOG cost function value and the estimation accuracy. For

a real AV, using calibration-aware trajectories leads to improved results (compared to heuristic trajectories), thus validating the effectiveness of the framework, in terms of the IMU/GPS position state estimation.

The objective of state estimation is to accurately determine a key variable of the system being controlled. By exploiting the system's dynamic model, state observers use available measurements to reconstruct accurate data, for use in the control loop. Parameters of the model, which are generally considered unknown but static values, can be estimated in a simpler manner compared to the state variables.

2.4.2 Online parameter identification

Accurate knowledge of the parameters that describe the dynamics of robots is critical to achieving high performance. This information is essential for setting the controller, in order to optimise the navigation's accuracy and reliability. Most of the time, system identification is performed offline, prior to any control task. An alternative point of view, which stems from adaptive control, see [Åström et al., 2013], is to provide an improved estimate of the parameter values in real-time, through an online estimation process. By including estimation dynamics in the controller, it is possible to update the model's parameters, using available measurements. This technique is commonly used in rocket launches, where the rapid consumption of fuel requires accounting for changes in total mass during flight. It can also be used to improve the accuracy of parameters that are assumed to be static, but are inaccurately measured during offline system identification.

A possible approach for estimating the parameters of a quadrotor UAV is by using a state observer. As the dynamics are highly non-linear, in [Abas et al., 2011] the Unscented Kalman Filter (UKF) is introduced, since it offers a more accurate estimation than the EKF. Unlike the EKF that constructs a linear approximation of the non-linear transformation function using a first-order Taylor expansion, the UKF generates an approximation with multiple sigma points. When the function to be estimated is highly non-linear, the UKF leads to higher resolution belief and generally better results. The UKF method was applied to a quadrotor to estimate moments of inertia at x, y, z-axes and rotor inertia. The simulation results indicate the effectiveness of the UKF in real-time data analysis, when the model and identification data are suitable, showing good convergence time and reliable estimations.

To determine unknown or inaccurate parameters in a non-linear dynamical system, the batch least-square procedure is commonly used [Bar-Shalom et al., 2004]. The goal is to compare the expected trajectory of the system with experimental measurements, for a given set of control

inputs. This comparison is repeated multiple times, and the set of parameters that minimise the least-square difference between the model and measurements is retained. If the model is correct, the state observer is precise, and the initial parameters are in the neighbourhood of the actual values, the estimated parameters will converge to the true values. Motivated by the fact that delivery quadrotors can experience changes in properties during package grasping or release, [Dhaybi et al., 2020] applied the recursive least-square estimation algorithm to such UAVs, with covariance resetting. In this research, an experiment was designed to continuously generate exciting inputs, ensuring the convergence of the estimated parameters towards the actual values. The simulation and experimental results demonstrate the estimator's accuracy, efficiency, and convergence rate.

Based on the EELOG metric, see Subsection 2.4.1, a new research study presented in [Böhm, G. Li, et al., 2020] aims to plan informative trajectories, for self-calibration of geometric and inertial parameters of a quadrotor. The state vector of the UAV is extended with a time-independent vector that includes mass, moments of inertia, position of the CoG, and poses of the sensors in use. To maximise the quality of observability, as determined by the EELOG metric, the researchers use an framework that optimises the EELOG cost function. The soundness of the strategy is assessed through experiments on a real UAV, with an EKF as the state observer.

The extensive research conducted on observability, see, *e.g.*, [Böhm, G. Li, et al., 2020; Hausman, J. Preiss, et al., 2017; J. A. Preiss et al., 2018], has led to further investigation. In [Böhm, Scheiber, et al., 2021], the observability of the extended state was evaluated for various multi-rotor UAV configurations, and four different sensor setups. The extended state estimation included parameters such as the position and orientation of each AU, thrust and drag coefficients of each propeller, mass of the AV, moments of inertia, and others. The estimation was performed using an Iterative Error-state Kalman Filter (IEKF), and the generated trajectories, which were produced by an aggressively tuned MPC, sufficiently excited all six DoFs. In this work, the authors suggest to use the more elaborate observability-aware trajectories from [Böhm, G. Li, et al., 2020], in order to improve even further the convergence of the estimates.

2.4.3 Summary on state/parameter estimation

In Subsection 2.4.1, we have examined various state estimations techniques that can be beneficial for the control of ARs. These techniques provide a solid foundation for further extending state observation, to include the estimation of system model parameters, as discussed in Subsection 2.4.2. The generic feedback control system depicted in Fig. 2.3 is composed of three

key elements: the system, the controller that enables it to execute navigation tasks, and an observer, that is responsible for providing accurate feedback information, and closes the control loop. Although each of these components has been previously reviewed, successful autonomous navigation of robots depends on the ability to generate well-designed reference motions.

2.5 Trajectory generation for aerial robots

Trajectory planning is of utmost importance in the field of modern robotics and automation. Regardless of the mechanical structure, the completion of a given task is based on the execution of specific movements imposed on the robot. This is particularly crucial for mobile systems such as quadrotor UAVs, with recent, complex applications, that require operating in cluttered indoor environments. Factors such as under-actuation, aerodynamic effects, external disturbances, geometric and actuation constraints, make the task even more challenging. Nevertheless, in recent years, a great deal of effort has been put into developing frameworks that cater to the diverse requirements of various applications. A review of many planning algorithms can be found in [LaValle, 2006].

2.5.1 Path finding

As UAVs often encounter adverse conditions in both indoor and outdoor environments, a competent path planner is necessary to chart a successful course for their mission. In this subsection, we present an overview of various path planning approaches, serving as a foundation for trajectory planning. For a comprehensive survey of the available algorithms, we refer to [L. Yang et al., 2014].

Random sampling techniques, such as the Probabilistic Road-map Method (PRM) [Kavraki, Kolountzakis, et al., 1998; Kavraki, Svestka, et al., 1996], can solve the problem of providing effective paths for mobile robots. The path determination process is divided into two stages. The first stage, which is completed offline, involves constructing a complete network graph of the robot's environment using available information. In the second stage, the graph is used to connect the starting point of the AR to its destination. Once the global path is established, a smoothing method can be applied to find the shortest path. In a study of a quadrotor UAV [Chen et al., 2019], the PRM was combined with the Artificial Potential Fields (APF) method [Khatib, 1986]. The resulting algorithm generated a repulsive force field around obstacles and an attractive force field towards the target, resulting in a more efficient and improved strategy, compared

to the traditional PRM. The authors acknowledge that their framework only considered static environments, and that the dynamics of AVs should be addressed in future research.

The Rapidly-exploring Random Trees (RRT) algorithm [Hsu et al., 2002; Kuffner et al., 2000] is a widely used random sampling strategy for path planning. It starts by growing a tree from the initial point in state space, by using random samples in the search space. Each generated sample is connected to the nearest point in the tree, and if the connection is feasible, it is added to the tree as a new possible state. The growth of the tree can be guided by adjusting the probability of generating samples in certain areas. In [Bouzid et al., 2017], an upgraded version of the RRT is combined with the Travelling Salesman Problem (TSP) solved using a Genetic Algorithms (GA) to generate optimal paths for a quadrotor in cluttered environments. This two-stage approach balances the objective of avoiding collisions with obstacles while minimising the energy consumption of the AV.

Dijkstra's algorithm [Dijkstra, 1959] has been utilised for finding optimal paths, in a known environment with obstacles or constraints. This procedure, which exists in many variants, determines an adequate cost function, further used to optimise the path. For instance, [Gautam et al., 2013] presents a framework that uses Dijkstra's algorithm to plan a minimal distance and obstacle-free path for a quadrotor, equipped with a self-tuning fuzzy logic controller.

Another trend to path planning involves drawing inspiration from biological systems. One of the key advantages of these methods is that they do not require complex environment models. For example, the efficacy of Evolutionary Algorithmss (EAs) for quadrotor path generation has been demonstrated in studies like [Hasircioglu et al., 2008]. This optimisation approach uses natural selection to identify the fittest individuals within a population. The best individuals are chosen as parents to produce offspring for the next generation, and the population continues to evolve towards an optimal solution, through repeated cycles of selection and reproduction. In some applications, such as rescue operations, the speed of exploring an environment is critical. To address this issue, [Cheng et al., 2009] developed an Ant Colony Optimisation (ACO) path planner for a swarm of UAVs. The results of the simulation show that this cooperative method can significantly enhance the overall coverage of the AV.

The primary distinction between path planning and trajectory planning is that the latter is defined by time. Most path finding methods only focus on determining a geometric reference, without taking temporal considerations into account. To guarantee that a navigation task is feasible for an UAV, it is imperative to consider the limitations of its actuation system. Trajectory planning, which encompasses path planning, consistently addresses this critical issue.

2.5.2 Motion planning

In general, TO problems are formulated to minimise a cost function of interest, while satisfying geometric, kinematic and actuation constraints, ensuring that the resulting reference trajectory is suitable for the tracking. For quadrotors, as the capabilities continue to increase, the problem has been addressed many times in order to maximise the potential of new applications. Moreover, a system that operates quickly is always preferred. Therefore, the challenges brought to trajectory planning for UAVs require that the motion:

— is collision-free, and the drone safely avoids all obstacles, *i.e*. the tracking is accurate enough;
— is dynamically feasible, *i.e*. the inputs never saturate;
— allows for minimum execution time of the task;
— is computationally efficient.

As for any robot, the performance of quadrotor is limited by their AUs, because the currents of the BLDC motors are bounded, to avoid overheating. This highlights the importance of ensuring that UAVs operate within the safe limits of their actuators. To address this issue, the work of [Hehn et al., 2011] focuses on developing an algorithm that generates feasible trajectories for drones. The input saturations are defined as non-linear constraints in an optimisation problem, which generates a trajectory from a starting point to a final target, subject to kinematic limit conditions. The effectiveness of this approach has been validated through experiments.

The pioneer work of [Mellinger and Kumar, 2011] presents a minimum snap algorithm for aggressive flight. The authors aim to address the limitations of previous research, that used near hovering flight controllers, and linearised dynamics under the assumption of small pitch and roll angles. As this assumption is not suitable for all applications, the authors provide a trajectory planning methodology based on the quadrotor's differential flatness property (see Subsubsection 2.3.2.1), to design smooth motions that pass through key-frames of space position and yaw orientation, at specified times. The optimisation problem minimises the integral of a weighted combination of squared position snap and squared yaw acceleration, while ensuring continuity between each piece, kinematic conditions at the start and end of the motion, and actuation limits. Additionally, corridor constraints can be imposed to ensure that the motion follows a specific path to reach a key-frame. The soundness of the method was verified through experiments on a real-life quadrotor, demonstrating its ability to design reference motions for aggressive flights in challenging environments.

In [Richter et al., 2016], the minimum snap algorithm was extended by combining it with the geometric controller of [T. Lee, Leok, et al., 2010b]. This resulted in a polynomial TO frame-

work that minimises the integral of the snap during the motion, while considering obstacle avoidance, and input saturation constraints. A comparison was made between this framework and RRT* with a polynomial steer function, which showed that the new method was faster and more stable, in terms of generation speed and numerical stability. The differential flatness property allowed for a low-dimensional search for the shortest path, which could then be translated into a dynamically feasible trajectory.

The Co-variant Hamiltonian Optimisation for Motion Planning (CHOMP) procedure [Ratliff et al., 2009] has evolved from traditional path planners (Subsection 2.5.1), and is used for the design of trajectories with dynamic and task-specific criteria. This efficient motion planning technique handles obstacle avoidance and optimal control simultaneously. As it was originally tested on manipulators and quadrupedal robots, CHOMP served as an inspiration in [Oleynikova, Burri, et al., 2016], which proposes a real-time TO framework, for quadrotor UAVs. The approach generates obstacle-avoiding trajectories using a piece-wise polynomial trajectory, subject to continuity conditions, and minimises a cost that includes the integral of the squared snap, as well as collision metric. The optimisation procedure is put to use in an online re-planning system that:

— updates mapping information to detect new obstacles during the exploration;
— finds a free point to generate a path toward the desired target with the new map;
— performs the trajectory re-planning to the new point with a function of Euclidean Signed Distance Field (ESDF) as the collision map.

The soundness of the procedure has been validated in a forest environment for a quadrotor, and compared to RRT* and standard CHOMP. The advantage with this new technique is the representation of the trajectory, which is inherently compact. Besides, it is able to better control the derivatives of the position, *i.e.* the snap. The use of the ESDF as the map representation for the motion planning has a limitation, as it only provides binary information about the space occupation. This becomes a problem when the sensors measurements are not dense enough, leading to an over-estimation of occupied space, due to high uncertainties in the measurements. To overcome this challenge, the method was improved in [Oleynikova, Taylor, et al., 2017], resulting in an even more efficient framework.

The time it takes to complete a task is a critical factor in motion planning, as a shorter duration is often more advantageous. To address this issue, [Gao et al., 2018] proposes a two-step algorithm. First, it generates a smooth and safe spatial path, and then decides how a quadrotor should move along this path, in order to minimise the time of the motion, while satisfying kinodynamic limits. The spatial path is represented by a piece-wise Bézier curve, generated with

minimal spatial jerk and arc length, for curve regularisation. The motion planning problem is set with the total time as the cost function, and is subject to continuity constraints between each piece, input saturation constraints, and kinematic constraints at the starting and ending points. The decoupled method has been shown to be effective in simulations and experiments, as it allows for full utilisation of the quadrotor's actuators. This issue has also been addressed in [Penin et al., 2018], with the added consideration of state reconstruction, using an EKF that integrates the measurements of standard onboard sensors and a camera, in cases where the state is not directly available. For flat dynamic systems, this planner takes into account the perceptual limitations, and provides minimum-time trajectories that are dynamically feasible.

UAVs consume a large amount of power, leading to limited flight endurance, even with good batteries. Therefore, researchers have focused on energy-aware trajectory planning. For instance, [Morbidi et al., 2016] presents (and validates the theory through numerical experiments) an optimisation problem that exploits the electrical model of a BLDC motor, to design trajectories that minimise energy consumption [Chamseddine et al., 2012] proposes a flatness-based method for minimum-time/energy trajectory planning/re-planning, and demonstrates its effectiveness for real-time applications.

2.5.3 Summary on trajectory generation

The path and motion planning methods discussed in Subsection 2.5.1 and Subsection 2.5.2 address a range of typical challenges faced in modern mobile robotics applications. This includes strategies for generating obstacle-free, dynamically feasible, smooth, and kinetically constrained paths and motions, as well as minimising time or energy consumption. As previously established in Section 1.1, it is critical for systems to operate with high accuracy and reliability, despite uncertainty in the dynamics model and actuation. However, none of the methods in Section 2.5 consider the effect of parameter uncertainty on the AVs tracking capabilities. Some literature acknowledges the issue, but fail to provide a solution. In the next section, we will examine the current state-of-the-art in robust trajectory generation for robotic systems.

2.6 Robust trajectory planning

This section presents the main corpus of works upon which this thesis is built. It focuses on the study of robust trajectory generation. As previously mentioned, AVs and especially quadrotor UAVs are subject to model uncertainties. Therefore, the main challenge here is to overcome

the impact of model uncertainties on the behaviour of the robot. Uncertain measurements of the *nominal* values, *e.g*. of a propeller thrust and drag coefficients, can deeply affect the control action, and thus the general performance of the system. Classical control policies for UAVs were reviewed in Section 2.3, but robust controllers often require increased hardware capacities, and may not be desired for small, embedded systems. In Section 2.4, the trend of obtaining a better estimate of the parameters while the robot operates was discussed, but online estimation may introduce a coupling between the estimation and dynamics, which might not be optimal. An alternative approach is to concentrate on finding feed-forward trajectories, whose very design makes them intrinsically robust against possible uncertainties in the system/environment models and parameters, and statistically enhance the accuracy of the tracking task.

The authors in [Candido et al., 2010] conducted research by exploiting the belief road-map method [Prentice et al., 2009], to tackle the challenge posed by the inherent noise in system components, as well as the imperfect modelling of robots, sensors, and environments. This approach, which is based on probabilistic robotics [Thrun, 2002], involves modelling the system, sensors, and environment as Partially Observable Markov Decision Process (POMPD) and using stochastic uncertainty propagation functions, to minimise the uncertainty along the path to the desired target, in the output space. The proposed framework has the advantage of being adaptable to a wide range of non-trivial robot models, with non-Gaussian processes and observation models for trajectory planning with minimum uncertainty. The strategy was put to the test by comparing three path generation methods using a plane unicycle. The results showed that the minimum uncertainty planning approach was able to identify paths that were unfeasible for tracking, due to potential collisions between obstacles. Moreover, the method was able to choose better nominal references that would never result in a collision, even with perturbed parameters, a superiority that sets it apart from other methods. An extension of this work was further explored in [Candido et al., 2011], with the introduction of a particle filtering algorithm, that tracks collision-free trajectories and estimates the probability of collision. This update resulted in paths that have reduced uncertainties at the final goal, and minimised the probability of collisions during the motion. The authors suggest that the framework could be improved by adding sensitivity functions along the trajectory to compute tubes around the nominal motion, which would help maintain the system within the desired bounds, with high likelihood.

A formulation of the *sensitivity* minimisation problem was carried out in [Kreindler, 1969], with the aim of generating robust trajectories for linear closed-loop feedback systems. This work laid the foundation for numerous advancements in the field of robotics over the years. In [Byrne et al., 1976], the theory was extended by comparing several algorithms for the optimal

linear regulator problem. The new algorithm was found to provide the greatest reduction in sensitivity cost with the least increase in net energy cost, thus demonstrating the promising potential for improved control of linear and time-invariant systems, through the integration of both feed-forward and feedback elements.

However, the dynamics of quadrotors are highly non-linear, rendering the previously mentioned methods inapplicable. To tackle non-linear systems, [Singer et al., 1990] investigated a method for shaping control inputs that result in vibration-free outputs, as well as being insensitive to errors in the structural damping coefficient. This robust command approach was tested on a space shuttle remote manipulator system with an uncertain natural frequency, and the achieved vibration reductions showed the efficacy of the method.

The problem of generating feed-forward trajectories with minimum *state sensitivity* has been recently addressed in [Ansari et al., 2013a, 2013b] and further expanded upon in [Ansari et al., 2016]. The latter work presents a method to generate robust trajectories through an open-loop optimisation routine, taking into consideration deviations in model parameters. Unlike probabilistic approaches, this framework assumes the models are accurate, and seeks to show that motions that minimise the *first-order sensitivities* result in reduced deviations around the nominal state when the model parameters are inaccurate. The method starts by formulating an optimal control algorithm, that minimises the state errors and control inputs. Then, using these non-linear systems and control laws, a TO problem is formulated, to reduce the parametric open-loop state sensitivity. The effectiveness of this procedure was demonstrated through tests on a four-wheeled robotic vehicle with adjustable CoM, which showed significantly less deviation from nominal motions when tracking control-aware optimised references (compared to heuristic trajectories) under deviations in nominal expected friction.

The study [Robuffo Giordano et al., 2018] builds upon previous research in feed-forward trajectory planning with minimum state sensitivity, by introducing the new concept of *closed-loop state sensitivity*. The focus of the research is to develop a control-aware optimisation process, that takes into account the coupling between the system and its controller when tracking a reference motion, and which is expected to provide trajectories, whose realisation by the tracking controller is most insensitive against variations in the model parameters. The closed-loop state sensitivity and its gradients are integrated to form a cost functional for the optimisation problem, which includes kinematic constraints at the start and end of the motion. The process is demonstrated through two case studies, a plane unicycle and a plane quadrotor, both equipped with a DFL controller. The uncertain parameters considered in the unicycle case study are the radius of the wheels and the distance between the wheels, while the inaccuracies considered in

the quadrotor case study include the thrust coefficient to mass ratio, the drag coefficient to inertia ratio, and the body frame air drag coefficients along the horizontal and vertical directions. A statistical analysis is performed for both systems, on a single trajectory for each, and the results provide strong validation of the proposed optimisation scheme, under the described conditions.

2.7 Summary of the literature review

In this chapter, we have presented an overview of the current state-of-the-art in robust trajectory planning for UAVs. In Section 2.2, we have examined different designs of AVs. The highlighted benefits and drawbacks of each mechanical configuration have allowed to understand the challenges related to their automation. In Section 2.3, we presented robust control algorithms for quadrotors, with a focus on those relevant to this thesis. Although some of these techniques offer improved robustness, they may also have a higher computational demand than standard controllers, making them unsuitable for embedded systems. In Section 2.4, another point of view has been brought, that rather focuses on online parameter identification. As this trend is strongly related to state estimation, both have been presented together. The last two sections of this chapter have focused on the generation of feed-forward trajectories. Common path and trajectory planning techniques for ARs have been examined in 2.5. Lastly, we focused our interest on the planning of trajectories whose very design makes them robust against model parametric uncertainties.

Building upon these rich works, this thesis is interested in defining a TO framework, that provides trajectories with minimum closed-loop state sensitivity. We believe that our method can significantly enhance uncertain robotics performances. Before presenting our main contributions, we first introduce the notation and formalism related to the design, control and TO for UAVs. These foundations will be needed for the development of our robust trajectory planning algorithms in Chapters 4 to 6.

CHAPTER 3

MODELLING

3.1 Introduction

Before delving into the core of this thesis, we provide some mathematical models that are used throughout the thesis, and subject to modifications in the contributions. First, we set the formalism with conventions, notations. After a brief discussion of the mathematical representations of orientation in space, a standard dynamical model of a quadrotor UAV is given. As several control policies and trajectory representations are used in the manuscript, their equations will be detailed when convenient.

3.2 Conventions and notations

This section gives the mathematical writing conventions used throughout the manuscript. All variables are written using Latin or Greek letters, with the following rules:

— normal font for scalars, *e.g.* s;

— small bold for vectors, *e.g.* $\mathbf{v}$;

— capital bold for matrices or higher order tensors, *e.g.* $\mathbf{M}$ or $\mathbf{T}$ respectively.

The null and identity matrices of sizes $n \times m$ are respectively denoted $\mathbf{O}_{n\times m}$ and $\mathbf{I}_{n\times m}$, and can be simplified as $\mathbf{O}_n$ and $\mathbf{I}_n$ when square.

Sets are written using dedicated letters, *e.g.* $\mathbb{R}$ for real numbers, $\mathbb{Q}$ for quaternions, and $\mathbb{S}$ for the unit quaternion sphere of $\mathbb{Q}$. Besides, continuous intervals are written using single straight brackets $[a,\, b]$, while discrete sets are written using curly ones $\{a,\, b\}$, and lastly, sets of successive integers are written using double straight brackets $[\![a,\, b]\!]$.

In Table 3.1, we present the main mathematical symbols that denote variables and parameters used in the thesis.

A last point to mention is that, inside the developments of the manuscript, functions will be defined, and further employed. All of the arguments, whether they are variables or parameters,

Definition	Symbol
State vector	$\mathbf{q}$
Input vector	$\mathbf{u}$
i-th rotor angular velocity	w_i
Output vector	$\mathbf{y}$
Linear position vector	$\mathbf{r}$
Linear velocity vector	$\mathbf{v}$
Linear acceleration vector	$\boldsymbol{\gamma}$
Rotation matrix	$\mathbf{R}$
Euler angles vector	$\boldsymbol{\eta}$
roll	φ
pitch	θ
yaw	ψ
Orientation quaternion	$\boldsymbol{\rho}$
Angular velocity vector	$\boldsymbol{\omega}$
Angular acceleration vector	$\boldsymbol{\alpha}$
Earth gravity acceleration constant	g
Mass	m
Inertia tensor	$\mathbf{J}$
Propeller thrust coefficient	k_f
Propeller torque coefficient	k_τ
UAV arm length	ℓ
State sensitivity matrix	$\mathbf{\Pi}$
Internal state sensitivity matrix	$\mathbf{\Pi}_{\boldsymbol{\xi}}$
Input sensitivity matrix	$\mathbf{\Theta}$
Output sensitivity matrix	$\mathbf{\Upsilon}$

Table 3.1 – Usual mathematical symbols used throughout the manuscript.

shall be explicitly stated in each of these definitions. Anyhow, after the first definition of a function, in order to simplify the mathematical expressions, some arguments can be omitted when further using the function, *e.g.* the time variable can be omitted when referring to the state. Though, it can be possible that sometimes we write them explicitly, when judged convenient for the understanding.

3.3 Mathematical representation of the world

A fundamental requirement in robotics is to be able to express positions and orientations, of the different components and objects that exist in the world. This section clarifies the mathe-

matical representation of the world.

We denote the world inertial frame $\boldsymbol{\mathcal{F}}_{\mathcal{W}} = \{\mathbf{O}_{\mathcal{W}}, (\mathbf{x}_{\mathcal{W}}, \mathbf{y}_{\mathcal{W}}, \mathbf{z}_{\mathcal{W}})\}$ where $\mathbf{O}_{\mathcal{W}}$ is its origin and $(\mathbf{x}_{\mathcal{W}}, \mathbf{y}_{\mathcal{W}}, \mathbf{z}_{\mathcal{W}})$ its canonical basis. The body frame attached to the system that navigates in $\boldsymbol{\mathcal{F}}_{\mathcal{W}}$ is denoted $\boldsymbol{\mathcal{F}}_{\mathcal{B}} = \{\mathbf{O}_{\mathcal{B}}, (\mathbf{x}_{\mathcal{B}}, \mathbf{y}_{\mathcal{B}}, \mathbf{z}_{\mathcal{B}})\}$. $\mathbf{O}_{\mathcal{B}}$ is placed at the geometric centre of the AV, and its coordinates are the Cartesian position, while $(\mathbf{x}_{\mathcal{B}}, \mathbf{y}_{\mathcal{B}}, \mathbf{z}_{\mathcal{B}})$ is the canonical basis which fluctuates in accordance with the orientation of the platform. For any other mobile solid S, we attach the frame $\boldsymbol{\mathcal{F}}_S$ to it. For instance, $\boldsymbol{\mathcal{F}}_{\mathrm{AU}_i}$ denotes the frame attached to the i-th AU of the UAV.

3.3.1 Positional information

The vector representation of the space translation, from frame $\boldsymbol{\mathcal{F}}_1$ with origin $\mathbf{O}_1$ to frame $\boldsymbol{\mathcal{F}}_2$ with origin $\mathbf{O}_2$, is denoted as $\mathbf{O}_1\mathbf{O}_2 = \begin{bmatrix} x_{12} & y_{12} & z_{12} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^3$, where each component represents the displacement along a specific direction, *e.g.* $x_{12} = x_{\mathbf{O}_2} - x_{\mathbf{O}_1}$. Basic vector operations can be performed, as long as the points are expressed in the same frame, and the vector can be written in any preferred basis. For example, the Cartesian position of a quadrotor can be written as $\mathbf{r} = \begin{bmatrix} x & y & z \end{bmatrix}^\mathsf{T} \in \mathbb{R}^3$. The transposition operator $[\bullet]^\mathsf{T}$, which reflects the elements of any $\mathbf{A} \in \mathbb{R}^{n\times m}$ matrix along its main diagonal is often used in writing vectors, as it allows for column vectors to be written in rows, as shown above.

3.3.2 Orientation representations in space

While expressing position in the world space is straightforward, it is not so for orientation. Indeed, space rotations constitute a Lie group, called the special orthogonal group $\mathrm{SO}(3)$. Therefore, this non-euclidean group requires further definitions and operations. For a comprehensive review on how to parameterise the attitude of an object in three-dimensional space, please refer to the unified reference of [Diebel, 2006], and the references therein. Hereafter, we mention important properties of the elements of $\mathrm{SO}(3)$, and detail some possible representations.

3.3.2.1 Rotation matrices

The fundamental and complete expression of a spatial rotation is that of a rotation matrix, denoted as $\mathbf{R} \in \mathrm{SO}(3)$. $\mathbf{R}$ is 3×3 matrix, and its columns are the coordinates of the unit vectors of the canonical basis that it represents, as depicted in Fig. 3.1.

Furthermore, the rotation from $\boldsymbol{\mathcal{F}}_1$ to $\boldsymbol{\mathcal{F}}_2$ is noted ${}^2\mathbf{R}_1$. For a point in space with coordinates ${}^1\mathbf{x}$ expressed in the basis of $\boldsymbol{\mathcal{F}}_1$, ${}^2\mathbf{R}_1$ can be used to express the coordinates over the old basis

$$\mathbf{R} = \begin{bmatrix} \mathrm{R}_{11} & \mathrm{R}_{12} & \mathrm{R}_{13} \\ \mathrm{R}_{21} & \mathrm{R}_{22} & \mathrm{R}_{23} \\ \mathrm{R}_{31} & \mathrm{R}_{32} & \mathrm{R}_{33} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_x & \mathbf{R}_y & \mathbf{R}_z \end{bmatrix} =$$

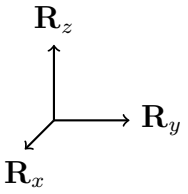

Figure 3.1 – Visualisation of the components of a rotation matrix $\mathbf{R}$, composed of $\mathbf{R}_x$, $\mathbf{R}_y$ and $\mathbf{R}_z$.

in term of the coordinates over the new basis of $\boldsymbol{\mathcal{F}}_2$. As such, the change of basis formula is ${}^2\mathbf{x} = {}^2\mathbf{R}_1 \cdot {}^1\mathbf{x}$, where $\cdot$ denotes the standard matrix multiplication.

For $\mathbf{R}$ to be a valid rotation and lie in $\mathrm{SO}(3)$, it must be orthonormal and consist of proper rotation, *i.e.* it verifies respectively

$$\begin{cases} \mathbf{R}^\mathsf{T} \cdot \mathbf{R} = \mathbf{R} \cdot \mathbf{R}^\mathsf{T} = \mathbf{I}_3 \\ \det(\mathbf{R}) = \det(\mathbf{R}^\mathsf{T}) = 1 \end{cases} . \tag{3.1}$$

Since rotation matrices are orthonormal, the inverse is equal to the transpose, $\mathbf{R}^{-1} = \mathbf{R}^\mathsf{T}$. Moreover, the inverse of a rotation matrix corresponds to the opposite rotation, which can be written

$${}^2\mathbf{R}_1^{-1} = {}^2\mathbf{R}_1^\mathsf{T} = {}^1\mathbf{R}_2. \tag{3.2}$$

Rotation matrices are linear applications, thus they can be composed via standard multiplication in order to obtain new frame transformation, *e.g.*

$${}^3\mathbf{R}_1 = {}^3\mathbf{R}_2 \cdot {}^2\mathbf{R}_1. \tag{3.3}$$

Consequently, composing a rotation matrix with its inverse leads to the neutral element of the $\mathrm{SO}(3)$ group, that is, the 3×3 identity matrix $\mathbf{I}_3 = {}^2\mathbf{R}_1 \cdot {}^1\mathbf{R}_2 = {}^2\mathbf{R}_2$.

From the definitions and properties above, we can note that rotation matrices preserve lengths and angles between vectors, hence they are good candidates to denote the orientation of rigid bodies of frame $\boldsymbol{\mathcal{F}}_\mathcal{B}$ in the world inertial frame $\boldsymbol{\mathcal{F}}_\mathcal{W}$. As robotics studies the dynamics of objects, the time derivative of a $\mathbf{R}$ is also of interest. With the knowledge of the angular rates $\boldsymbol{\omega} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^\mathsf{T} \in \mathbb{R}^3$ of an object w.r.t. the world frame $\boldsymbol{\mathcal{F}}_\mathcal{W}$, the derivative can be written

$$\dot{\mathbf{R}} = \mathbf{R} \cdot \left[{}^\mathcal{B}\boldsymbol{\omega} \right]_\times = \left[{}^\mathcal{W}\boldsymbol{\omega} \right]_\times \cdot \mathbf{R}, \tag{3.4}$$

where ${}^\mathcal{B}\boldsymbol{\omega}$ is the local expression in $\boldsymbol{\mathcal{F}}_\mathcal{B}$ and ${}^\mathcal{W}\boldsymbol{\omega}$ is the global one, in $\boldsymbol{\mathcal{F}}_\mathcal{W}$. Eq. (3.4) uses the

skew-symmetric cross product operator of a vector, denoted $[\bullet]_\times$ and defined as

$$\forall \mathbf{v} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3,\ [\mathbf{v}]_\times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \in \mathbb{R}^{3\times 3}. \tag{3.5}$$

For the sake of completeness, we also denote $[\bullet]^\vee$ as the operator that maps back a skew-symmetric matrix to a vector, defined as

$$\forall \mathbf{M} = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \in \mathbb{R}^{3\times 3},\ [\mathbf{M}]^\vee = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3. \tag{3.6}$$

Skew-symmetric matrices are often used to represent cross products as matrix multiplications. Consider vectors $(\mathbf{v},\, \mathbf{w}) \in (\mathbb{R}^3)^2$, then the cross product between these can be written $\mathbf{v} \times \mathbf{w} = [\mathbf{v}]_\times \cdot \mathbf{w}$.

Nonetheless, the use of rotation matrices is not very convenient to represent the orientation of a system. Indeed, each matrix $\mathbf{R}$ is defined by nine coordinates of which only four are independent, to encode a transformation of three DoFs. Furthermore, to compose two rotations, the product of the two 3×3 corresponding matrices needs to be computed, thus requiring twenty-seven multiplications and eighteen additions. Consequently, more concise orientation representations have been proposed.

3.3.2.2 Euler angles

Euler angles is one of the most widely used attitude representation of a rigid body w.r.t. a fixed coordinate system, to which frames $\boldsymbol{\mathcal{F}}_\mathcal{B}$ and $\boldsymbol{\mathcal{F}}_\mathcal{W}$ are attached respectively. It consists of describing the angle of rotation as a series of three sequential rotations, each one around an axis of the base frame. Since rotations are not commutative, the order in which they are performed matters. Many conventions exist for Euler angles, among which we have selected the Z-Y-X or 321 sequence, since it is a common representation of orientation in the aeronautical field. It consists of first rotating about $(\mathbf{O}_\mathcal{B},\, \mathbf{z}_\mathcal{W})$ by the yaw angle ψ, then rotate about the intermediate axis $(\mathbf{O}_\mathcal{B},\, \mathbf{y})$ by the pitch angle θ, and finally rotate about $(\mathbf{O}_\mathcal{B},\, \mathbf{x}_\mathcal{B})$ by the roll angle φ. The

Euler angles vector is defined as

$$\boldsymbol{\eta} = \begin{bmatrix} \varphi \\ \theta \\ \psi \end{bmatrix} \mid (\varphi,\, \theta,\, \psi) \in \times\,]-\pi,\, \pi] \times \left]-\frac{\pi}{2},\, \frac{\pi}{2}\right[\times\,]-\pi,\, \pi]\,. \tag{3.7}$$

This representation is very compact since $\boldsymbol{\eta} \in \mathbb{R}^3$ requires only three angles to encode all spatial rotations. It can be used to define the elementary rotation matrices about each axis, and their combination yields a rotation matrix of $\mathrm{SO}(3)$. Let the matrices $\mathbf{R}_1$, $\mathbf{R}_2$ and $\mathbf{R}_3$ be respectively the elementary rotation matrices about the x, y and z axes, defined as

$$\begin{cases} \mathbf{R}_1\left(\varphi\right) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{bmatrix} \in \mathrm{SO}(3) \\ \mathbf{R}_2\left(\theta\right) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \in \mathrm{SO}(3) \\ \mathbf{R}_3\left(\psi\right) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathrm{SO}(3) \end{cases}, \tag{3.8}$$

then carrying out the matrix multiplication in the adequate order leaves us with the 321 Euler angles sequence $\mathbf{R}_1\left(\varphi\right) \cdot \mathbf{R}_2\left(\theta\right) \cdot \mathbf{R}_3\left(\psi\right) = \mathbf{R}_{321}(\varphi,\, \theta,\, \psi) = {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}$, that yields

$${}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}} = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\varphi\sin\theta\cos\psi - \cos\varphi\sin\psi & \sin\varphi\sin\theta\sin\psi + \cos\varphi\cos\psi & \sin\varphi\cos\theta \\ \cos\varphi\sin\theta\cos\psi + \sin\varphi\sin\psi & \cos\varphi\sin\theta\sin\psi - \sin\varphi\cos\psi & \cos\varphi\cos\theta \end{bmatrix}. \tag{3.9}$$

The above transformation matrix can now be applied to vectors expressed w.r.t. the inertial frame $\boldsymbol{\mathcal{F}}_{\mathcal{W}}$, to obtain their expressions in the body frame $\boldsymbol{\mathcal{F}}_{\mathcal{B}}$. To get the inverse transformation, going from the body frame to the inertial frame, simply transpose ${}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}$, as it is orthonormal. To obtain the Euler angles vector from a given orientation matrix, one can use the reverse mapping

(see Fig. 3.1 for the matrix terms R_{ij} definition)

$$\boldsymbol{\eta} = \begin{bmatrix} \varphi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan\left(\dfrac{\mathrm{R}_{23}}{\mathrm{R}_{33}}\right) \\ \arcsin\left(\mathrm{R}_{13}\right) \\ \arctan\left(\dfrac{\mathrm{R}_{12}}{\mathrm{R}_{11}}\right) \end{bmatrix}. \tag{3.10}$$

The relationship between Euler angles rates and local body axis rates is given by the set of non-linear Ordinary Differential Equations (ODEs)

$$\dot{\boldsymbol{\eta}} = \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & \dfrac{\sin\varphi}{\cos\theta} & \dfrac{\cos\varphi}{\cos\theta} \\ 0 & \cos\varphi & -\sin\varphi \\ 1 & \sin\varphi\tan\theta & \cos\varphi\tan\theta \end{bmatrix} \cdot {}^{\mathcal{B}}\boldsymbol{\omega} = \mathbf{E}_{\boldsymbol{\eta}} \cdot {}^{\mathcal{B}}\boldsymbol{\omega}. \tag{3.11}$$

Nonetheless, the Euler angles representation suffers from drawbacks. First, there is a large number of different possible sequences, and because of this, practical applications that rely on Euler angles must be consistent, or else non-trivial problems can appear. Secondly, the angles are defined on finite intervals, hence discontinuities arise at π for roll and yaw angles. Lastly, the Euler angles are subject to another singularity, commonly known as the *gimbal lock*. A gimbal lock refers to the loss of a DoF that happens when two axes of rotation align during the three successive transformations, *i.e.* when the pitch angle $\theta \equiv \frac{\pi}{2} \mod 2\pi$. Such angles are not properly defined, as observed in Eqs. (3.11) and (3.10).

As this representation is limited by its singularities, other representations have been used to prevent these inconveniences.

3.3.2.3 Axis-angle representation

Another common representation is the axis-angle, which provides the advantages of being minimal while avoiding singularities. It consists of the *Euler vector* $\mathbf{r}_{\text{a-a}} = \theta_{\text{a}}\mathbf{u}_{\text{a}}$, where $\mathbf{u}_{\text{a}} \in \mathbb{R}^3$ is a unit vector used to describe the axis around which to rotate, while $\theta_{\text{a}} \in [0,\, \pi[$ is the magnitude of the rotation around the axis, in radians. This parameterisation is tightly linked to rotation matrices: the latter are members of the $\mathrm{SO}(3)$ group, while the former are an expression of the $\mathfrak{so}(3)$ Lie algebra. The relationship between the two can be expressed in terms of exponential

and logarithmic maps. The logarithmic map is defined as

$$\begin{aligned} &\log\colon \mathrm{SO}(3) \to \mathfrak{so}(3) \\ &\mathbf{R} \mapsto \mathbf{r}_{\text{a-a}} = \theta_{\text{a}}\mathbf{u}_{\text{a}} \mid \begin{cases} \theta_{\text{a}} = \arccos\left(\dfrac{\mathrm{Tr}(\mathbf{R})-1}{2}\right) \\ \mathbf{u}_{\text{a}} = \dfrac{1}{2\sin\theta_{\text{a}}}\begin{bmatrix} \mathrm{R}_{32}-\mathrm{R}_{23} \\ \mathrm{R}_{13}-\mathrm{R}_{31} \\ \mathrm{R}_{21}-\mathrm{R}_{12} \end{bmatrix} \end{cases} , \end{aligned} \tag{3.12}$$

where $\mathrm{Tr}(\bullet)$ is the trace operator. The exponential map is the following Taylor expansion:

$$\begin{aligned} &\exp\colon \mathfrak{so}(3) \to \mathrm{SO}(3) \\ &\mathbf{r}_{\text{a-a}} \mapsto \mathbf{R} = \sum_{i=0}^{+\infty} \frac{[\theta_{\text{a}}\mathbf{u}_{\text{a}}]_{\times}^{i}}{i!} , \end{aligned} \tag{3.13}$$

which also has a closed-form solution, using *Rodrigues' rotation formula*:

$$\mathbf{R}(\theta_{\text{a}}, \mathbf{u}_{\text{a}}) = \mathbf{I}_3 + \sin\theta_{\text{a}}\,[\mathbf{u}_{\text{a}}]_{\times} + (1-\cos\theta_{\text{a}})\,[\mathbf{u}_{\text{a}}]_{\times}^{2} . \tag{3.14}$$

This representation is very intuitive and minimal since it requires four real numbers to encode all rotations, namely three coordinates and one angle. It is preferred compared to Euler angles as it avoids singularities.

3.3.2.4 Unit quaternions

In mathematics, quaternions are an extension of complex numbers to the three-dimensional space. They are parameterised as $\boldsymbol{\rho} = \rho_w + \rho_x i + \rho_y j + \rho_z k \in \mathbb{Q}$ where $(\rho_w,\ \rho_x,\ \rho_y,\ \rho_z) \in \mathbb{R}^4$ and i, j and k are the *basic quaternions* such that $i^2 = j^2 = k^2 = ijk = -1$. The quaternion space $\mathbb{Q}$ is homeomorphic to $\mathbb{R}^4$ as the complex plane $\mathbb{C}$ is homeomorphic to $\mathbb{R}^2$. For a thorough study on these hyper-complex numbers, please refer to [Sola, 2017] and the references therein. Anyhow, here we present some important definitions and properties of quaternions. A quaternion $\boldsymbol{\rho}$ can

be conveniently written in the vector form

$$\boldsymbol{\rho} = \begin{bmatrix} \rho_w \\ \rho_x \\ \rho_y \\ \rho_z \end{bmatrix} \in \mathbb{R}^4. \tag{3.15}$$

Alternative conventions exist to write quaternions, for instance when placing the scalar ρ_w as the last component of $\boldsymbol{\rho}$. Hereafter, all the formula that we present use the convention from Eq. (3.15).

The *Hamilton product* $\otimes$ is defined on $\mathbb{Q}$ as the multiplicative operator, and is such that

$$\boldsymbol{\rho}_1 \otimes \boldsymbol{\rho}_2 = \begin{bmatrix} \rho_{w_1}\rho_{w_2} - \rho_{x_1}\rho_{x_2} - \rho_{y_1}\rho_{y_2} - \rho_{z_1}\rho_{z_2} \\ \rho_{w_1}\rho_{x_2} + \rho_{x_1}\rho_{w_2} + \rho_{y_1}\rho_{z_2} - \rho_{z_1}\rho_{y_2} \\ \rho_{w_1}\rho_{y_2} - \rho_{x_1}\rho_{z_2} - \rho_{y_1}\rho_{w_2} + \rho_{z_1}\rho_{w_2} \\ \rho_{w_1}\rho_{z_2} + \rho_{x_1}\rho_{y_2} - \rho_{y_1}\rho_{x_2} + \rho_{z_1}\rho_{w_2} \end{bmatrix}. \tag{3.16}$$

From the Hamilton product of Eq. (3.16) it can be derived that

$$\|\boldsymbol{\rho}_1 \otimes \boldsymbol{\rho}_2\| = \|\boldsymbol{\rho}_1\|\|\boldsymbol{\rho}_2\|. \tag{3.17}$$

Unit quaternions, such that $\|\boldsymbol{\rho}\| = 1$, provide a mathematical notation for representing spatial orientations and rotations of elements in the three-dimensional space. More precisely, they encode information about an axis-angle rotation around an arbitrary axis. Nowadays, they are greatly used for many applications such as computer graphics, computer vision, robotics, flight dynamics, and so forth. When used to represent rotation, unit quaternions are also commonly called as rotation quaternions. Starting from the earlier defined Euler vector $\mathbf{r}_{\text{a-a}} = \theta_{\text{a}}\mathbf{u}_{\text{a}}$ of a rotation, the associated rotation quaternion is defined as

$$\boldsymbol{\rho}(\mathbf{r}_{\text{a-a}}) = \begin{bmatrix} \cos\dfrac{\theta_{\text{a}}}{2} \\ \mathbf{u}_{\text{a}} \sin\dfrac{\theta_{\text{a}}}{2} \end{bmatrix} \in \mathbb{S}, \tag{3.18}$$

which satisfies $\|\boldsymbol{\rho}\| = 1$. $\mathbb{S} = \{\boldsymbol{\rho} \in \mathbb{Q} \mid \|\boldsymbol{\rho}\| = 1\}$ is the *unit quaternion sphere* of $\mathbb{Q}$. One can obtain an orientation matrix from a unit quaternion $\boldsymbol{\rho} = \begin{bmatrix} \rho_w & \rho_x & \rho_y & \rho_z \end{bmatrix}^{\mathsf{T}} \in \mathbb{S}$ with the

mapping

$$\mathbf{R}(\boldsymbol{\rho}) = \begin{bmatrix} 2(\rho_w^2 + \rho_x^2) - 1 & 2(\rho_x\rho_y - \rho_w\rho_z) & 2(\rho_x\rho_z + \rho_w\rho_y) \\ 2(\rho_x\rho_y + \rho_w\rho_z) & 2(\rho_w^2 + \rho_y^2) - 1 & 2(\rho_y\rho_z - \rho_w\rho_x) \\ 2(\rho_x\rho_z - \rho_w\rho_y) & 2(\rho_y\rho_z + \rho_w\rho_x) & 2(\rho_w^2 + \rho_z^2) - 1 \end{bmatrix} \in \mathrm{SO}(3). \tag{3.19}$$

Combining the Rodrigues' rotation formula of Eq. (3.14), the Hamilton product of Eq. (3.16) and the rotation quaternion definition of Eq. (3.18), it can be shown that the rotation action of a unit quaternion $\boldsymbol{\rho}$ on a given vector $\mathbf{v}$ is:

$$\forall \mathbf{v} \in \mathbb{R}^3, \ \begin{bmatrix} 0 \\ \mathbf{R}\mathbf{v} \end{bmatrix} = \boldsymbol{\rho} \otimes \begin{bmatrix} 0 \\ \mathbf{v} \end{bmatrix} \otimes \boldsymbol{\rho}^* \in \mathbb{S}, \tag{3.20}$$

where $\mathbf{R}$ and $\boldsymbol{\rho}$ are respectively the rotation matrix and unit quaternion that encode the same rotation, see Eq. (3.19), and $\boldsymbol{\rho}^*$ is the conjugate quaternion of $\boldsymbol{\rho}$, defined as

$$\boldsymbol{\rho}^* = \begin{bmatrix} \rho_w \\ -\rho_x \\ -\rho_y \\ -\rho_z \end{bmatrix} \in \mathbb{S}. \tag{3.21}$$

Unit quaternions inverse and conjugate are equal, since the inverse is defined as $\boldsymbol{\rho}^{-1} = \dfrac{\boldsymbol{\rho}^*}{\|\boldsymbol{\rho}\|^2}$. Therefore in the literature, $\boldsymbol{\rho}^*$ is sometimes replaced by $\boldsymbol{\rho}^{-1}$ in Eq. (3.20).

From Eq. (3.18) it can be observed that unit quaternions perform a double coverage of the rotation set, as illustrated in Fig. 3.2. This implies that the unit quaternions $\boldsymbol{\rho} \in \mathbb{S}$ and $-\boldsymbol{\rho} \in \mathbb{S}$ encode the same spatial rotation, which can be established easily with standard trigonometric formulas. This, of course, has an impact on the proper definitions of quaternion metrics.

Eq. (3.17) shows that $\mathbb{S}$ is stable through the Hamilton product, which acts as a composition for rotation quaternions. Starting from Eq. (3.20), let $\mathbf{R}_1$ and $\mathbf{R}_2$ be the matrices that represent two space rotations, and let respectively $\boldsymbol{\rho}_1$ and $\boldsymbol{\rho}_2$ be the associate rotation quaternions, it can be shown that they verify

$$\forall \mathbf{v} \in \mathbb{R}^3, \ \begin{bmatrix} 0 \\ \mathbf{R}_1\mathbf{R}_2\mathbf{v} \end{bmatrix} = \boldsymbol{\rho}_1 \otimes \boldsymbol{\rho}_2 \otimes \begin{bmatrix} 0 \\ \mathbf{v} \end{bmatrix} \otimes \boldsymbol{\rho}_2^* \otimes \boldsymbol{\rho}_1^* \in \mathbb{S}. \tag{3.22}$$

Rotation quaternions provide a minimal singularity-free representation of orientations. As they

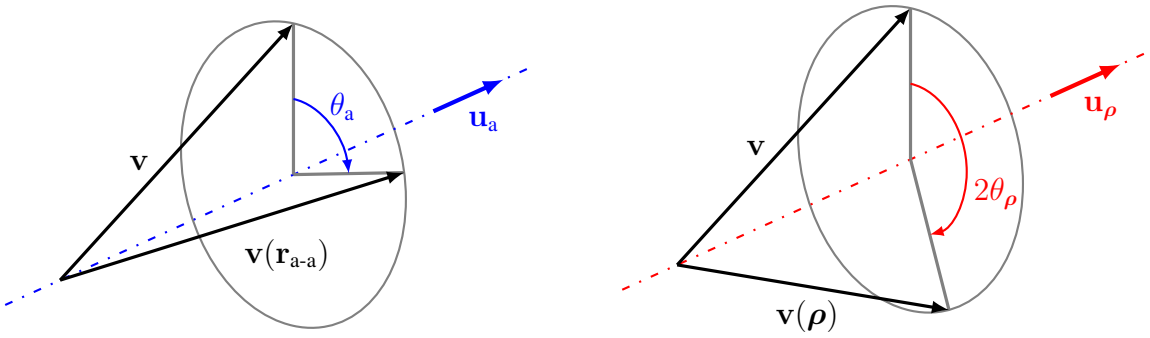

Figure 3.2 – Left: rotation of a vector $\mathbf{v}$ with the Euler vector $\mathbf{r}_{a\text{-}a} = \theta_a \mathbf{u}_a$ (in blue). Right: rotation of vector $\mathbf{v}$ with a unit quaternion $\boldsymbol{\rho}$ of internal angle $\theta_\rho = \theta_a$ (in red).

do not require to be transformed into rotation matrices to perform the rotation action, they are convenient.

The derivative of a rotation quaternion can be expressed in terms of the locally or globally defined body axis rates, giving the set of ODEs

$$\dot{\boldsymbol{\rho}} = \frac{1}{2}\boldsymbol{\rho} \otimes \begin{bmatrix} 0 \\ {}^{\mathcal{B}}\boldsymbol{\omega} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 \\ {}^{\mathcal{W}}\boldsymbol{\omega} \end{bmatrix} \otimes \boldsymbol{\rho}. \tag{3.23}$$

The compactness of quaternions, which needs four components to encode all rotations, makes them easier to handle compared to rotation matrices. Therefore, controllers based on quaternions have been recently designed for UAVs [Carino et al., 2015; Fresk et al., 2013]. Nevertheless, a common issue regarding unit quaternions is that their numerical integration using Eq. (3.23) tends to not necessarily remain inside of $\mathbb{S}$. This issue has been tackled in [Rucker, 2018], which proposes a mapping from $\mathbb{Q}$ to $\mathbb{S}$, such that the quaternions are stable numerically.

The aforementioned orientation representations are commonly used in the literature. Throughout this manuscript, we mostly encode spatial rotations with quaternions, which may be converted to rotation matrices or Euler angles when needed, the latter conversion being mostly used to compute the yaw of the UAV.

3.4 Standard 3D quadrotor model

In this section, we provide a standard dynamical model of a space quadrotor, the orientation of which is represented by unit quaternions of $\mathbb{S}$, as this is a compact, singularity-free representation. This model will be further used as a basis, and modified if necessary.

3.4.1 Definitions

Let $\mathcal{F}_\mathcal{W} = \{\mathbf{O}_\mathcal{W}, (\mathbf{x}_\mathcal{W}, \mathbf{y}_\mathcal{W}, \mathbf{z}_\mathcal{W})\}$ be the world inertial frame, with $\mathbf{O}_\mathcal{W}$ its origin and $(\mathbf{x}_\mathcal{W}, \mathbf{y}_\mathcal{W}, \mathbf{z}_\mathcal{W})$ the canonical basis. The body frame attached to the quadrotor UAV is denoted $\mathcal{F}_\mathcal{B} = \{\mathbf{O}_\mathcal{B}, (\mathbf{x}_\mathcal{B}, \mathbf{y}_\mathcal{B}, \mathbf{z}_\mathcal{B})\}$, and $\mathbf{z}_\mathcal{B}$ is aligned with the thrust of the four AUs. As depicted in Fig. 3.3, $\mathbf{O}_\mathcal{B}$ is placed at the geometric centre of the AV, and is coincident with the CoM. The state vector $\mathbf{q}$ consists of its linear position $\mathbf{r} = \mathbf{O}_\mathcal{W}\mathbf{O}_\mathcal{B} = \begin{bmatrix} x & y & z \end{bmatrix}^\mathsf{T} \in \mathbb{R}^3$ and velocity $\mathbf{v} = \dot{\mathbf{r}} = \begin{bmatrix} \mathrm{v}_x & \mathrm{v}_y & \mathrm{v}_z \end{bmatrix}^\mathsf{T} \in \mathbb{R}^3$, both expressed in the world frame $\mathcal{F}_\mathcal{W}$. The body orientation is expressed through the unit quaternion $\boldsymbol{\rho} = \begin{bmatrix} \rho_w & \rho_x & \rho_y & \rho_z \end{bmatrix}^\mathsf{T} \in \mathbb{S}$, which can also be mapped to the orientation matrix $\mathbf{R}(\boldsymbol{\rho}) \in \mathrm{SO}(3)$, see Eq. (3.19), such that, *e.g.*, $\mathbf{z}_\mathcal{B} = \mathbf{R}(\boldsymbol{\rho}) \cdot \mathbf{z}_\mathcal{W}$. Lastly, the angular velocity is denoted as $\boldsymbol{\omega} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^\mathsf{T} \in \mathbb{R}^3$, and is always expressed locally in the body frame $\mathcal{F}_\mathcal{B}$. Therefore, we take $\mathbf{q} = \begin{bmatrix} \mathbf{r}^\mathsf{T} & \mathbf{v}^\mathsf{T} & \boldsymbol{\rho}^\mathsf{T} & \boldsymbol{\omega}^\mathsf{T} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{S} \times \mathbb{R}^3$ as the state vector.

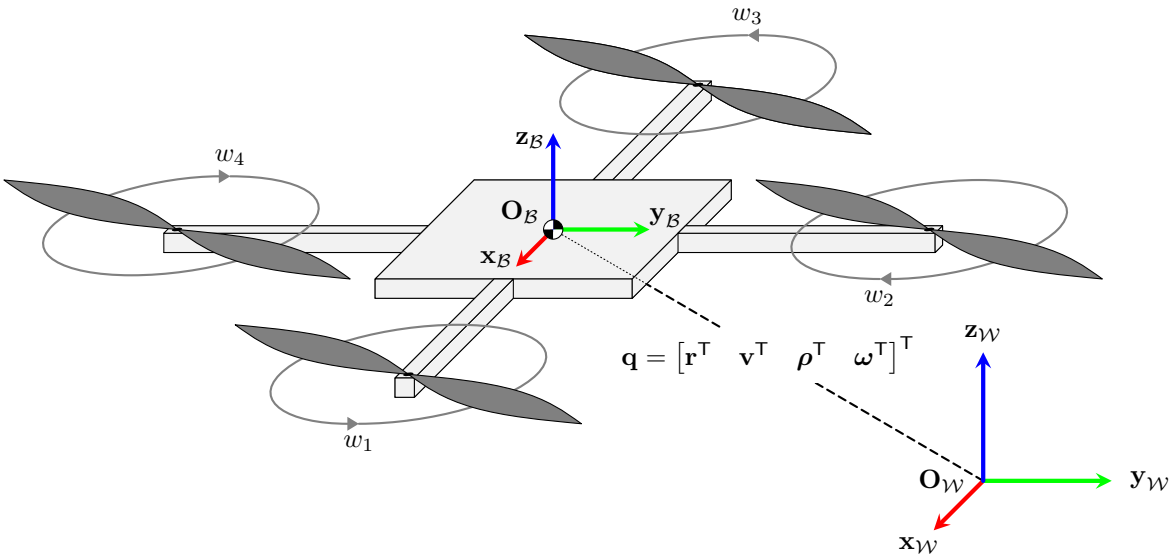

Figure 3.3 – Standard quadrotor model, oriented with unit quaternion. Notice that the odd numbered AUs have their propellers spinning CCW while the even number ones have their propellers spinning CW.

As described in Subsection 2.2.1, each AU of the UAV is able to exert aerodynamic thrust through the conversion of electrical energy into mechanical rotational energy of the propeller, whose geometry makes it possible to generate a pressure gradient in the air. The first AU is on the positive $\mathbf{x}_\mathcal{B}$-axis, the second on the positive $\mathbf{y}_\mathcal{B}$-axis, the third on the negative $\mathbf{x}_\mathcal{B}$-axis, and the fourth on the negative $\mathbf{y}_\mathcal{B}$-axis. Let f be the total aerodynamic thrust, carried by $\mathbf{z}_\mathcal{B}$, and let $\boldsymbol{\tau} = \begin{bmatrix} \tau_x & \tau_y & \tau_z \end{bmatrix}^\mathsf{T} \in \mathbb{R}^3$ be the total torque vector, expressed in the body frame $\mathcal{F}_\mathcal{B}$. The

fictitious inputs, namely the total external thrust and torque $\begin{bmatrix} f & \boldsymbol{\tau}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^4$ applied to the AV, can be linked to the *real* input vector $\mathbf{u} = \begin{bmatrix} w_1^2 & w_2^2 & w_3^2 & w_4^2 \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^4$, that contains the squared versions of the propeller angular velocities. These are related by the well-known input mapping matrix $\mathbf{S}$, such that

$$\begin{bmatrix} f \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = k_f \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & \ell & 0 & -\ell \\ -\ell & 0 & \ell & 0 \\ k_\tau & -k_\tau & k_\tau & -k_\tau \end{bmatrix} \begin{bmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ w_4^2 \end{bmatrix} = \mathbf{S}\mathbf{u}, \tag{3.24}$$

where k_f, k_τ are, in first approximation, calibration parameters that depend on the propeller characteristics, namely the thrust and torque aerodynamic coefficients, and ℓ is half the distance between two opposite AUs. We take $\mathbf{u} = \begin{bmatrix} w_1^2 & w_2^2 & w_3^2 & w_4^2 \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^4$ as the space quadrotor control inputs throughout the manuscript.

3.4.2 Quadrotor dynamics

Let m be the mass of the UAV and $\mathbf{J} = \mathbf{J}_{\mathbf{O}_\mathcal{B}}$ its inertia tensor. As the quadrotor has two planes of symmetry, that are $(\mathbf{O}_\mathcal{B}, \mathbf{x}_\mathcal{B}, \mathbf{z}_\mathcal{B})$ and $(\mathbf{O}_\mathcal{B}, \mathbf{y}_\mathcal{B}, \mathbf{z}_\mathcal{B})$, the inertia, defined in the body frame $\boldsymbol{\mathcal{F}}_\mathcal{B}$ as

$$\mathbf{J}_{\mathbf{O}_\mathcal{B}} = \begin{bmatrix} \mathrm{J}_{xx} & 0 & 0 \\ 0 & \mathrm{J}_{yy} & 0 \\ 0 & 0 & \mathrm{J}_{zz} \end{bmatrix} \in \mathbb{R}^{3\times 3}, \tag{3.25}$$

is diagonal and also minimal, since in this model, the CoM is coincident to the geometric centre $\mathbf{O}_\mathcal{B}$ of the platform.

The external forces applied to the body of the quadrotor are its weight plus the total acting thrust and torques, that are generated by the propellers. Therefore, we can write the Newton-Euler equations of motion at $\mathbf{O}_\mathcal{B}$, in the matrix form

$$\begin{bmatrix} \mathbf{R}(\boldsymbol{\rho}) \cdot f(\mathbf{u})\mathbf{z}_\mathcal{W} - mg\mathbf{z}_\mathcal{W} \\ \boldsymbol{\tau}(\mathbf{u}) \end{bmatrix} = \begin{bmatrix} m\mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{J} \end{bmatrix} \begin{bmatrix} \boldsymbol{\gamma}(\mathrm{t}) \\ \boldsymbol{\alpha}(\mathrm{t}) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ [\boldsymbol{\omega}]_\times \cdot \mathbf{J} \cdot \boldsymbol{\omega} \end{bmatrix}, \tag{3.26}$$

where $\boldsymbol{\gamma}(\mathrm{t})$ is the body acceleration expressed in $\boldsymbol{\mathcal{F}}_\mathcal{W}$ and $\boldsymbol{\alpha}(\mathrm{t})$ is the angular acceleration, in $\boldsymbol{\mathcal{F}}_\mathcal{B}$. Here, g is the Earth gravity acceleration constant. The orientation matrix $\mathbf{R}(\boldsymbol{\rho}) = {}^\mathcal{B}\mathbf{R}_\mathcal{W}$ is expressed through the unit quaternion $\boldsymbol{\rho} \in \mathbb{S}$, see Eq. (3.19). Now, Eq. (3.26) can be rewritten

to express the linear acceleration $\boldsymbol{\gamma}$ and the angular acceleration $\boldsymbol{\alpha}$, which yields

$$\begin{bmatrix} \boldsymbol{\gamma}(\mathrm{t}) \\ \boldsymbol{\alpha}(\mathrm{t}) \end{bmatrix} = \begin{bmatrix} \dfrac{f(\mathbf{u})}{m}\mathbf{R}(\boldsymbol{\rho}) \cdot \mathbf{z}_{\mathcal{W}} - g\mathbf{z}_{\mathcal{W}} \\ \mathbf{J}^{-1} \cdot \Big(\boldsymbol{\tau}(\mathbf{u}) - [\boldsymbol{\omega}]_{\times} \cdot \mathbf{J} \cdot \boldsymbol{\omega}\Big) \end{bmatrix}. \tag{3.27}$$

From Eq. (3.27), it is now possible to write the quadrotor complete state-space model

$$\dot{\mathbf{q}}(\mathrm{t}) = \begin{cases} \dot{\mathbf{r}}(\mathrm{t}) = \mathbf{v}(\mathrm{t}) \\ \dot{\mathbf{v}}(\mathrm{t}) = \dfrac{f(\mathbf{u})}{m}\mathbf{R} \cdot \mathbf{z}_{\mathcal{W}} - g\mathbf{z}_{\mathcal{W}} \\ \dot{\boldsymbol{\rho}}(\mathrm{t}) = \dfrac{1}{2}\,\boldsymbol{\rho} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \\ \dot{\boldsymbol{\omega}}(\mathrm{t}) = \mathbf{J}^{-1} \cdot \Big(\boldsymbol{\tau}(\mathbf{u}) - [\boldsymbol{\omega}]_{\times} \cdot \mathbf{J} \cdot \boldsymbol{\omega}\Big) \end{cases}, \tag{3.28}$$

where $\dot{\mathbf{r}}$ and $\dot{\mathbf{v}} = \boldsymbol{\gamma}(\mathrm{t})$ are expressed in $\boldsymbol{\mathcal{F}}_{\mathcal{W}}$, also $\dot{\boldsymbol{\rho}}$ and $\dot{\boldsymbol{\omega}} = \boldsymbol{\alpha}(\mathrm{t})$ are expressed in $\boldsymbol{\mathcal{F}}_{\mathcal{B}}$. Note that:

— the expression of $\mathbf{R}(\mathrm{t},\, \boldsymbol{\rho}) = {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}$ can be found in Eq. (3.19);

— the expressions of $f(\mathrm{t},\, \mathbf{u})$ and $\boldsymbol{\tau}(\mathrm{t},\, \mathbf{u})$ can be found in Eq. (3.24), which maps these latter to the real input vector $\mathbf{u}$.

Now, let vector $\mathbf{p} = \begin{bmatrix} m & \mathrm{J}_{xx} & \mathrm{J}_{yy} & \mathrm{J}_{zz} & k_f & k_\tau & \ell & g \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^8$ contain the parameters involved in the state-space model, we define the quadrotor dynamics $\mathbf{f}$, such that

$$\dot{\mathbf{q}}(\mathrm{t}) = \mathbf{f}(\mathbf{q}(\mathrm{t}),\, \mathbf{u}(\mathrm{t}),\, \mathbf{p}). \tag{3.29}$$

The dynamics $\mathbf{f}$ is the function associated to the state-space model defined just above, in Eq. (3.28).

3.5 Summary

In this thesis, we focus on generating trajectories that are robust to possible uncertainties in the model parameters of dynamic systems, such as a quadrotor. In this chapter, we have detailed the main mathematical tools and conventions that are further employed in the manuscript. In particular, we have detailed how to obtain the dynamics of a standard space quadrotor, oriented

with unit quaternions.

In our first contribution, Chapter 4, we limit ourselves to a case study of a planar quadrotor (for ease of computation). Thus, the dynamic model in the plane will be detailed in Subsection 4.5.1. Later in the second contribution, Chapter 5, we extend the level of complexity, and make the dynamic study of the space quadrotor. Therefore, we exploit the model detailed just above, of Section 3.4. In the last contribution, Chapter 6, we derive a new dynamic model of the quadrotor in space, for which we consider that the CoM can be shifted from the geometric centre $\mathbf{O}_{\mathcal{B}}$. This implies a rewriting of the Newton-Euler equations, with an additional coupling between the linear and angular accelerations. This model will be further detailed in Subsection 6.5.1.

Now, we can proceed to the presentation of the main ideas that we support in this thesis, through the principal contributions.

Part II

Uncertainty-aware trajectory planning for unmanned aerial vehicles

CHAPTER 4

ROBUST TRAJECTORY PLANNING WITH PARAMETRIC UNCERTAINTIES

4.1 Introduction

In this first contribution, we propose an extension of the base closed-loop sensitivity framework [Robuffo Giordano et al., 2018], aiming at robust trajectory generation under parametric uncertainties. In this chapter, we introduce the concept of *first-order closed-loop input sensitivity* to that of the state, and show how to exploit it in our TO framework. The method allows to generate optimal reference trajectories that minimise the state and input sensitivities, thus providing intrinsically robust motions plans. We parameterise the reference trajectories with Bézier curves, and also discuss how to consider linear and non-linear constraints in the process (*e.g.*, input saturation). Then, the whole machinery is validated via an extensive statistical campaign that clearly shows the interest of the proposed strategy. The key components of our method are:

— rather than only considering the first-order closed-loop state sensitivity [Robuffo Giordano et al., 2018], we also exploit the closed-loop input sensitivity in the cost objective, and seek to minimise a combination of both indexes, to better ensure the feasibility/safety of the robotic task. We formulate a new weighted optimisation problem with both metrics;

— compared to [Robuffo Giordano et al., 2018] that used plain polynomials, we switched to Bézier curves for representing the reference trajectories, since this class of curves allows for better stability, and presents other interesting properties (*e.g.*, the trajectory remains within the complex envelope of the control points);

— we added the consideration of input saturations as non-linear constraints. It is indeed essential to ensure the feasibility of the task;

— the soundness of the approach is assessed through an extensive campaign of *perturbed* simulation, for many output targets.

This chapter is structured as follows: after defining the generic problem in Section 4.2, in

Section 4.3 we derive the closed-loop sensitivity metrics w.r.t. parameters. Then, in Section 4.4, we explain how one can minimise these metrics by defining several linear and non-linear constrained optimisation problems. Our framework is tested for a planar quadrotor equipped with a DFL controller (Section 4.5) in an extensive campaign of perturbed simulations, in Section 4.6: the analysis of the results gives a validation of the improvements in closed-loop performance, when minimising the sensitivities along the trajectories. Section 4.7 concludes the contribution, and opens to further perspectives.

This chapter was published in the research community, and then presented at ICRA 2021:

Pascal Brault, Quentin Delamare, et al. [2021], « Robust Trajectory Planning with Parametric Uncertainties », *in*: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11095–11101

4.2 Problem definition

In this section, we provide a generic definition of the problem linked to the closed-loop sensitivities. We first provide the main equations characterising an autonomous system whose performance we want to improve. Then, consider a general non-linear dynamical model for representing a robot behaviour

$$\begin{cases} \mathbf{q}(0) = \mathbf{q}_0 \\ \dot{\mathbf{q}}(\mathrm{t}) = \mathbf{f}(\mathbf{q}(\mathrm{t}),\, \mathbf{u}(\mathrm{t}),\, \mathbf{p}) \end{cases}, \tag{4.1}$$

where $\mathbf{q}(\mathrm{t}) \in \mathbb{R}^{\mathrm{n_q}}$ is the state of the system, $\mathbf{u}(\mathrm{t}) \in \mathbb{R}^{\mathrm{n_u}}$ is the input vector that are fed into the plant, and $\mathbf{p} \in \mathbb{R}^{\mathrm{n_p}}$ is the model parameters vector that directly affect the dynamics $\mathbf{f}$. This parameters vector can include information such as the mass, lengths, actuation properties, and so forth. In this framework, we always assume that $\mathbf{p}$ is constant w.r.t. time, since we only work with systems and parameters for which the assumption seems correct. In fact, for almost all parameters, their values are not constant w.r.t. time, but the variations remain small enough, which allows us not to model them (otherwise the complexity of the models would explode).

Now, let $\mathbf{y}_\mathrm{d}(\mathbf{a},\, \mathrm{t}) \in \mathbb{R}^{\mathrm{n_y}}$, defined on the time interval $\mathrm{t} \in [0,\, \mathrm{T}] = \mathbb{T} \subset \mathbb{R}$, be some desired output trajectory. This reference is tracked by some variables of interest of the system, represented by the output function $\mathbf{y}(\mathbf{q}(\mathrm{t})) \in \mathbb{R}^{\mathrm{n_y}}$, such that $\mathrm{n_y} \leqslant \mathrm{n_q}$. Vector $\mathbf{a} \in \mathbb{R}^{\mathrm{n_a}}$ contains the parameters that shape the reference trajectory, for the chosen class of curves. For instance, plain polynomials were used in [Robuffo Giordano et al., 2018], but other representations are

possible, such as Bézier curves, or even piece-wise curves. Note that for this kind of parameterisation with finite dimension $\mathrm{n}_{\mathbf{a}}$, the represented references form only a subset of all possible trajectories, *i.e.* only the smooth ones, that are infinitely differentiable, also denoted class $\mathcal{C}^{\infty}$.

We assume that the robot is equipped with a control law able to correctly perform the tracking task, *i.e.* it computes the appropriate input $\mathbf{u} \in \mathbb{R}^{\mathrm{n}_{\mathbf{u}}}$ that are fed to the system. For generality we consider possible internal states $\boldsymbol{\xi} \in \mathbb{R}^{\mathrm{n}_{\boldsymbol{\xi}}}$ that may represent, *e.g.*, an integral action, or dynamic extensions. Thereby, we define the control law as

$$\begin{cases} \boldsymbol{\xi}(0) = \boldsymbol{\xi}_0 \\ \dot{\boldsymbol{\xi}}(\mathrm{t}) = \mathbf{g}(\boldsymbol{\xi}(\mathrm{t}),\, \mathbf{q}(\mathrm{t}),\, \mathbf{y}_{\mathrm{d}}(\mathbf{a},\, \mathrm{t}),\, \mathbf{p}_{\mathrm{c}},\, \mathbf{k}_{\mathrm{c}}) \\ \mathbf{u}(\mathrm{t}) = \mathbf{h}(\boldsymbol{\xi}(\mathrm{t}),\, \mathbf{q}(\mathrm{t}),\, \mathbf{y}_{\mathrm{d}}(\mathbf{a},\, \mathrm{t}),\, \mathbf{p}_{\mathrm{c}},\, \mathbf{k}_{\mathrm{c}}) \end{cases} , \tag{4.2}$$

where $\mathbf{k}_{\mathrm{c}} \in \mathbb{R}^{\mathrm{n}_{\mathbf{k}_{\mathrm{c}}}}$ is the controller gains vector and $\mathbf{p}_{\mathrm{c}} \in \mathbb{R}^{\mathrm{n}_{\mathbf{p}}}$ is the *nominal* parameters vector: most of the time, there is little chance that the control loop parameters, *i.e.* $\mathbf{p}_{\mathrm{c}}$, match the 'real' parameters, *i.e.* $\mathbf{p}$, since the accuracy of the system model is limited.

Knowing Eqs. (4.1) and (4.2) and by using the same reasoning as in [Robuffo Giordano et al., 2018], we can now define and compare two different cases which highlight one of the typical issues in robot control:

— on the first hand, in the *nominal case* where $\mathbf{p} = \mathbf{p}_{\mathrm{c}}$, the controller is able to perform the tracking task with utmost accuracy, delivering its best distributions-loop performance, with the smallest error possible $\mathbf{e}(\mathrm{t}) = \mathbf{y}_{\mathrm{d}}(\mathbf{a},\, \mathrm{t}) - \mathbf{y}(\mathbf{q}(\mathrm{t}))$;

— on the other hand, in the *perturbed case* where $\mathbf{p} \neq \mathbf{p}_{\mathrm{c}}$, the dynamics given to the controller $\mathbf{f}(\mathbf{q}(\mathrm{t}),\, \mathbf{u}(\mathrm{t}),\, \mathbf{p}_{\mathrm{c}})$ differs from $\mathbf{f}(\mathbf{q}(\mathrm{t}),\, \mathbf{u}(\mathrm{t}),\, \mathbf{p})$ and the tracking task is done on a system that does not match the reality. With this lack of knowledge, the closed-loop behaviour will perform worse, resulting in a tracking error $\mathbf{e}(\mathrm{t})$ that might be larger than in the previous nominal case.

For optimal navigation efficiency and safety in systems, it is crucial to have a minimal deviation in behaviour between the two scenarios. This can be achieved by generating optimal reference trajectories, the tracking of which is intrinsically robust to variations in the model parameters. By doing so, the robot in ensured to remain reliable and efficient, even in the event of a perturbation, *i.e.* when $\mathbf{p} \neq \mathbf{p}_{\mathrm{c}}$. In the subsequent section, we shall delve into the process of assessing the alterations in the performance of systems, by defining the relevant metrics. In the context of robust TO in particular, we will show how to obtain these quantities along any reference trajectory, as well as their gradients.

4.3 Closed-loop sensitivity metrics

In this section, we explain how to compute the key metrics that quantify the changes in a system's behaviour w.r.t. the variation in its model parameters. The computation of the *closed-loop sensitivities* is required for the optimisation process, and exploited in the cost function. We also show how the associated gradients w.r.t. the optimisation vector $\mathbf{a}$ can be obtained.

4.3.1 Definitions

First, let us define the *closed-loop state sensitivity*

$$\mathbf{\Pi}(\mathrm{t}) = \left.\frac{\partial \mathbf{q}(\mathrm{t})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_\mathrm{c}} \in \mathbb{R}^{\mathrm{n_q} \times \mathrm{n_p}} \tag{4.3}$$

which represents the evolution of the state w.r.t. variations in the parameter vector $\mathbf{p}$, and is evaluated on the nominal value $\mathbf{p} = \mathbf{p}_\mathrm{c}$. This quantity has already been introduced in [Robuffo Giordano et al., 2018], thereby we shortly recall its use for this work. To improve the system behaviour in presence of parameter inaccuracies, one can minimise some norm of $\mathbf{\Pi}(\mathrm{t})$ w.r.t. the optimisation variables $\mathbf{a}$. An optimal shape of the trajectory $\mathbf{y}_\mathrm{d}(\mathbf{a}^\star, \mathrm{t})$ with a minimal state sensitivity would make the closed-loop state evolution $\mathbf{q}(\mathrm{t})$ in the perturbed case as close as possible to its evolution in the nominal case. We illustrate this in Fig. 4.1, for a specific instant of the trajectory $\mathrm{t} \in \mathbb{T}$. Let $\mathrm{q}_i(\mathrm{t}) \in \mathbf{q}(\mathrm{t}) \mid i \in [\![1, \mathrm{n_q}]\!]$ be one of the state variables and $\mathrm{p}_j(\mathrm{t}) \in \mathbf{p}(\mathrm{t}) \mid j \in [\![1, \mathrm{n_p}]\!]$ be one of the parameters, we display the variation in the state $\Delta\mathrm{q}_i(\mathrm{t}) = \mathrm{q}_i(\mathrm{t}) - \mathrm{q}_{\mathrm{c}_i}(\mathrm{t})$ w.r.t. the variation in the parameter $\Delta\mathrm{p}_j = \mathrm{p}_j - \mathrm{p}_{\mathrm{c}_j}$, where $\mathrm{q}_{\mathrm{c}_i}(\mathrm{t})$ is the nominal state, obtained for $\Delta\mathrm{p}_j = 0 \Leftrightarrow \mathrm{p}_j = \mathrm{p}_{\mathrm{c}_j}$. From the optimisation, we expect that the reduction in one component of the state sensitivity $\Pi_{ij}(\mathbf{a}^\star, \mathrm{t}) < \Pi_{ij}(\mathbf{a}, \mathrm{t})$ brings the perturbed state $\mathrm{q}_i(\mathrm{t})$ closer to its nominal $\mathrm{q}_{\mathrm{c}_i}(\mathrm{t})$. Note that this figure only gives a general trend of what is expected. Indeed, minimising the state sensitivity will guarantee a smaller deviation $\Delta\mathrm{q}_i(\mathrm{t})$, only around the nominal case $\mathrm{p}_j = \mathrm{p}_{\mathrm{c}_j}$, thus when $\Delta\mathrm{p}_j$ remains small enough (close to the origin).

In this chapter, we introduce the novel notion of *closed-loop input sensitivity*

$$\mathbf{\Theta}(\mathrm{t}) = \left.\frac{\partial \mathbf{u}(\mathrm{t})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_\mathrm{c}} \in \mathbb{R}^{\mathrm{n_u} \times \mathrm{n_p}} \tag{4.4}$$

which quantifies the amount of variations that would occur on the inputs w.r.t. deviations in the model parameters $\mathbf{p}$, and is also evaluated at the nominal $\mathbf{p} = \mathbf{p}_\mathrm{c}$. We strongly think that

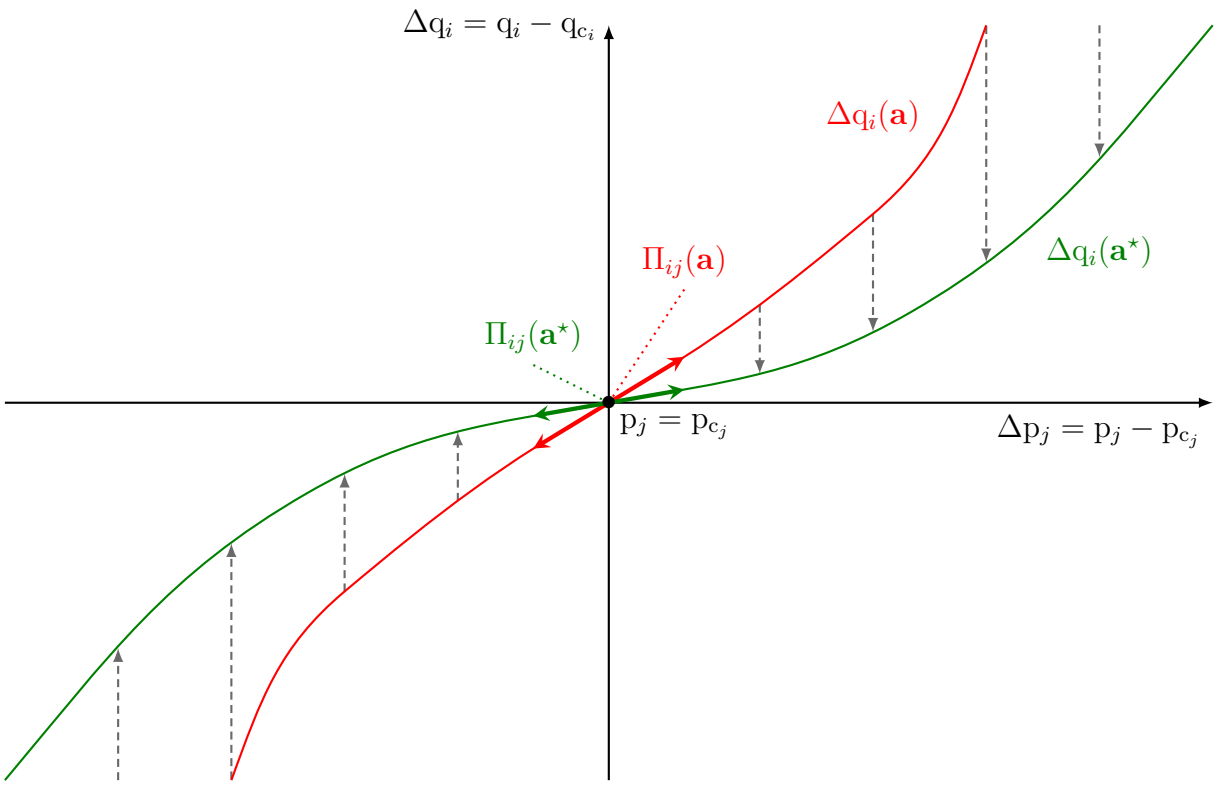

Figure 4.1 – Visual representation of the impact of reducing a component of the state sensitivity $\mathbf{\Pi}(t)$, for all $t \in \mathbb{T}$. On the graphic, we observe the gap $\Delta q_i(t) = q_i(t) - q_{c_i}(t)$ in the i-th component of the states $\mathbf{q}$, w.r.t. the variation in the j-th component of the parameters $\mathbf{p}$, $\Delta p_j = p_j - p_{c_j}$, where $(i, j) \in [\![1, n_{\mathbf{q}}]\!] \times [\![1, n_{\mathbf{p}}]\!]$. We show the gap of behaviour between $\Delta q_i(\mathbf{a})$ before the optimisation, in red, and $\Delta q_i(\mathbf{a}^\star)$ after optimising, in green. At the origin (also at the nominal $p_j = p_{c_j}$), we observe the reduction of the slope, corresponding to the reduction of the state sensitivity, *i.e.* $\Pi_{ij}(\mathbf{a}^\star, t) < \Pi_{ij}(\mathbf{a}, t)$. Note that on the graphic, the time dependencies have been removed to minimise writing.

both state and input sensitivities should me minimised in order to ensure the best performance possible. Illustrating the necessity of the input sensitivity minimisation, consider the example of a mobile robot executing pick-up and drop-off operations in a factory. In the ideal scenario where we have a complete knowledge of the system, represented by the nominal case $\mathbf{p} = \mathbf{p}_c$, it is possible to choose a reference motion that can be physically executed by the actuators. However in the perturbed case, when the parameters deviate from their nominal values, *i.e.* $\mathbf{p} \neq \mathbf{p}_c$, the discrepancy in inputs can lead to significant tracking errors. If the input gap is too large, it becomes impossible for the controller to correct the state, once the actuation limits are reached. Moreover, even if the actuation limits are not reached, the increased tracking errors can result in excessive inputs from the tracking controller over a cycle, leading to overloading of the actuators over time. This could cause damage to the motors, and ultimately compromise the integrity of

the entire system in the long run. By understanding how security could be jeopardised because of an undesired shift in the inputs, we find it very important to plan reference trajectories, also with minimum input sensitivity, so that the inputs become as predictable as possible. Therefore, we will formulate an optimisation problem that makes it possible to minimise both metrics at once, thus ensuring an improvement in performance while guaranteeing the dynamic feasibility. To summarise, the main motivation for considering this metric is that the discrepancy between the control parameters $\mathbf{p}_\mathrm{c}$ and the true ones $\mathbf{p}$ may result in some undesired inputs variation: in any system, actuators are specifically chosen for the desired application, hence they need to be operated as close as possible to the conditions they were designed for.

Now that we have defined the state and input sensitivities $\mathbf{\Pi}(\mathrm{t})$ and $\mathbf{\Theta}(\mathrm{t})$, we show how to obtain them along a reference trajectory $\mathbf{y}_\mathrm{d}(\mathbf{a},\, \mathrm{t})$ by forward integration.

4.3.2 Numerical integration

In the general case, it is not possible to compute a closed-form of $\mathbf{\Pi}(\mathrm{t})$, however it is possible to differentiate Eq. (4.3) over time, which yields

$$\begin{cases} \mathbf{\Pi}(0) = \mathbf{\Pi}_0 = \mathbf{0}_{\mathrm{n_q} \times \mathrm{n_p}} \\ \dot{\mathbf{\Pi}}(\mathrm{t}) = \dfrac{\partial \mathbf{f}}{\partial \mathbf{q}} \cdot \mathbf{\Pi} + \dfrac{\partial \mathbf{f}}{\partial \mathbf{u}} \cdot \mathbf{\Theta} + \dfrac{\partial \mathbf{f}}{\partial \mathbf{p}} \end{cases}, \tag{4.5}$$

where $\mathbf{f}$ is the system dynamics of Eq. (4.1).

Integrating Eq. (4.5) is not obvious because $\mathbf{\Theta}(\mathrm{t})$ is not known *a priori*. However, using the expression of $\mathbf{u}(\mathrm{t})$ in Eq. (4.2), we can rewrite Eq. (4.4) as

$$\mathbf{\Theta}(\mathrm{t}) = \frac{\partial \mathbf{h}}{\partial \mathbf{q}} \cdot \mathbf{\Pi} + \frac{\partial \mathbf{h}}{\partial \boldsymbol{\xi}} \cdot \frac{\partial \boldsymbol{\xi}}{\partial \mathbf{p}}. \tag{4.6}$$

There is still an unknown term, the *internal state sensitivity* that we denote as

$$\mathbf{\Pi}_{\boldsymbol{\xi}}(\mathrm{t}) = \left.\frac{\partial \boldsymbol{\xi}}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_\mathrm{c}} \in \mathbb{R}^{\mathrm{n}_{\boldsymbol{\xi}} \times \mathrm{n}_\mathbf{p}}. \tag{4.7}$$

Derivation of Eq. (4.7) leads to

$$\begin{cases} \mathbf{\Pi_{\xi}}(0) = \mathbf{\Pi_{\xi_0}} = \mathbf{0}_{\mathrm{n_{\xi}} \times \mathrm{n_p}} \\ \dot{\mathbf{\Pi}}_{\boldsymbol{\xi}}(\mathrm{t}) = \dfrac{\partial \mathbf{g}}{\partial \mathbf{q}} \cdot \mathbf{\Pi} + \dfrac{\partial \mathbf{g}}{\partial \boldsymbol{\xi}} \cdot \mathbf{\Pi_{\xi}} \end{cases} . \tag{4.8}$$

Eq. (4.5), Eq. (4.6) and Eq. (4.8) can be regrouped in one single set of differential equations, which allows to compute the state and input sensitivities. For the sake of readability, we introduce the notation $\mathbf{x}_{,\mathbf{y}}$ in order to refer to the jacobian of a vector function $\mathbf{x}$ w.r.t. one of its arguments $\mathbf{y}$. With this shorthand, the set of differential equations becomes

$$\begin{cases} \mathbf{\Pi}(0) = \mathbf{\Pi}_0 = \mathbf{0}_{\mathrm{n_q} \times \mathrm{n_p}} \\ \mathbf{\Pi_{\xi}}(0) = \mathbf{\Pi_{\xi_0}} = \mathbf{0}_{\mathrm{n_{\xi}} \times \mathrm{n_p}} \\ \dot{\mathbf{\Pi}}(\mathrm{t}) = \mathbf{f}_{,\mathbf{q}} \cdot \mathbf{\Pi} + \mathbf{f}_{,\mathbf{u}} \cdot \mathbf{\Theta} + \mathbf{f}_{,\mathbf{p}} \\ \dot{\mathbf{\Pi}}_{\boldsymbol{\xi}}(\mathrm{t}) = \mathbf{g}_{,\mathbf{q}} \cdot \mathbf{\Pi} + \mathbf{g}_{,\boldsymbol{\xi}} \mathbf{\Pi_{\xi}} \\ \mathbf{\Theta}(\mathrm{t}) = \mathbf{h}_{,\mathbf{q}} \cdot \mathbf{\Pi} + \mathbf{h}_{,\boldsymbol{\xi}} \mathbf{\Pi_{\xi}} \end{cases} . \tag{4.9}$$

Thereby, given a reference trajectory $\mathbf{y}_{\mathrm{d}}(\mathbf{a},\, \mathrm{t})$, one can obtain the evolutions of $\mathbf{\Pi}$, $\mathbf{\Pi_{\xi}}$ and $\mathbf{\Theta}$ for the whole time window $\mathbb{T}$. Since the real parameters of the system $\mathbf{p}$ are not known, it is impossible to integrate this set of differential equations to evaluate the sensitivities w.r.t. the true parameters. That being said, one can evaluate these quantities at $\mathbf{p}_{\mathrm{c}}$ instead. $\mathbf{\Pi}(\mathbf{p}_{\mathrm{c}})$, $\mathbf{\Pi_{\xi}}(\mathbf{p}_{\mathrm{c}})$ and $\mathbf{\Theta}(\mathbf{p}_{\mathrm{c}})$ are still close enough from $\mathbf{\Pi}(\mathbf{p})$, $\mathbf{\Pi_{\xi}}(\mathbf{p})$ and $\mathbf{\Theta}(\mathbf{p})$ respectively, as we assume that $\mathbf{p}_{\mathrm{c}}$ is a good approximation of $\mathbf{p}$, and, indeed, the validity of this assumption will be confirmed in the extensive tests of Section 4.6.

4.3.3 Gradient derivation

As in [Robuffo Giordano et al., 2018], we now show how to obtain $\partial\mathbf{\Pi}/\partial\mathbf{a}$ and $\partial\mathbf{\Theta}/\partial\mathbf{a}$ that are respectively, the gradients of $\mathbf{\Pi}$ and $\mathbf{\Theta}$ w.r.t. the optimisation variable $\mathbf{a}$. This subsection is an updated version of the gradients computation, in which $\partial\mathbf{\Theta}/\partial\mathbf{a}$ is also treated. Since $\mathbf{\Pi}$ and $\mathbf{\Theta}$ are matrices, their gradients are tensors: for simplicity, we express each component of the gradient $\partial\mathbf{\Pi}/\partial\mathrm{a}_i$ w.r.t. each individual i-th component of a_i. Let then

$$\mathbf{\Pi}_{\mathrm{a}_i}(\mathrm{t}) = \left.\frac{\partial \mathbf{\Pi}(\mathrm{t})}{\partial \mathrm{a}_i}\right|_{\mathbf{p}=\mathbf{p}_{\mathrm{c}}} \in \mathbb{R}^{\mathrm{n_q} \times \mathrm{n_p}}, \tag{4.10}$$

$$\mathbf{\Pi}_{\boldsymbol{\xi}_{\mathrm{a}_i}}(\mathrm{t}) = \left.\frac{\partial \mathbf{\Pi}_{\boldsymbol{\xi}}(\mathrm{t})}{\partial \mathrm{a}_i}\right|_{\mathbf{p}=\mathbf{p}_\mathrm{c}} \in \mathbb{R}^{\mathrm{n}_{\boldsymbol{\xi}} \times \mathrm{n}_\mathbf{p}}, \tag{4.11}$$

$$\mathbf{\Theta}_{\mathrm{a}_i}(\mathrm{t}) = \left.\frac{\partial \mathbf{\Theta}(\mathrm{t})}{\partial \mathrm{a}_i}\right|_{\mathbf{p}=\mathbf{p}_\mathrm{c}} \in \mathbb{R}^{\mathrm{n}_\mathbf{u} \times \mathrm{n}_\mathbf{p}} \tag{4.12}$$

be the gradients (matrices) of respectively $\mathbf{\Pi}$, $\mathbf{\Pi}_{\boldsymbol{\xi}}$ and $\mathbf{\Theta}$ w.r.t. a_i. Analogously to $\mathbf{\Pi}_{\boldsymbol{\xi}}$, the quantity $\mathbf{\Pi}_{\boldsymbol{\xi}_{\mathrm{a}_i}}$ is introduced to evaluate $\mathbf{\Pi}_{\mathrm{a}_i}$. We also define

$$\mathbf{q}_{\mathrm{a}_i}(\mathrm{t}) = \left.\frac{\partial \mathbf{q}(\mathrm{t})}{\partial \mathrm{a}_i}\right|_{\mathbf{p}=\mathbf{p}_\mathrm{c}} \in \mathbb{R}^{\mathrm{n}_\mathbf{q}}, \tag{4.13}$$

$$\boldsymbol{\xi}_{\mathrm{a}_i}(\mathrm{t}) = \left.\frac{\partial \boldsymbol{\xi}(\mathrm{t})}{\partial \mathrm{a}_i}\right|_{\mathbf{p}=\mathbf{p}_\mathrm{c}} \in \mathbb{R}^{\mathrm{n}_{\boldsymbol{\xi}}}, \tag{4.14}$$

$$\mathbf{u}_{\mathrm{a}_i}(\mathrm{t}) = \left.\frac{\partial \mathbf{u}(\mathrm{t})}{\partial \mathrm{a}_i}\right|_{\mathbf{p}=\mathbf{p}_\mathrm{c}} \in \mathbb{R}^{\mathrm{n}_\mathbf{u}} \tag{4.15}$$

respectively as the gradients of the system state $\mathbf{q}$, the controller internal states $\boldsymbol{\xi}$ and the inputs $\mathbf{u}$ w.r.t. changes in a_i, which are also necessary to evaluate $\mathbf{\Pi}_{\mathrm{a}_i}$, $\mathbf{\Pi}_{\boldsymbol{\xi}_{\mathrm{a}_i}}$ and $\mathbf{\Theta}_{\mathrm{a}_i}$. It is possible to compute these quantities along the whole trajectory by the same reasoning as in Eq. (4.9), resulting in

$$\begin{cases} \mathbf{q}_{\mathrm{a}_i}(0) = \mathbf{q}_{\mathrm{a}_{i_0}} = \mathbf{0} \\ \boldsymbol{\xi}_{\mathrm{a}_i}(0) = \boldsymbol{\xi}_{\mathrm{a}_{i_0}} = \mathbf{0} \\ \dot{\mathbf{q}}_{\mathrm{a}_i}(\mathrm{t}) = \mathbf{f}_{,\,\mathbf{q}} \cdot \mathbf{q}_{\mathrm{a}_i} + \mathbf{f}_{,\,\mathbf{u}} \cdot \mathbf{u}_{\mathrm{a}_i} \\ \dot{\boldsymbol{\xi}}_{\mathrm{a}_i}(\mathrm{t}) = \mathbf{g}_{,\,\mathbf{q}} \cdot \mathbf{q}_{\mathrm{a}_i} + \mathbf{g}_{,\,\boldsymbol{\xi}} \cdot \boldsymbol{\xi}_{\mathrm{a}_i} + \mathbf{g}_{,\,\mathrm{a}_i} \\ \mathbf{u}_{\mathrm{a}_i}(\mathrm{t}) = \mathbf{h}_{,\,\mathbf{q}} \cdot \mathbf{q}_{\mathrm{a}_i} + \mathbf{h}_{,\,\boldsymbol{\xi}} \cdot \boldsymbol{\xi}_{\mathrm{a}_i} + \mathbf{h}_{,\,\mathrm{a}_i} \end{cases}, \tag{4.16}$$

which allows us to compute $\mathbf{q}_{\mathrm{a}_i}$, $\boldsymbol{\xi}_{\mathrm{a}_i}$ and $\mathbf{u}_{\mathrm{a}_i}$ by forward integration.

Let now $\mathbf{x}_{,\,\mathbf{y},\,\mathbf{z}}$ be the third order tensor that refers to the second order jacobian of the jacobian matrix $\mathbf{x}_{,\,\mathbf{y}}$ w.r.t. one of its arguments $\mathbf{z}$. Also let $\mathbf{T} \in \mathbb{R}^{\mathrm{n}_1 \times \mathrm{n}_2 \times \mathrm{n}_3}$ be a tensor and $\mathbf{v} \in \mathbb{R}^{\mathrm{n}_3}$ be a vector, we define

$$(\mathbf{T} \circ \mathbf{v})_{i,\,j} = \sum_{k=1}^{\mathrm{n}_3} \mathrm{T}_{i,\,j,\,k} \mathrm{v}_k, \quad \forall (i,\, j) \in [\![1,\, \mathrm{n}_1]\!] \times [\![1,\, \mathrm{n}_2]\!]. \tag{4.17}$$

Differentiating Eq. (4.9) w.r.t. a_i with these notations yields

$$\begin{cases} \boldsymbol{\Pi}_{\mathrm{a}_i}(0) = \boldsymbol{\Pi}_{\mathrm{a}_{i_0}} = \mathbf{0}_{\mathrm{n}_\mathbf{q} \times \mathrm{n}_\mathbf{p}} \\ \boldsymbol{\Pi}_{\boldsymbol{\xi}_{\mathrm{a}_i}}(0) = \boldsymbol{\Pi}_{\boldsymbol{\xi}_{\mathrm{a}_{i_0}}} = \mathbf{0}_{\mathrm{n}_{\boldsymbol{\xi}} \times \mathrm{n}_\mathbf{p}} \\ \\ \dot{\boldsymbol{\Pi}}_{\mathrm{a}_i}(\mathrm{t}) = \Big(\mathbf{f}_{,\,\mathbf{q},\,\mathbf{q}} \circ \mathbf{q}_{\mathrm{a}_i}(\mathrm{t}) + \mathbf{f}_{,\,\mathbf{q},\,\mathbf{u}} \circ \mathbf{u}_{\mathrm{a}_i}(\mathrm{t})\Big) \cdot \boldsymbol{\Pi}(\mathrm{t}) + \mathbf{f}_{,\,\mathbf{q}} \cdot \boldsymbol{\Pi}_{\mathrm{a}_i}(\mathrm{t}) + \\ \qquad \Big(\mathbf{f}_{,\,\mathbf{u},\,\mathbf{q}} \circ \mathbf{q}_{\mathrm{a}_i}(\mathrm{t}) + \mathbf{f}_{,\,\mathbf{u},\,\mathbf{u}} \circ \mathbf{u}_{\mathrm{a}_i}(\mathrm{t})\Big) \cdot \boldsymbol{\Theta}(\mathrm{t}) + \mathbf{f}_{,\,\mathbf{u}} \cdot \boldsymbol{\Theta}_{\mathrm{a}_i}(\mathrm{t}) + \\ \qquad \Big(\mathbf{f}_{,\,\mathbf{p},\,\mathbf{q}} \circ \mathbf{q}_{\mathrm{a}_i}(\mathrm{t}) + \mathbf{f}_{,\,\mathbf{p},\,\mathbf{u}} \circ \mathbf{u}_{\mathrm{a}_i}(\mathrm{t})\Big) \\ \dot{\boldsymbol{\Pi}}_{\boldsymbol{\xi}_{\mathrm{a}_i}}(\mathrm{t}) = \Big(\mathbf{g}_{,\,\boldsymbol{\xi},\,\boldsymbol{\xi}} \circ \boldsymbol{\xi}_{\mathrm{a}_i}(\mathrm{t}) + \mathbf{g}_{,\,\boldsymbol{\xi},\,\mathbf{q}} \circ \mathbf{q}_{\mathrm{a}_i}(\mathrm{t}) + \mathbf{g}_{,\,\boldsymbol{\xi},\,\mathrm{a}_i}\Big) \cdot \boldsymbol{\Theta}(\mathrm{t}) + \\ \qquad \Big(\mathbf{g}_{,\,\mathbf{q},\,\boldsymbol{\xi}} \circ \boldsymbol{\xi}_{\mathrm{a}_i}(\mathrm{t}) + \mathbf{g}_{,\,\mathbf{q},\,\mathbf{q}} \circ \mathbf{q}_{\mathrm{a}_i}(\mathrm{t}) + \mathbf{g}_{,\,\mathbf{q},\,\mathrm{a}_i}\Big) \cdot \boldsymbol{\Pi}(\mathrm{t}) + \\ \qquad \mathbf{g}_{,\,\boldsymbol{\xi}} \cdot \boldsymbol{\Pi}_{\boldsymbol{\xi}_{\mathrm{a}_i}}(\mathrm{t}) + \mathbf{g}_{,\,\mathbf{q}} \cdot \boldsymbol{\Pi}_{\mathrm{a}_i}(\mathrm{t}) \\ \boldsymbol{\Theta}_{\mathrm{a}_i}(\mathrm{t}) = \Big(\mathbf{h}_{,\,\boldsymbol{\xi},\,\boldsymbol{\xi}} \circ \boldsymbol{\xi}_{\mathrm{a}_i}(\mathrm{t}) + \mathbf{h}_{,\,\boldsymbol{\xi},\,\mathbf{q}} \circ \mathbf{q}_{\mathrm{a}_i}(\mathrm{t}) + \mathbf{h}_{,\,\boldsymbol{\xi},\mathrm{a}_i}\Big) \cdot \boldsymbol{\Pi}_{\boldsymbol{\xi}}(\mathrm{t}) + \\ \qquad \Big(\mathbf{h}_{,\,\mathbf{q},\,\boldsymbol{\xi}} \circ \boldsymbol{\xi}_{\mathrm{a}_i}(\mathrm{t}) + \mathbf{h}_{,\,\mathbf{q},\,\mathbf{q}} \circ \mathbf{q}_{\mathrm{a}_i}(\mathrm{t}) + \mathbf{h}_{,\mathbf{q},\mathrm{a}_i}\Big) \cdot \boldsymbol{\Pi} + \\ \qquad \mathbf{h}_{,\,\boldsymbol{\xi}} \cdot \boldsymbol{\Pi}_{\boldsymbol{\xi}_{\mathrm{a}_i}}(\mathrm{t}) + \mathbf{h}_{,\,\mathbf{q}} \cdot \boldsymbol{\Pi}_{\mathrm{a}_i}(\mathrm{t}) \end{cases} . \tag{4.18}$$

To sum up, with the equations of this section, one can compute $\boldsymbol{\Pi}_{\mathrm{a}_i}(\mathrm{t})$, $\boldsymbol{\Pi}_{\boldsymbol{\xi}_{\mathrm{a}_i}}(\mathrm{t})$ and $\boldsymbol{\Theta}_{\mathrm{a}_i}(\mathrm{t})$, that are the gradients of respectively the state, the internal state, and the input sensitivities w.r.t. the optimisation variables $\mathrm{a}_{i \in [\![1,\, \mathrm{n}_\mathbf{a}]\!]}$. These can be fed to an appropriate optimisation routine, by forward integrating both Eq. (4.16) and Eq. (4.18), in order to obtain $\mathbf{q}_{\mathrm{a}_i}(\mathrm{t})$, $\boldsymbol{\xi}_{\mathrm{a}_i}(\mathrm{t})$ and $\mathbf{u}_{\mathrm{a}_i}(\mathrm{t})$, then $\boldsymbol{\Pi}_{\mathrm{a}_i}(\mathrm{t})$, $\boldsymbol{\Pi}_{\boldsymbol{\xi}_{\mathrm{a}_i}}(\mathrm{t})$ and $\boldsymbol{\Theta}_{\mathrm{a}_i}(\mathrm{t})$ along the trajectory. Computing $\boldsymbol{\Pi}(\mathrm{t})$, $\boldsymbol{\Pi}_{\boldsymbol{\xi}}(\mathrm{t})$ and $\boldsymbol{\Theta}(\mathrm{t})$ with Eq. (4.9) gives the current values of the sensitivities, and the gradients indicate the direction ensuring a reduction of the metrics through the steps of the optimisation process.

The metrics that are modified by the optimisation process are parameters that describe the shape of the trajectory, for the chosen class of curves. Therefore, in the following subsection, we detail the selected representation.

4.3.4 Trajectory representation

Compared to [Robuffo Giordano et al., 2018], in which *plain polynomials* were used to represent the trajectories, we now adopt *Bézier curves*, see *e.g.*, [F. Zhou et al., 2011], to specify the shape of the reference motions.

Let $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_{\mathrm{d}_\mathcal{B}}$ be the $\mathrm{n}_\mathbf{a}/\mathrm{n}_\mathbf{y} = \mathrm{d}_\mathcal{B} + 1$ control points of the reference to be tracked,

and also let $\mathrm{s} = \mathrm{t}/\mathrm{T} \in [0,\, 1]$ be the normalised time, then the associated Bézier curve is the set of points defined by the parametric representation

$$\boldsymbol{\mathcal{C}}(\mathrm{s}) = \sum_{i=0}^{\mathrm{d}_{\mathcal{B}}} \mathcal{B}_i^{\mathrm{d}_{\mathcal{B}}}(\mathrm{s})\mathbf{P}_i, \tag{4.19}$$

where $\mathcal{B}_i^{\mathrm{d}_{\mathcal{B}}}(\mathrm{s})$ is a *Bernstein polynomial* of degree $\mathrm{d}_{\mathcal{B}}$, and $\mathbf{P}_i \in \mathbb{R}^{\mathrm{n_y}} \mid i \in [\![0,\, \mathrm{d}_{\mathcal{B}}]\!]$ is the i-th control point of the curve. The $\mathrm{d}_{\mathcal{B}} + 1$ Bernstein polynomials, which are the base function in the Bézier curve expression, are defined by

$$\forall (i,\, \mathrm{s}) \in [\![0,\, \mathrm{d}_{\mathcal{B}}]\!] \times [0,\, 1]\,, \quad \mathcal{B}_i^{\mathrm{d}_{\mathcal{B}}}(\mathrm{s}) = \binom{\mathrm{d}_{\mathcal{B}}}{i} \mathrm{s}^i (1-\mathrm{s})^{\mathrm{d}_{\mathcal{B}}-i}. \tag{4.20}$$

The Bernstein basis polynomials of degree $\mathrm{d}_{\mathcal{B}}$ form a basis for the vector space $\mathbb{R}_{\mathrm{d}_{\mathcal{B}}}[X]$ of polynomials of degree at most $\mathrm{d}_{\mathcal{B}}$. Besides, they also form a partition of unity. Hence, a Bézier curve of degree $\mathrm{d}_{\mathcal{B}}$ is completely defined as

$$\forall \mathrm{s} \in [0,\, 1]\,, \quad \boldsymbol{\mathcal{C}}(\mathrm{s}) = \sum_{i=0}^{\mathrm{d}_{\mathcal{B}}} \binom{\mathrm{d}_{\mathcal{B}}}{i} \mathrm{s}^i (1-\mathrm{s})^{\mathrm{d}_{\mathcal{B}}-i} \mathbf{P}_i. \tag{4.21}$$

Eq. (4.21) can be differentiated w.r.t. time, which yields

$$\forall \mathrm{s} \in [0,\, 1]\,, \quad \dot{\boldsymbol{\mathcal{C}}}(\mathrm{s}) = \frac{\mathrm{d}_{\mathcal{B}}}{\mathrm{T}} \sum_{i=0}^{\mathrm{d}_{\mathcal{B}}-1} \binom{\mathrm{d}_{\mathcal{B}}-1}{i} \mathrm{s}^i (1-\mathrm{s})^{\mathrm{d}_{\mathcal{B}}-1} (\mathbf{P}_{i+1} - \mathbf{P}_i). \tag{4.22}$$

In particular, we have

$$\begin{cases} \dot{\boldsymbol{\mathcal{C}}}(0) = \dot{\boldsymbol{\mathcal{C}}}(\mathrm{t}=0) = \dfrac{\mathrm{d}_{\mathcal{B}}(\mathbf{P}_1 - \mathbf{P}_0)}{\mathrm{T}} \\ \dot{\boldsymbol{\mathcal{C}}}(1) = \dot{\boldsymbol{\mathcal{C}}}(\mathrm{t}=T) = \dfrac{\mathrm{d}_{\mathcal{B}}(\mathbf{P}_{\mathrm{d}_{\mathcal{B}}} - \mathbf{P}_{\mathrm{d}_{\mathcal{B}}-1})}{\mathrm{T}} \end{cases}. \tag{4.23}$$

This latter relation can be useful to formulate linear constraints on the reference trajectory, such as kinematic limit conditions. Of course, Eq. (4.21) can be differentiated w.r.t. time as many times as needed, to add more constraints on the curve, *e.g.* on the acceleration, jerk, and so forth.

Even if this representation is a bit more complex (mathematically) than plain polynomials,

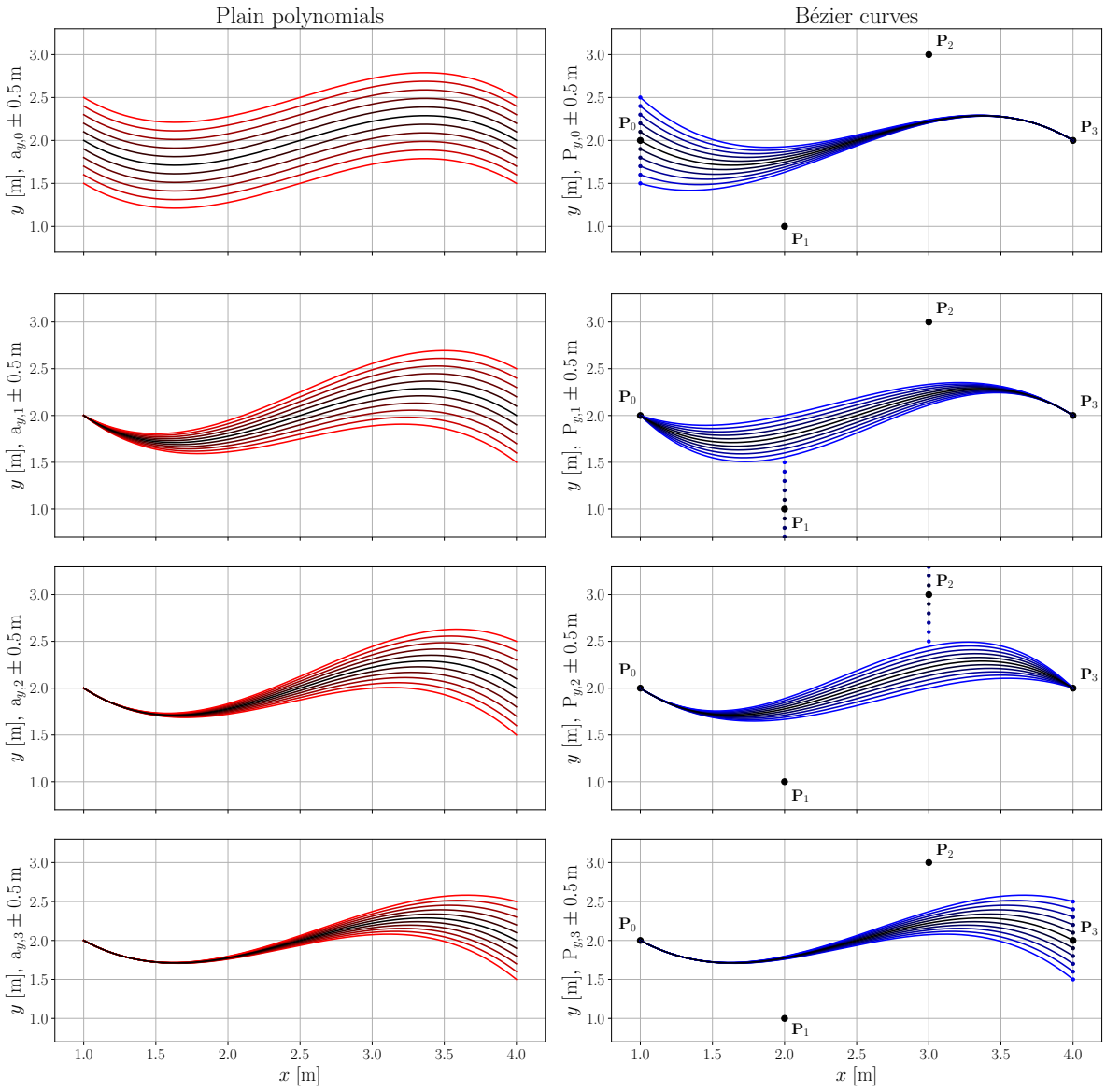

Figure 4.2 – Comparison of the change in shape when modifying one parameter of a cubic curve (on each graphic, the original curve is in black). The plain polynomial representation is displayed on the left column (shades of black to red), while the Bézier curves are on the right (shades of black to blue). In each line, we can observe successively the influence of the offset of the first, second, third and last parameter on the whole curve. Note that here, the offset is applied only in th y-direction, and by $\pm k/10$ [m], where $k \in [\![1, 5]\!]$.

it confers interesting properties, among which we mention:

— the curve begins at $\mathbf{P}_0$ and ends at $\mathbf{P}_{\mathrm{d}_\mathcal{B}}$;

— the curve is tangent to the *control polygon* at its endpoints;

— the entire curve is contained within the *convex hull* of the control points, which *e.g.*, can

be useful for collision avoidance;

— the partition of unity property of the Bernstein polynomials ensures that the shape of the Bézier curve remains the same under translation/rotation of all of its control points.

Moreover, the Bézier curve representation has the advantage of being numerically more stable than simple polynomials, as illustrated in Fig. 4.2. Indeed, the different graphics show that the coefficients of the plain polynomials have unbalanced influence on the whole curve: for instance, modifying the coefficient of lowest degree only translates the curve, see Fig. 4.2 at the top left, while coefficients with higher degree have gradually less influence at the beginning of the curve. Conversely, the control points $\mathbf{P}_i$ of the Bézier curve have a balanced/local influence on the trajectory: one can observe on the right row at each line, that the variation of a control point impacts the curve locally, in a more balanced manner. Hence, the displacement of a control point in its admissible space will ensure a better overall behaviour during the TO process, compared to simple polynomials.

To summarise, with $\mathrm{n}_{\mathbf{a}} = \mathrm{n}_{\mathbf{y}}(\mathrm{d}_{\mathcal{B}} + 1)$, the parameter vector $\mathbf{a} \in \mathbb{R}^{\mathrm{n}_{\mathbf{a}}}$ shapes the reference to be tracked by the system, and contains the information of all control points. Now that we have detailed the trajectory representation that we exploit in this contribution, we formulate the TO problem.

4.4 Trajectory planning

To enhance the global performance of the system at hand, we consider a TO problem: knowing our system dynamics $\mathbf{f}$, referring to Eq. (4.1), a reference trajectory $\mathbf{y}_{\mathrm{d}}(\mathbf{a},\, \mathrm{t})$ defined over the time interval $\mathbb{T}$ (parameterised by the class of curves studied in Subsection 4.3.4), the controller internal dynamics $\mathbf{g}$ and its input function $\mathbf{h}$, both defined in Eq. (4.2), the optimisation consists in finding the optimal vector $\mathbf{a}^\star$, such that

$$\mathbf{a}^\star = \arg\min_{\mathbf{a}\in\mathcal{A}} \left(\mathrm{w}_1 \|\mathbf{\Pi}(\mathrm{T})\|^2 + \mathrm{w}_2 \int_0^{\mathrm{T}} \|\mathbf{\Theta}(\tau)\|^2 \,\mathrm{d}\tau \right), \tag{4.24}$$

where $\|\bullet\|$ is a suitable norm for $\mathbf{\Pi}$ and $\mathbf{\Theta}$, w_1 and w_2 are suitable weights for the optimisation, whose use is described afterwards (other possibilities exist, such as Pareto optimality [I. Y. Kim et al., 2005]), and $\mathcal{A}$ is the admissible set of $\mathbf{a}$. The problem of Eq. (4.24) aims at:

— minimising the closed-loop state sensitivity $\mathbf{\Pi}$ at the final time T, such that the final (possibly perturbed) state $\mathbf{q}(\mathrm{T})$ is as close as possible to its nominal $\mathbf{q}_{\mathrm{c}}(\mathrm{T})$;

— reducing the integral of the closed-loop input sensitivity $\mathbf{\Theta}$, in order to minimise the de-

viation between the (again, possibly perturbed) input $\mathbf{u}(\mathrm{t})$ and its nominal $\mathbf{u}_{\mathrm{c}}(\mathrm{t})$, which would enhance the predictability of the inputs This consideration allows the system to follow optimal paths, during which the inputs result less perturbed if $\mathbf{p}$ deviates $\mathbf{p}_{\mathrm{c}}$. Therefore, the actuators would be less likely to, *e.g.* approach their saturations $\mathbf{u}_{\min}$ and $\mathbf{u}_{\max}$. Note that f we seek to reduce $\|\mathbf{\Pi}\|$ only, then $\|\mathbf{\Theta}\|$ might be greater after the optimisation process. Thereby, there would be higher probability to get a bad behaviour from this 'optimised' motion. To avoid that, we also consider the input sensitivity in the TO problem.

Eq. (4.24) is relevant when needing to reach an accurate pose at final time, while ensuring that the inputs remains close to their nominal. Note that problem

$$\mathbf{a}^{\star} = \arg\min_{\mathbf{a}\in\mathcal{A}} \left(\mathrm{w}_1 \int_0^{\mathrm{T}} \|\mathbf{\Pi}(\mathrm{T})\|^2 \,\mathrm{d}\tau + \mathrm{w}_2 \int_0^{\mathrm{T}} \|\mathbf{\Theta}(\tau)\|^2 \,\mathrm{d}\tau \right), \tag{4.25}$$

which seeks at minimising the norm of the state sensitivity $\mathbf{\Pi}$ during the whole trajectory duration, could also be of interest, if one wants to improve the predictability of the state evolution $\mathbf{q}(\mathrm{t}),\ \forall \mathrm{t} \in \mathbb{T}$. This can be relevant, *e.g.* when needing to avoid obstacle collisions throughout the navigation. In this chapter, we only consider the problem of Eq. (4.24), but in Chapter 5 we will consider the minimisation of Eq. (4.25).

For the sake of finding a motion that will guarantee better performance of the system, we chose to break down problem Eq. (4.24) into several related sub-problems defined by the weighting vector $\Omega = (\mathrm{w}_1,\ \mathrm{w}_2)$, used as follows:

— $\Omega_{\mathbf{\Pi}} = (1,\ 0)$ allocates the whole optimisation for the state sensitivity, and outputs $\mathbf{a}^{\star}_{\mathbf{\Pi}}$;

— $\Omega_{\mathbf{\Theta}} = (0,\ 1)$ aims at minimising the input deviation, and outputs $\mathbf{a}^{\star}_{\mathbf{\Theta}}$;

— $\Omega_{\mathrm{W}} = \left(\frac{1}{\|\mathbf{\Pi}_{\text{opt}}\|},\ \frac{1}{\|\mathbf{\Theta}_{\text{opt}}\|}\right)$ allows the optimiser to reduce both the state and input sensitivities at once: it normalises both $\|\mathbf{\Pi}\|$ and $\|\mathbf{\Theta}\|$ costs, that are not similar metrics by nature, and also grants more weight to the sensitivity that has the lowest value after its first minimisation. This way, the last case should be the one that will give overall the smallest errors for both the state and input. This optimisation outputs $\mathbf{a}^{\star}_{\mathrm{W}}$.

Summarising, the described constrained minimisation problems can be solved by any suitable optimiser. Since we are able to compute the gradients of the metrics studied in Section 4.3, a gradient descent algorithm with linear and non-linear constraints will be used.

Let us consider a case where the initial and final values of $\mathbf{y}_{\mathrm{d}}(\mathbf{a},\ \mathrm{t})$ and their needed time derivatives are given to the optimiser, these linear constraints impose the admissible set $\mathcal{A}$ such that $\mathbf{M}_{\mathcal{C}} \cdot \mathbf{a} = \mathbf{b}$. Here, $\mathbf{M}_{\mathcal{C}}$ is constructed by derivating the Bézier curves of Eq. (4.21), and

$\mathbf{b}$ contains the motion limit conditions. Moreover, the robot is necessarily subject to actuation limits, *i.e.* $\mathbf{u}_{\min}$ and $\mathbf{u}_{\max}$, and these physical capabilities must be considered as non-linear constraints. Vector $\mathbf{a}$ can then be optimised with any constrained non-linear optimisation routine (for instance, we used the well-known 'fmincon' function in Matlab), starting from an initial guess $\mathbf{a}_0$ satisfying both the linear and non-linear constraints, *e.g.*, subject to (s.t.)

$$\begin{aligned} \mathbf{M}_{\mathcal{C}} \cdot \mathbf{a}_0 &= \mathbf{b}, \\ \forall \mathrm{t} \in \mathbb{T},\ \mathbf{u}_{\min} &\leqslant \mathbf{u}_{\mathrm{c}}(\mathrm{t}) \leqslant \mathbf{u}_{\max}. \end{aligned} \tag{4.26}$$

Later, the optimisation routine can be halted with standard termination criteria, *e.g.* based on the gradient norms. Note that since problem Eq. (4.24) is in general non-convex in $\mathbf{a}$, the optimisation algorithm can only guarantee convergence towards a local minimum.

Since we need to evaluate the costs $\|\mathbf{\Pi}\|$ and $\|\mathbf{\Theta}\|$, a suitable norm choice needs to be made. In this context, we chose the Frobenius matrix norm, which derives from the scalar product associated with the matrix space, *i.e.*, for a matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$,

$$\|\mathbf{M}\|_{\mathrm{F}} = \sqrt{\mathrm{Tr}\left(\mathbf{M}^{\mathsf{T}} \cdot \mathbf{M}\right)} = \sqrt{\sum_{i,j} \mathrm{M}_{i,j}^2}, \tag{4.27}$$

but we will discuss in Chapter 6 a better norm choice for the sensitivity metrics.

With the settings described before, the complete closed-loop state/input sensitivities TO problem is written

$$\mathbf{a}^{\star}_{\{\mathbf{\Pi},\,\mathbf{\Theta},\,\mathrm{w}\}} = \arg\min_{\mathbf{a} \in \mathcal{A}} \quad \left(\mathrm{w}_1 \|\mathbf{\Pi}(\mathrm{T})\|_{\mathrm{F}}^2 + \mathrm{w}_2 \int_0^{\mathrm{T}} \|\mathbf{\Theta}(\tau)\|_{\mathrm{F}}^2 \,\mathrm{d}\tau \right) \quad \text{s.t.} \quad \begin{cases} \mathbf{M}_{\mathcal{C}} \cdot \mathbf{a} = \mathbf{b} \\ \mathbf{u}_{\min} \leqslant \mathbf{u}_{\mathrm{c}}(\mathrm{t}) \leqslant \mathbf{u}_{\max}, \quad \forall \mathrm{t} \in \mathbb{T} = [0,\, \mathrm{T}] \end{cases}, \tag{4.28}$$

where $\mathbf{a}$ is the optimisation vector that shapes the reference trajectory, and which always remains within its admissible set $\mathcal{A}$. The problem of Eq. (4.28) aims at minimising the final state error and the integral (through time) of the input error, between the perturbed and the nominal, while verifying kinematic limit conditions (linear equality constraints), and input saturations (non-linear inequality constraints). Therefore, this problem should output reference trajectories whose tracking is intrinsically robust to deviations in the parameters $\mathbf{p}$, provided these latter are not too large. Other formulations of this problem are possible, *e.g.* with an integral cost of the state sensitivity $\mathbf{\Pi}$, see Eq. (4.25), but in this contribution, we only seek that the system reaches

the final point accurately, and thus give full room for this objective.

4.5 Application to a 2D quadrotor

Let us now detail the dynamics and controller equations of a planar quadrotor, in order to get the expressions of Eqs. (4.1) and (4.2).

Let $\mathcal{F}_\mathcal{B} = \{\mathbf{O}_\mathcal{B}, (\mathbf{x}_\mathcal{B}, \mathbf{z}_\mathcal{B})\}$ be the body frame attached to the the quadrotor CoM, with $\mathbf{z}_\mathcal{B}$ aligned with the thrust direction, see Fig. 4.3. For the planar quadrotor, the state consists of the Cartesian position $\mathbf{r} = \begin{bmatrix} x & z \end{bmatrix}^\mathsf{T} \in \mathbb{R}^2$ as well as its first time derivative, the linear velocity $\mathbf{v} = \begin{bmatrix} \mathrm{v}_x & \mathrm{v}_z \end{bmatrix}^\mathsf{T} = \begin{bmatrix} \dot{x} & \dot{z} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^2$, both expressed in the world frame $\mathcal{F}_\mathcal{W} = \{\mathbf{O}_\mathcal{W}, (\mathbf{x}_\mathcal{W}, \mathbf{z}_\mathcal{W})\}$, and of the body orientation $\theta = (\mathbf{z}_\mathcal{W}, \mathbf{z}_\mathcal{B})$ as well as the angular velocity $\omega = \dot{\theta}$. Thus, the state vector of the planar quadrotor is written $\mathbf{q} = \begin{bmatrix} \mathbf{r}^\mathsf{T} & \mathbf{v}^\mathsf{T} & \theta & \omega \end{bmatrix}^\mathsf{T} \in \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R} \times \mathbb{R}$.

Let $\begin{bmatrix} f & \tau \end{bmatrix}^\mathsf{T} \in \mathbb{R}^2$ be respectively the total thrust and torque of the quadrotor, we can distinguish these effective inputs and the actual input vector $\mathbf{u} = \begin{bmatrix} w_\mathrm{R}^2 & w_\mathrm{L}^2 \end{bmatrix}^\mathsf{T} \in \mathbb{R}^2$, that contains the squared versions of the left and right propeller angular rates. These four values are related by

$$\begin{bmatrix} f \\ \tau \end{bmatrix} = \begin{bmatrix} k_f & k_f \\ k_f \ell & -k_f \ell \end{bmatrix} \begin{bmatrix} w_\mathrm{R}^2 \\ w_\mathrm{L}^2 \end{bmatrix} = \mathbf{S} \begin{bmatrix} w_\mathrm{R}^2 \\ w_\mathrm{L}^2 \end{bmatrix}, \tag{4.29}$$

where k_f is the thrust aerodynamic coefficient that characterises the propeller used in the AU, see *e.g.* [Mahony et al., 2012], and ℓ is the arm length.

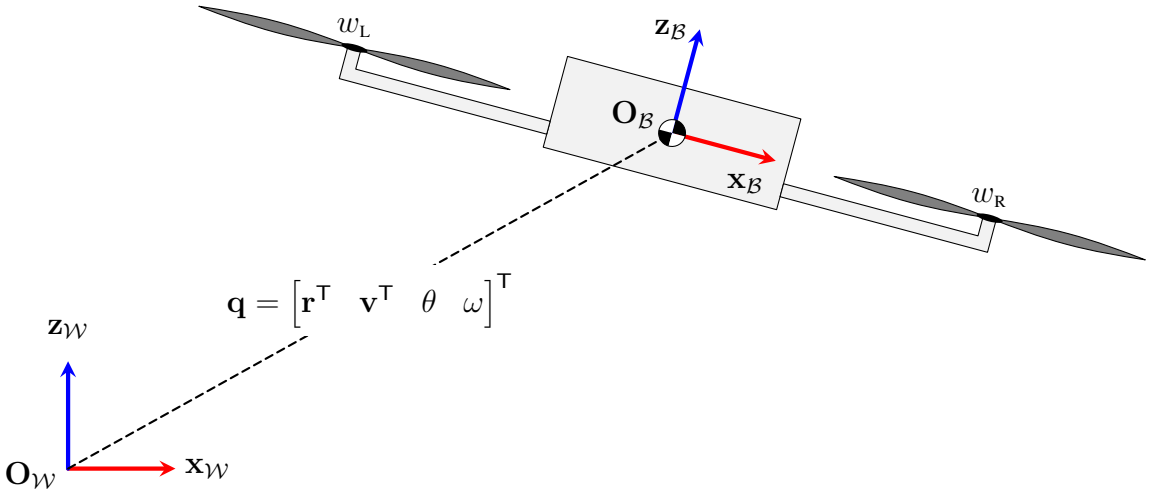

Figure 4.3 – Illustration of the main quantities characterising the planar quadrotor model.

4.5.1 Dynamics in the plane

Let m be the mass of the planar quadrotor and $\mathrm{J} = \mathrm{J}_{\mathbf{O}_\mathcal{B}}$ its scalar inertia. The external forces applied to its body are the weight and the total acting thrust f and torque τ, that are exerted by the two propellers. Thus, the Newton-Euler equations in the plane yield

$$\begin{cases} f(\mathbf{u})\mathbf{z}_\mathcal{B} - mg\mathbf{z}_\mathcal{W} = m\boldsymbol{\gamma}(\mathrm{t}) \\ \tau(\mathbf{u}) = \mathrm{J}\alpha(\mathrm{t}) \end{cases}, \tag{4.30}$$

that is equivalent to

$$\begin{cases} \boldsymbol{\gamma}(\mathrm{t}) = \dfrac{f(\mathbf{u})}{m}\mathbf{R}\cdot\mathbf{z}_\mathcal{W} - g\mathbf{z}_\mathcal{W} \\ \alpha(\mathrm{t}) = \dfrac{\tau(\mathbf{u})}{\mathrm{J}} \end{cases}, \tag{4.31}$$

where $\boldsymbol{\gamma}(\mathrm{t}) = \begin{bmatrix} \ddot{x} & \ddot{z} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^2$ is the UAV linear acceleration, expressed in the world frame $\boldsymbol{\mathcal{F}}_\mathcal{W}$, and $\alpha(\mathrm{t}) = \ddot{\theta}$ is its angular acceleration, expressed in the body frame $\boldsymbol{\mathcal{F}}_\mathcal{B}$. As the body orientation is encoded by $\theta = (\mathbf{z}_\mathcal{W}, \mathbf{z}_\mathcal{B})$, the associate rotation matrix $\mathbf{R} = {}^\mathcal{B}\mathbf{R}_\mathcal{W}$ is defined as

$$\mathbf{R} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \in \mathrm{SO}(2), \tag{4.32}$$

such that $\mathbf{z}_\mathcal{B} = \mathbf{R}\cdot\mathbf{z}_\mathcal{W}$. Now, we can write the state-space model of the plane quadrotor

$$\dot{\mathbf{q}}(\mathrm{t}) = \begin{cases} \dot{\mathbf{r}}(\mathrm{t}) = \mathbf{v}(\mathrm{t}) \\ \dot{\mathbf{v}}(\mathrm{t}) = \dfrac{f(\mathbf{u})}{m}\begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} + \begin{bmatrix} 0 \\ -g \end{bmatrix} \\ \dot{\theta}(\mathrm{t}) = \boldsymbol{\omega}(\mathrm{t}) \\ \dot{\omega}(\mathrm{t}) = \dfrac{\tau(\mathbf{u})}{\mathrm{J}} \end{cases} \tag{4.33}$$

where $\dot{\mathbf{r}}$ and $\dot{\mathbf{v}} = \boldsymbol{\gamma}(\mathrm{t})$ are expressed in the world frame $\boldsymbol{\mathcal{F}}_\mathcal{W}$, whereas $\dot{\theta}$ and $\dot{\omega} = \alpha(\mathrm{t})$ are expressed in the local frame $\boldsymbol{\mathcal{F}}_\mathcal{B}$. Note that even if the input $\mathbf{u}$ of the quadrotor are not explicitly in Eq. (4.33), they affect the values of $\begin{bmatrix} f & \tau \end{bmatrix}^\mathsf{T}$ through Eq. (4.29). Therefore, k_f, ℓ, m, J and g are the parameters that affect the state-space model. In this contribution, we consider that

the mass, the inertia, the aerodynamic thrust coefficient and the arm length measurements are uncertain. Of course, the Earth gravity acceleration constant g value is known with very good precision. Consequently, in this framework we consider $\mathbf{p} = \begin{bmatrix} m & \mathrm{J} & k_f & \ell \end{bmatrix}^\mathsf{T} \in \mathbb{R}^4$ as the parameter vector from which we derive the state and input sensitivity matrices, as well as their gradients.

To summarise, the plane quadrotor main quantities are:

— its state vector $\mathbf{q} = \begin{bmatrix} \mathbf{r}^\mathsf{T} & \mathbf{v}^\mathsf{T} & \theta & \omega \end{bmatrix}^\mathsf{T} \in \mathbb{R}^6$;
— its input vector $\mathbf{u} = \begin{bmatrix} w_\mathrm{R}^2 & w_\mathrm{L}^2 \end{bmatrix}^\mathsf{T} \in \mathbb{R}^2$;
— its parameter vector $\mathbf{p} = \begin{bmatrix} m & \mathrm{J} & k_f & \ell \end{bmatrix}^\mathsf{T} \in \mathbb{R}^4$, where g is omitted.

From these vectors, we can define the dynamics of the quadrotor as $\dot{\mathbf{q}}(\mathrm{t}) = \mathbf{f}(\mathbf{q}, \mathbf{u}, \mathbf{p})$, as a combination of the state-space model of Eq. (4.33) and the input map Eq. (4.29). As detailed in Section 4.3, the state dynamics function $\mathbf{f}$ is required to obtain the state sensitivity by forward integration, see Eq. (4.9). In the next subsection, we complete the missing information by giving the expressions of the controller internal state dynamics function $\mathbf{g}$ and inputs function $\mathbf{h}$, that are also needed for the computations of the internal state and input sensitivities.

4.5.2 Dynamic feedback linearisation controller

The chosen control task is that of letting the quadrotor output $\mathbf{y}(\mathbf{q}) = \mathbf{r} = \begin{bmatrix} x & z \end{bmatrix}^\mathsf{T}$ track a desired motion $\mathbf{y}_\mathrm{d}(\mathbf{a}, \mathrm{t}) \in \mathbb{R}^2$. This is done by implementing a DFL controller with integral term for the best performance in the nominal case $\mathbf{p}_\mathrm{c} = \mathbf{p}$, see, *e.g.* [Mistler et al., 2001]. Differentiating once the linear acceleration of the planar quadrotor w.r.t. time yields the jerk

$$\mathbf{j}(\mathrm{t}) = \frac{\dot{f}}{m} \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} - \frac{f\omega}{m} \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}, \tag{4.34}$$

and doing it once more yields the snap

$$\mathbf{s}(\mathrm{t}) = \frac{\ddot{f}}{m} \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} - \frac{2\dot{f}\omega}{m} \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} + \frac{f\omega^2}{m} \begin{bmatrix} \sin\theta \\ -\cos\theta \end{bmatrix} - \frac{f\dot{\omega}}{m} \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}, \tag{4.35}$$

where we recognise the term $\dot{\omega} = \tau/\mathrm{J}$ of Eq. (4.33), which allows us to rewrite the snap (at the nominal $\mathbf{p} = \mathbf{p}_\mathrm{c}$) as

$$\mathbf{s}(\mathrm{t}) = \underbrace{\begin{bmatrix} -\dfrac{\sin\theta}{m_\mathrm{c}} & -\dfrac{\xi_f \cos\theta}{\mathrm{J}_\mathrm{c}} \\ \dfrac{\cos\theta}{m_\mathrm{c}} & -\dfrac{\xi_f \sin\theta}{\mathrm{J}_\mathrm{c}} \end{bmatrix}}_{\mathbf{M}_\mathrm{DFL}(\mathbf{q},\boldsymbol{\xi},\mathbf{p}_\mathrm{c})} \cdot \begin{bmatrix} \ddot{f} \\ \tau \end{bmatrix} + \underbrace{\frac{\omega}{m_\mathrm{c}} \begin{bmatrix} \xi_f \omega \sin\theta - 2\xi_{\dot{f}} \cos\theta \\ -\xi_f \omega \cos\theta - 2\xi_{\dot{f}} \sin\theta \end{bmatrix}}_{\mathbf{v}_\mathrm{DFL}(\mathbf{q},\boldsymbol{\xi},\mathbf{p}_\mathrm{c})}, \tag{4.36}$$

where, $\boldsymbol{\xi} = \begin{bmatrix} \xi_f & \xi_{\dot{f}} & \xi_x & \xi_z \end{bmatrix}^\mathsf{T} \in \mathbb{R}^4$ denotes the controller internal states vector. Here, ξ_f, $\xi_{\dot{f}}$ are the dynamic extensions of respectively f, $\dot{f}$, and ξ_x, ξ_z are the states of the integral action for the linear position. From $\mathbf{y}_\mathrm{d}(\mathbf{a},\, \mathrm{t})$ and its derivatives w.r.t. time, let us define $\mathbf{r}_\mathrm{d}$, $\mathbf{v}_\mathrm{d}$, $\boldsymbol{\gamma}_\mathrm{d}$, $\mathbf{j}_\mathrm{d}$ $\mathbf{s}_\mathrm{d}$, as the reference position, velocity, acceleration, jerk and snap respectively. Then, we define

$$\begin{cases} \boldsymbol{\gamma}_\xi(\mathrm{t}) = \dfrac{\xi_f}{m_\mathrm{c}} \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} + \begin{bmatrix} 0 \\ -g \end{bmatrix} \\ \mathbf{j}_\xi(\mathrm{t}) = \dfrac{\xi_{\dot{f}}}{m_\mathrm{c}} \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} - \dfrac{\xi_f \omega}{m} \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \\ \boldsymbol{\xi}_{xz}(\mathrm{t}) = \begin{bmatrix} \xi_x & \xi_z \end{bmatrix}^\mathsf{T} \\ \boldsymbol{\nu}_\mathrm{DFL}(\mathrm{t}) = \mathbf{s}_\mathrm{d} + \mathrm{k}_\mathbf{j}(\mathbf{j}_\mathrm{d} - \mathbf{j}_\xi) + \mathrm{k}_{\boldsymbol{\gamma}}(\boldsymbol{\gamma}_\mathrm{d} - \boldsymbol{\gamma}_\xi) + \mathrm{k}_\mathbf{v}(\mathbf{v}_\mathrm{d} - \mathbf{v}) + \mathrm{k}_\mathbf{r}(\mathbf{r}_\mathrm{d} - \mathbf{r}) + \mathrm{k}_\mathbf{i}\boldsymbol{\xi}_{xz} \end{cases}, \tag{4.37}$$

where $\mathbf{k}_\mathrm{c} = \begin{bmatrix} \mathrm{k}_\mathbf{j} & \mathrm{k}_{\boldsymbol{\gamma}} & \mathrm{k}_\mathbf{v} & \mathrm{k}_\mathbf{r} & \mathrm{k}_\mathbf{i} \end{bmatrix}^\mathsf{T} \in (\mathbb{R}_*^+)^5$ are suitable control gains. Now, the dynamics of the controller internal states and the quadrotor control inputs can be written as

$$\begin{cases} \dot{\boldsymbol{\xi}}(\mathrm{t}) = \begin{cases} \dot{\xi}_f(\mathrm{t}) = \xi_{\dot{f}} \\ \dot{\xi}_{\dot{f}}(\mathrm{t}) = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \mathbf{M}_\mathrm{DFL}^{-1} \cdot (\boldsymbol{\nu}_\mathrm{DFL} - \mathbf{v}_\mathrm{DFL}) \\ \dot{\boldsymbol{\xi}}_{xz}(\mathrm{t}) = \mathbf{y}_\mathrm{d}(\mathbf{a},\, \mathrm{t}) - \mathbf{y}(\mathbf{q}) \end{cases} \\ \mathbf{u}(\mathrm{t}) = \mathbf{S}_\mathrm{c}^{-1} \cdot \begin{bmatrix} \xi_f \\ \begin{bmatrix} 0 & 1 \end{bmatrix} \cdot \mathbf{M}_\mathrm{DFL}^{-1} \cdot (\boldsymbol{\nu}_\mathrm{DFL} - \mathbf{v}_\mathrm{DFL}) \end{bmatrix} \end{cases}, \tag{4.38}$$

and gives us the sought expressions for Eq. (4.2). We recall that in Eq. (4.38), $\mathbf{S}_\mathrm{c}$ is the input mapping matrix evaluated at the nominal parameters $\mathbf{p}_\mathrm{c}$.

To summarise this section, we now have detailed the expressions of Eqs. (4.1) and (4.2), which allows us to compute the evolutions of the state, internal state and input sensitivities, as well as their gradients, see Eqs. (4.9) and (4.18). To perform the robust TO described by the problem of Eq. (4.28), one can perform the numerical integration along any reference trajectory $\mathbf{y}_\mathrm{d}(\mathbf{a},\, \mathrm{t})$ represented by the Bézier curves of Eq. (4.21).

4.6 Statistical analysis

In order to test the effectiveness of the previously described method, we conducted a statistical analysis of larger scale than in [Robuffo Giordano et al., 2018], which aims at testing the soundness of Eq. (4.28) when applied to various trajectories. The idea of this analysis is to generate a set of $\mathrm{n}_\oplus$ output targets on which we would like to test the framework, and the corresponding $\mathbf{a}^\star_{\mathbf{\Pi}}$, $\mathbf{a}^\star_{\mathbf{\Theta}}$ and $\mathbf{a}^\star_{\mathrm{W}}$. Then, we seek to evaluate the resulting performance for each trajectory case, by means of statistical analysis of the dynamical behaviour against $\mathrm{n}_{\mathrm{per}}$ parameter perturbations.

4.6.1 Settings

Concretely, the first phase of trajectory generation is done by picking a random final output target: as there is a spatial symmetry of the quadrotor dynamics w.r.t. the initial position, we take it in a right half disc, thus

$$\mathbf{y}_\mathrm{d}(\mathbf{a},\, \mathrm{T}) = \mathrm{r}_\oplus \begin{bmatrix} \cos\theta_\oplus \\ \sin\theta_\oplus \end{bmatrix} \mid (\mathrm{r}_\oplus,\, \theta_\oplus) \in [1,\, 3] \times \left[-\frac{\pi}{2},\, \frac{\pi}{2}\right], \tag{4.39}$$

where $\mathrm{r}_\oplus$ is in $[\mathrm{m}]$ and $\theta_\oplus$ is in $[\mathrm{rad}]$. Both of these values are randomly drawn from a uniform distribution within their respective admissible intervals. The kinematic limit conditions in velocity, acceleration and jerk are set to zero, so that the motion is rest-to-rest. Therefore, the linear constraints vector $\mathbf{b}$ is written as a vertical concatenation of the initial and final positions, velocities, accelerations and jerks for each dimension, namely x and z, which yields

$$\mathbf{b} = \Big[\underbrace{0 \quad \mathrm{r}_\oplus \cos\theta_\oplus \quad \mathbf{0}_{1\times 6}}_{x \text{ information}} \quad \underbrace{0 \quad \mathrm{r}_\oplus \sin\theta_\oplus \quad \mathbf{0}_{1\times 6}}_{z \text{ information}}\Big]^\mathsf{T} \in \mathbb{R}^{16}, \tag{4.40}$$

where $\mathbf{0}_{1\times 6}$ contains the information of initial and final velocities, then accelerations, and then jerks. Note that matrix $\mathbf{M}_{\boldsymbol{\mathcal{C}}}$ from Eq. (4.28) can be easily constructed from the Bézier curve, by deriving it to the jerk at the start and end of the time window $\mathbb{T}$.

We also consider actuation constraints on the total thrust f exerted by the quadrotor, such that $0 < f < 2m_{\mathrm{c}}g$, translated into propellers rotation rates with the inverse mapping of Eq. (4.29). The lower limit condition originates from the DFL controller structure, which is singular for $\xi_f = 0$: indeed, inverting matrix $\mathbf{M}_{\mathrm{DFL}}$ in Eq. (4.38) is not possible for this specific value. Besides, the upper limit keeps the inputs below some saturation, that represents a maximum propeller speed.

For each initial guess $\mathbf{a}_0$ generated according to these rules, the optimiser outputs the associated reference trajectories $\mathbf{a}^\star_{\boldsymbol{\Pi}}$, $\mathbf{a}^\star_{\boldsymbol{\Theta}}$ and $\mathbf{a}^\star_{\mathrm{W}}$. Afterwards, and for each of these reference trajectories, we run:

— a single nominal simulation where $\mathbf{p} = \mathbf{p}_{\mathrm{c}}$, that serves us as the reference;

— $\mathrm{n}_{\mathrm{per}} = 500$ perturbed simulations of the quadrotor tracking these four trajectories, while randomly drawing all the parameters $\mathbf{p}$ from a uniform distribution with a perturbation of $\delta_{\mathbf{p}} = \pm 10\,\%$.

From these $4(1 + \mathrm{n}_{\mathrm{per}})$ simulations, we can measure the final output error square norm

$$\mathbf{E}_{\mathbf{y}} = \|\mathbf{y}(\mathrm{T}) - \mathbf{y}_{\mathrm{c}}(\mathrm{T})\|^2 \tag{4.41}$$

and the average input error square norm

$$\mathbf{E}_{\mathbf{u}} = \int_0^{\mathrm{T}} \|\mathbf{u}(\tau) - \mathbf{u}_{\mathrm{c}}(\tau)\|^2 \,\mathrm{d}\tau. \tag{4.42}$$

Note that in Eq. (4.41), $\mathbf{y}_{\mathrm{c}}(\mathrm{T})$ could be replaced by $\mathbf{y}_{\mathrm{d}}(\mathrm{T})$, since the DFL controller has perfect tracking in the nominal case. For other control policies, it is not the case in general.

Then, on each of these eight resulting sets of error values, we compute the mean and the standard deviation. As a synthesis, starting from a single non-optimised trajectory we end up with sixteen numbers, namely the means $\mu_{\{\mathbf{E}_{\mathbf{y}},\,\mathbf{E}_{\mathbf{u}}\}}$ and the standard deviations $\sigma_{\{\mathbf{E}_{\mathbf{y}},\,\mathbf{E}_{\mathbf{u}}\}}$ of the output and the input errors, for the four optimisation cases ($\mathbf{a}_0$, $\mathbf{a}^\star_{\boldsymbol{\Pi}}$, $\mathbf{a}^\star_{\boldsymbol{\Theta}}$ and $\mathbf{a}^\star_{\mathrm{W}}$).

Finally, we aggregate these numbers over the whole set of $\mathrm{n}_{\oplus}$ targets, by computing the box-plot characteristics of these means and standard deviations. In other words, for every initial trajectory, we compute the median, the first and third quartiles, and the first and last centiles of the $\mathrm{n}_{\oplus}$ error means, and standard deviations, for all four non-optimised and optimised trajectories.

We have done this whole campaign of optimisations and perturbed simulations for two controller cases: one set with no integral term ($\mathrm{k_i} = 0$) and a second one with an integral term ($\mathrm{k_i} > 0$). Anyhow, the control gains $\mathbf{k}_c$ of the DFL controller have been chosen in order to give real and negatives closed-loop poles.

4.6.2 Results with no integral term

Fig. 4.4 shows the resulting boxplots of a full campaign of $\mathrm{n_{per}} = 500$ perturbed simulations for $\mathrm{n}_\oplus = 30$ initial targets for the DFL with no integral term ($\mathrm{k_i} = 0$), displaying only $\mu_{\{\mathbf{E_y}, \mathbf{E_u}\}}$.

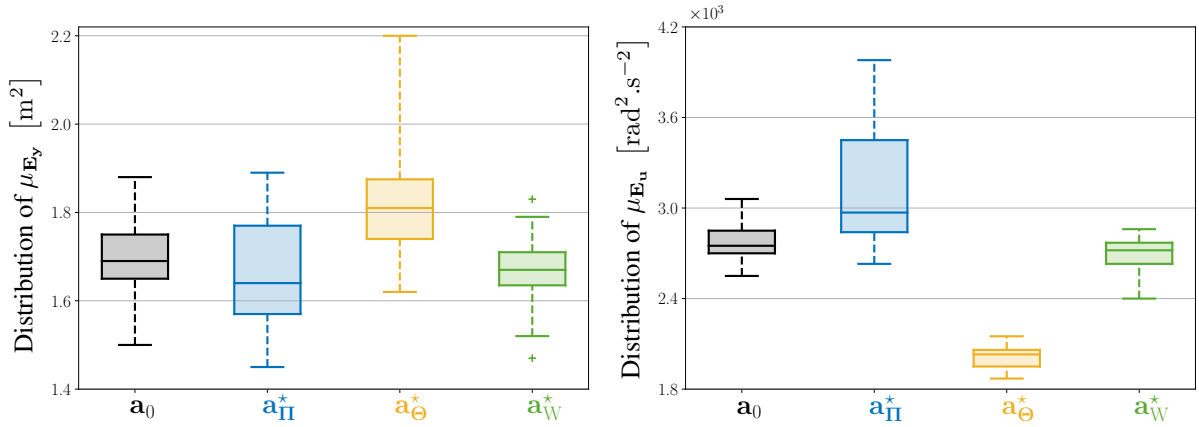

Figure 4.4 – Box-plots of the evaluated performances for the conducted statistical campaign when comparing all the initial guesses $\mathbf{a}_0$ (in black) to their associated optimised trajectories of problem Eq. (4.24), respectively $\mathbf{a}^\star_{\mathbf{\Pi}}$ (in blue), $\mathbf{a}^\star_{\mathbf{\Theta}}$ (in yellow) and $\mathbf{a}^\star_{\mathrm{W}}$ (in light green), when the DFL controller has no integral term ($\mathrm{k_i} = 0$). On the left, we display the distribution of the final output error means, and the distribution of the input error means on the right.

In the left graphic of Fig. 4.4, we can see a slight improvement of the error means height when comparing $\mathbf{a}^\star_{\mathbf{\Pi}}$ to $\mathbf{a}_0$. However, we can see in right graphic that the box-plot of the input tracking error means for $\mathbf{a}^\star_{\mathbf{\Pi}}$ is higher and spreads way more than the initial guess: this means that the generation of a minimal state sensitivity trajectory that will statistically reduce the output tracking error can also have the side effect of making the input less predictable, and more subject to variations due to parameters error. Comparing now $\mathbf{a}^\star_{\mathbf{\Theta}}$ to $\mathbf{a}_0$, we observe a very strong decrease in the input error means (right), while the final output error means (left) increase, which is the reverse case, and leads to the same conclusions. One can now observe that $\mathbf{a}^\star_{\mathrm{W}}$ displays improvements on the two error means, which is what we seek: getting $\mathbf{a}^\star_{\mathbf{\Pi}}$ and $\mathbf{a}^\star_{\mathbf{\Theta}}$ and their respective costs is useful determine Ω_{W}, see Section 4.4, in order to decrease both sensitivities in one adequate optimisation. As we stressed in the previous sections, the

trajectories that are fed to the system have to be minimal in both sensitivities, for accuracy and security. In that way, the reduced errors after optimising for $\mathbf{a}^\star_{\mathrm{W}}$ demonstrate the effectiveness of the method, at least in the conditions we described.

4.6.3 Results with an integral term

Let us now analyse the results for one specific trajectory, extracted from the second campaign of simulations for the DFL with an integral term ($\mathrm{k_i} > 0$). In Fig. 4.5 we can observe the initial guess and its associated optimised trajectories in the output plane. For the nominal case, the quadrotor exactly follows the references with no tracking errors. However, we want to assess the performance of each trajectory in the perturbed case. For this specific motion, after running the $\mathrm{n_{per}}$ perturbed simulations for the four trajectories, we collect all final outputs which gives a cloud of $\mathrm{n_{per}}$ output points for $\mathbf{a}_0$, $\mathbf{a}^\star_{\mathbf{\Pi}}$, $\mathbf{a}^\star_{\mathbf{\Theta}}$ and $\mathbf{a}^\star_{\mathrm{W}}$. Then we chose to plot, for each point cloud, the associated 90% confidence ellipse, as shown in Fig. 4.5, which allows us to easily assess the statistical performance obtained by tracking the reference trajectories of each optimisation.

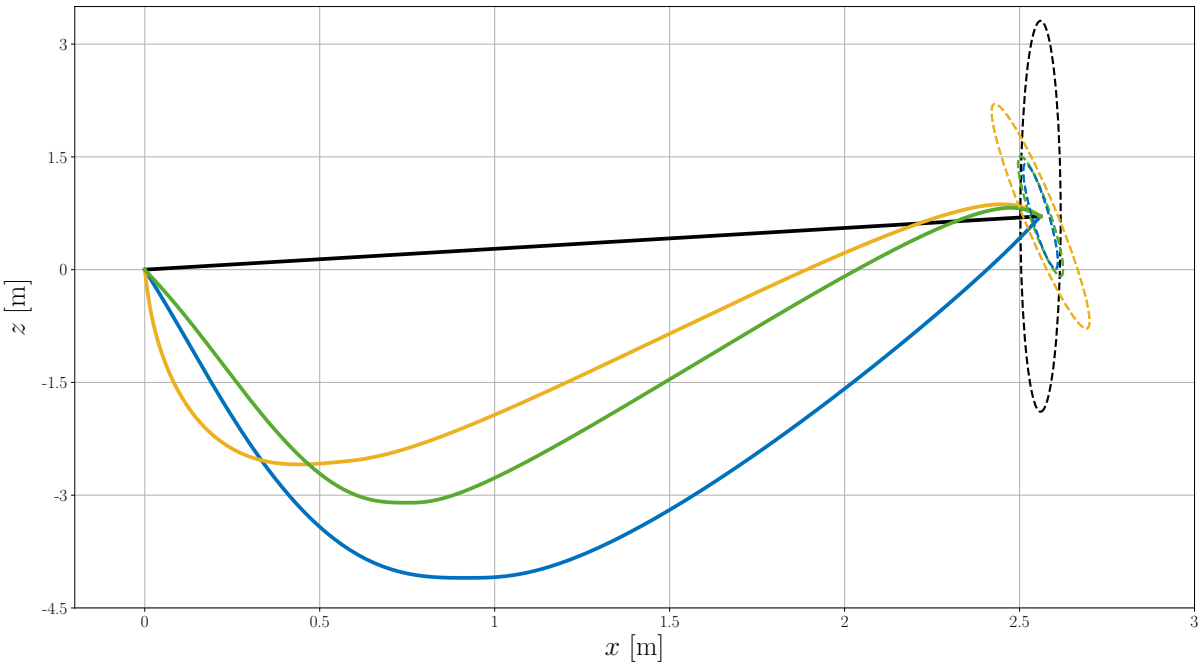

Figure 4.5 – Display of $\mathbf{a}_0$ (in black), $\mathbf{a}^\star_{\mathbf{\Pi}}$ (in blue), $\mathbf{a}^\star_{\mathbf{\Theta}}$ (in yellow) and $\mathbf{a}^\star_{\mathrm{W}}$ (in light green) for problem Eq. (4.24). For each optimisation case, we show the 90% confidence ellipse (dashed lines) associated to the cloud point of the final outputs $\mathbf{y}_i(\mathrm{T})_{i\in[\![1,\,\mathrm{n_{per}}]\!]}$, with $\delta_{\mathbf{p}} = 10\%$.

We can see that $\mathbf{a}^\star_{\mathbf{\Pi}}$ and $\mathbf{a}^\star_{\mathrm{W}}$ are showing both very good results in term of final output errors mean and dispersion. $\mathbf{a}^\star_{\mathbf{\Theta}}$ gives a smaller ellipse than the initial guess. Note that, interestingly,

conversely to the previous statistical analysis, where $\mathbf{\Pi}$ was shown to increase when minimising $\mathbf{\Theta}$, it is not the case for these specific conditions. From these results, we draw the conclusion that the optimised trajectories give lower state errors, with the lowest ones occurring for $\mathbf{a}^{\star}_{\mathbf{\Pi}}$ and $\mathbf{a}^{\star}_{\mathbf{W}}$. Therefore, these two optimisation cases are good references if we need the plan quadrotor to reach the final output target accurately.

Lastly, Fig. 4.6 shows the resulting boxplots of the initial guess $\mathbf{a}_0$ and $\mathbf{a}^{\star}_{\mathbf{W}}$, for a full campaign of perturbed simulations, when the DFL has an integral term ($\mathrm{k_i} > 0$). We chose to only display the results of $\mathbf{a}^{\star}_{\mathbf{W}}$, since these reference trajectories are the ones which are the most accomplished in terms of performance improvement. Anyway, the results for the two other optimisations for this case have the same trend.

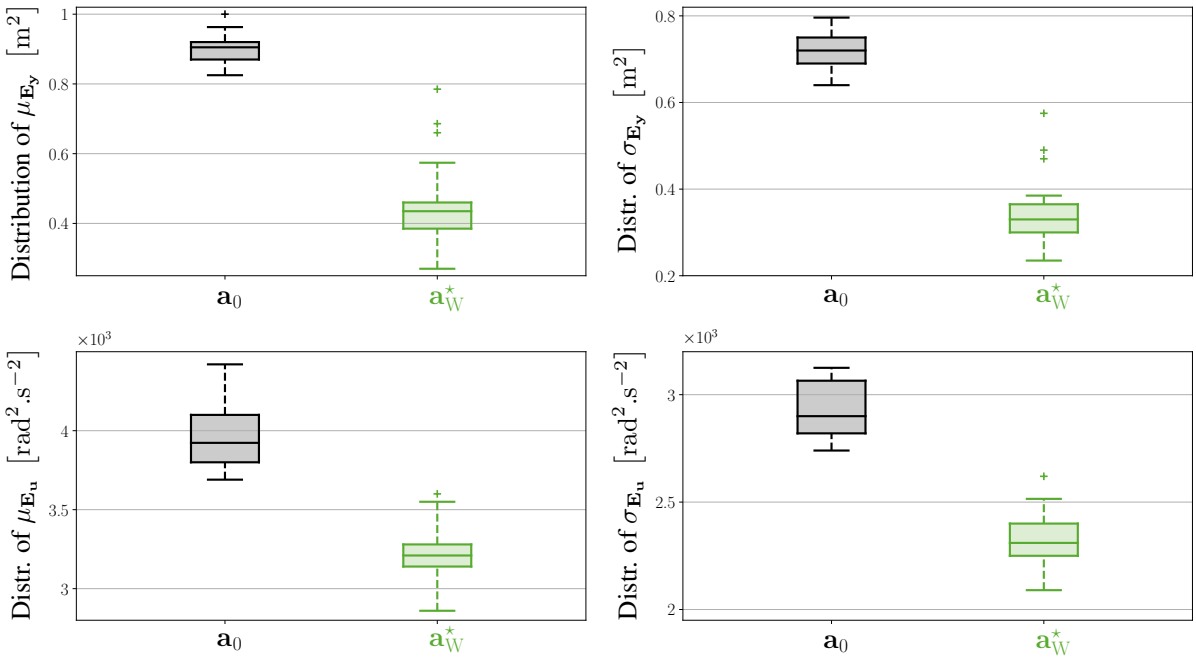

Figure 4.6 – Box-plots of the evaluated performances for the conducted statistical campaign when comparing all the initial guesses $\mathbf{a}_0$ (in black) to their associated trajectories resulting from the weighted optimisation of problem Eq. (4.28), $\mathbf{a}^{\star}_{\mathbf{W}}$ (in light green), when the DFL controller has integral term ($\mathrm{k_i} > 0$). At the top, distribution of the final output error means (left) and standard deviations (right); at the bottom, distribution of the input errors means (left) and standard deviations (right).

On the left we can observe the error means for the output (top) and the inputs (bottom), with significant improvements in both. On the right we can also observe the standard deviation means for the output (top) and the inputs (bottom). We can easily verify that the errors after the simulations were significantly reduced: in average, the state error means and standard deviations were decreased by a factor two, which is very good, and we also see clear improvements

concerning the inputs. In our opinion, these results strongly demonstrate the effectiveness of the method.

4.7 Conclusion

In this chapter, we gave a novel constrained optimisation problem for robust trajectory planning under parametric uncertainties. After defining the associated generic robotic problem, we have introduced the closed-loop input sensitivity, which complements the state sensitivity. We further incorporate it in the objective cost of our TO problem. The methodology for determining the sensitivities and their gradients is described in detail in Section 4.3.

To overcome the stability issues faced by previous contributions that utilised plain polynomials, we adopted a new class of curves, *i.e*. Bézier curves, to represent the planned motions. These offer numerous advantages while ensuring stability, given that their degree remains relatively low. With this new representation, we have defined the state/input sensitivity minimisation problem, under linear and non-linear constraints (respectively for kinematic limit conditions and input saturations). This formulation results in reference trajectories, whose tracking is (by construction) robust against deviations in the model parameters. The cost objective allows to minimise the Frobenius norm of the state sensitivity at the final time, to reach the final output target with enhanced precision, and also allows to reduce the integral of the input sensitivity during the whole motion. This ensures a minimisation of the input deviation (around its nominal), in leads to more predictability in the perturbed cases.

The validity of the proposed optimisation framework was tested using a planar quadrotor equipped with a DFL controller. To this end, an extensive statistical campaign of simulations was conducted to evaluate the performance of the method under uncertain robotic conditions, for multiple output targets. From our point of view, the result section demonstrates that the proposed sensitivity reduction method can significantly enhance the accuracy and safety of systems.

CHAPTER 5

COP: CONTROL & OBSERVABILITY-AWARE PLANNING

5.1 Introduction

In this chapter, we aim to combine two trajectory generation methods: the first, introduced in [Robuffo Giordano et al., 2018] and later extended in Chapter 4, generates *control-aware* trajectories that are robust to parametric uncertainties of the system's model; the second, studied in [Böhm, G. Li, et al., 2020; Hausman, J. Preiss, et al., 2017; J. A. Preiss et al., 2018], results in *observability-aware* trajectories that enhance the online estimation process of the system's state/parameters. Therefore, in this second contribution we aim to answer the question: how to combine control-aware with observability-aware trajectory planning ? These possibly opposite optimisation objectives could be exploited together to improve trajectory tracking and, at the same time, estimation performance. In particular, we show how to exploit all the related metrics in a single objective cost, and formulate a new constrained TO problem, to generate trajectories that have minimal State and Input Sensitivities (S/I-S), and which facilitate the parameter estimation process.

5.1.1 Motivation

A single metric may not fully capture the complexity of a task, or the various factors that contribute to successful execution. By combining multiple metrics, it is possible to identify multiple areas for improvement, and generate trajectories that can mitigate the potential biases and limitations of those who are obtained using a single objective. This can be illustrated by the following: for instance in Section 4.6, we have observed that reducing the state sensitivity alone could lead to an increase of the input sensitivity, and vice versa. This highlights the possible disadvantages of reducing a single metric in an optimisation: even if it ensures better behaviour w.r.t. the corresponding criterion, it could result in a trajectory that makes other aspects perform

much worse, compared to non-optimised motion. Hence, any trajectory planning framework is subject to drawbacks.

5.1.2 Observability-aware trajectories

State estimation with proper system modelling, see [Böhm, Scheiber, et al., 2021], relies on the design of sufficiently informative system input for accurate and fast estimate convergence. The works in [Ansari et al., 2016; Böhm, G. Li, et al., 2020; Hausman, J. Preiss, et al., 2017; Ponda et al., 2009; J. A. Preiss et al., 2018; Van Den Berg et al., 2011; Wilson et al., 2014] show that taking the estimation or parametric uncertainty into account drastically improves system parameter estimation results while allowing task execution. However, these informative trajectories might show poor robustness against uncertainties in the robot model, for the tracking controller that executes them, *e.g.* [Kamel, Stastny, et al., 2017; T. Lee, Leok, et al., 2010a, 2010b].

5.1.3 Control-aware trajectories

To address this issue, the notion of closed-loop state sensitivity has been recently introduced in [Robuffo Giordano et al., 2018] and also extended to closed-loop S/I-S in Chapter 4, as suitable metrics to be optimised. Minimising some appropriate norm of the S/I-S results in trajectories, whose tracking is minimally sensitive to model uncertainties of the robot state and input. This is important as it increases the robustness and accuracy of the trajectory tracking while improving the repeatability of the control inputs under uncertain robotic tasks. However, the S/I-S method is based on the assumption that the state is fully known during the tracking task, which, of course, is impossible. This knowledge of the actual robot state and nominal values of the model parameters, which may not be directly available, can be provided by state estimation.

5.1.4 Link and combination

Therefore, a link between observability-related metrics and S/I-S metrics exists, since the evaluation of S/I-S needs a good knowledge of states and parameters, and the tracking of maximally observable trajectories benefits from increased robustness in the trajectory execution. That fact motivates our study on how to combine both methods in a unified trajectory optimisation problem, taking into account that these two objectives can conflict among themselves, as

confirmed in Section 5.4.

5.1.5 Contributions

In this context, the essential elements of our approach are:

— we propose Control & Observability-aware Planning (COP) as a way to balance two *possibly contradicting* optimisation problems: planning trajectories whose execution is *minimally sensitive* to model uncertainties, and generating trajectories that can be *sufficiently informative* for accurate state/parameter estimation;

— we leverage previous contributions, and combine the metrics in the formulation and resolution of a Single-Objective Optimisation Problem (SOOP) based on the augmented weighted Chebyshev method;

— we replace the (high-order) Bézier curves of Chapter 4 by piece-wise Bézier curves, for dimension reduction;

— for the first time within the sensitivities framework, we examine a quadrotor UAV that navigates in a full three-dimensional space (oriented with unit quaternions), striving to mimic reality as closely as possible;

— we also replace the previously used DFL controller (Subsubsection 4.5.2) with a geometric tracking controller, better suited to real-world scenarios due to its simplicity, and widely utilised in the field of multi-rotor aerial robotics.

This chapter is structured as follows: first, we detail the model of a space quadrotor, equipped with a geometric tracking controller (Section 5.2). Then in Section 5.3 we detail the COP method, by giving the objectives for the TO problem, showing how to combine them with the augmented weighted Chebyshev method, and detailing the multi-steps optimisation procedure. In a case study (Section 5.4, considering the robust trajectory tracking and parameter estimation for the system at hand, we discuss the statistical results of a realistic simulation campaign that validates the potential of the proposed framework. Finally, Section 5.5 concludes this contribution.

The study of this chapter was conducted in collaboration with the research group 'Control of Networked Systems', located at the University of Klagenfurt. This contribution was published in the research community, and then presented at ICRA 2022:

Christoph Böhm, Pascal Brault, et al. [2022], « COP: Control & Observability-aware Planning », *in*: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3364–3370

5.2 Preliminaries

This section serves the purpose of detailing the system at hand for this study, *i.e.* a space quadrotor oriented with unit quaternions, and equipped with a geometric tracking controller.

We have made the choice of representing attitude through the unit length quaternions of $\mathbb{S}$, see Subsubsection 3.3.2.4, to avoid the singularities induced by the gimbal lock of Euler angles (Subsubsection 3.3.2.2). Thus, the encoding of the orientation is not minimal, as we require four values instead of three for orienting the platform in its space of navigation. This contribution marks the first use of a space quadrotor model with the S/I-S framework. Of course, we avoid the use of rotation matrix since nine values instead of three or four would be required in the state vector $\mathbf{q}$. This would make the complexity of the sensitivity derivations explode and is not welcome at all.

Besides, as the DFL controller is complex to implement in practice, *i.e.* it requires to derive the dynamics to the snap, see Subsection 4.5.2, Eqs. (4.35) and (4.36), especially when switching to a space model, again, the complexity is too high. Therefore, we discard it in this contribution, and switch to the much simpler geometric controller of [T. Lee, Leok, et al., 2010a, 2010b], that we adapted to the quaternion representation. The advantages and inconvenients of this control policy will be further studied when detailing the equations.

5.2.1 3D quadrotor dynamics

The state-space model of the quadrotor oriented with unit quaternions, has already been completely derived in Chapter 3. Therefore, we refer to Section 3.4 for detailed information.

5.2.2 Geometric tracking controller

We now present the controller that equips the quadrotor to perform the reference motion tracking. The chosen control task is to let the system output $\mathbf{y}(\mathbf{q}) = \begin{bmatrix} x & y & z & \psi \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^4$ track a desired motion $\mathbf{y}_{\mathrm{d}}(\mathbf{a},\, \mathrm{t}) \in \mathbb{R}^4$, where ψ is the heading angle of the quadrotor, and can easily be linked to the quaternion $\boldsymbol{\rho}$, using successively Eqs. (3.19) and (3.10). In [Robuffo Giordano et al., 2018] and Chapter 4, a DFL controller with integral term was used as a the planar quadrotor tracking controller. This control policy is able to perfectly track any desired motion, in the absence of input constraints and uncertainties in the states and parameters. However, the DFL strategy is rarely used in practice because of its complexity, since it requires additional internal states with complex dynamics, and has a poor robustness against model uncertainties. Therefore,

we chose to use the geometric tracking controller of [T. Lee, Leok, et al., 2010a, 2010b], which performs slightly worse in an ideal case compared to the DFL, but it is much less complex to implement and tune. Indeed, the use of this controller and variants is widespread in the community. Although we implemented the same structure of the controller as in [T. Lee, Leok, et al., 2010a, 2010b], we added some minor changes in order to match the dynamics of the space. We now detail our adaptation of this control law.

First, from $\mathbf{y}_{\mathrm{d}}(\mathbf{a},\,\mathrm{t})$ and its derivatives, we define $\mathbf{r}_{\mathrm{d}}$, $\mathbf{v}_{\mathrm{d}}$, $\boldsymbol{\gamma}_{\mathrm{d}}$, as the reference linear position, velocity and acceleration respectively. Then, let

$$\mathbf{e}_{\mathbf{r}}(\mathrm{t}) = \mathbf{r}(\mathrm{t}) - \mathbf{r}_{\mathrm{d}}(\mathbf{a},\,\mathrm{t}) \in \mathbb{R}^3 \tag{5.1}$$

be the linear position error of the AV, and also let

$$\mathbf{e}_{\mathbf{v}}(\mathrm{t}) = \mathbf{v}(\mathrm{t}) - \mathbf{v}_{\mathrm{d}}(\mathbf{a},\,\mathrm{t}) \in \mathbb{R}^3 \tag{5.2}$$

be the linear velocity error. Moreover we define

$$\boldsymbol{\xi}(t) = \begin{bmatrix} \xi_x \\ \xi_y \\ \xi_z \end{bmatrix} = \int_0^{\mathrm{t}} \mathbf{e}_{\mathbf{r}}(\tau)\,\mathrm{d}\tau \in \mathbb{R}^3 \tag{5.3}$$

as the linear position integrator, and which corresponds to the internal states vector. We chose the control gains vector $\mathbf{k}_{\mathrm{c}} = \begin{bmatrix} \mathbf{k}_{x,\,y,\,z}^{\mathsf{T}} & \mathbf{k}_{\mathrm{v}_x,\,\mathrm{v}_y,\,\mathrm{v}_z}^{\mathsf{T}} & \mathbf{k}_{\xi_x,\,\xi_x,\,\xi_x}^{\mathsf{T}} & \mathbf{k}_{\varphi,\,\theta,\,\psi}^{\mathsf{T}} & \mathbf{k}_{\omega_x,\,\omega_y,\,\omega_z}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^{15}$ such that the geometric controller delivers the best overall tracking performance. It is time-independent, and defines the diagonal matrices $(\mathbf{K}_{\mathbf{r}},\,\mathbf{K}_{\mathbf{v}},\,\mathbf{K}_{\boldsymbol{\xi}},\,\mathbf{K}_{\mathbf{R}},\,\mathbf{K}_{\boldsymbol{\omega}}) \in (\mathbb{R}^{3\times3})^5$. Now, we exploit the linear position, velocity errors, and position integrator, as well as the control gains vector to define the desired third body-fixed frame canonical basis vector

$$\mathbf{r}_{3_{\mathrm{d}}} = \frac{-\mathbf{K}_{\mathbf{r}}\cdot\mathbf{e}_{\mathbf{r}} - \mathbf{K}_{\mathbf{v}}\cdot\mathbf{e}_{\mathbf{v}} - \mathbf{K}_{\boldsymbol{\xi}}\cdot\boldsymbol{\xi} + m(g\mathbf{z}_{\mathcal{W}} + \boldsymbol{\gamma}_{\mathrm{d}})}{\|-\mathbf{K}_{\mathbf{r}}\cdot\mathbf{e}_{\mathbf{r}} - \mathbf{K}_{\mathbf{v}}\cdot\mathbf{e}_{\mathbf{v}} - \mathbf{K}_{\boldsymbol{\xi}}\cdot\boldsymbol{\xi} + m(g\mathbf{z}_{\mathcal{W}} + \boldsymbol{\gamma}_{\mathrm{d}})\|} \in \mathbb{R}^3. \tag{5.4}$$

The desired heading vector is defined using the desired yaw $\psi_{\mathrm{d}} = \psi_{\mathrm{d}}(\mathbf{a},\,\mathrm{t})$, as

$$\mathbf{x}_{\psi_{\mathrm{d}}} = \begin{bmatrix} \cos\psi_{\mathrm{d}} \\ \sin\psi_{\mathrm{d}} \\ 0 \end{bmatrix} \in \mathbb{R}^3 \tag{5.5}$$

and further used to define the second body-fixed frame canonical basis vector

$$\mathbf{r}_{2_\mathrm{d}} = \frac{\left[\mathbf{r}_{3_\mathrm{d}}\right]_\times \cdot \mathbf{x}_{\psi_\mathrm{d}}}{\left\|\left[\mathbf{r}_{3_\mathrm{d}}\right]_\times \cdot \mathbf{x}_{\psi_\mathrm{d}}\right\|} \in \mathbb{R}^3. \tag{5.6}$$

Lastly, we obtain the first body-fixed frame canonical basis vector

$$\mathbf{r}_{1_\mathrm{d}} = \left[\mathbf{r}_{2_\mathrm{d}}\right]_\times \cdot \mathbf{r}_{3_\mathrm{d}} \in \mathbb{R}^3. \tag{5.7}$$

From these computations, we are able to define the desired rotation matrix

$$\mathbf{R}_\mathrm{d}(\mathbf{a},\, \mathrm{t}) = \begin{bmatrix} \mathbf{r}_{1_\mathrm{d}} & \mathbf{r}_{2_\mathrm{d}} & \mathbf{r}_{3_\mathrm{d}} \end{bmatrix} \in \mathrm{SO}(3). \tag{5.8}$$

Now, we can define the attitude error

$$\mathbf{e}_\mathbf{R}(\mathrm{t}) = \frac{1}{2}\left[\mathbf{R}_\mathrm{d}^\mathsf{T}(\mathbf{a},\, \mathrm{t}) \cdot \mathbf{R}(\boldsymbol{\rho},\, \mathrm{t}) - \mathbf{R}^\mathsf{T}(\boldsymbol{\rho},\, \mathrm{t}) \cdot \mathbf{R}_\mathrm{d}(\mathbf{a},\, \mathrm{t})\right]^\vee \in \mathbb{R}^3 \tag{5.9}$$

where $\mathbf{R}(\boldsymbol{\rho})$ is the rotation matrix that encodes the same rotation as the unit quaternion $\boldsymbol{\rho}$, see Eq. (3.19), and lastly the angular velocity error

$$\mathbf{e}_{\boldsymbol{\omega}}(\mathrm{t}) = \boldsymbol{\omega}(\mathrm{t}) - \boldsymbol{\omega}_\mathrm{d}(\mathrm{t}) = \boldsymbol{\omega}(\mathrm{t}) \in \mathbb{R}^3 \tag{5.10}$$

where we make the choice $\boldsymbol{\omega}_\mathrm{d}(\mathrm{t}) = \mathbf{0}$, to reduce the overall complexity of the controller. Indeed, computing $\mathbf{R}_\mathrm{d}(\mathbf{a},\, \mathrm{t})$ is already quite heavy. The resulting fictitious control inputs are then

$$\begin{cases} f(\mathrm{t}) = \left[-\mathbf{K}_\mathbf{r} \cdot \mathbf{e}_\mathbf{r} - \mathbf{K}_\mathbf{v} \cdot \mathbf{e}_\mathbf{v} - \mathbf{K}_{\boldsymbol{\xi}} \cdot \boldsymbol{\xi} + m(g\mathbf{z}_\mathcal{W} + \boldsymbol{\gamma}_\mathrm{d})\right]^\mathsf{T} \cdot (\mathbf{R}(\boldsymbol{\rho}) \cdot \mathbf{z}_\mathcal{W}) \\ \boldsymbol{\tau}(\mathrm{t}) = -\mathbf{K}_\mathbf{R} \cdot \mathbf{e}_\mathbf{R} - \mathbf{K}_{\boldsymbol{\omega}} \cdot \mathbf{e}_{\boldsymbol{\omega}} \end{cases}. \tag{5.11}$$

As defined before in Eqs. (5.1) and (5.10), $\mathbf{e}_\mathbf{r}$, $\mathbf{e}_\mathbf{v}$, $\mathbf{e}_\mathbf{R}$ and $\mathbf{e}_{\boldsymbol{\omega}}$ denote respectively the linear position, linear velocity, attitude, and angular velocity errors. We then compute the real control inputs $\mathbf{u}(\mathrm{t})$ of the quadrotor, via the inverse of the input mapper of Eq. (3.24), that yields

$$\mathbf{u}(\mathrm{t}) = \mathbf{S}_\mathrm{c}^{-1} \cdot \begin{bmatrix} f(\mathrm{t}) \\ \boldsymbol{\tau}(\mathrm{t}) \end{bmatrix} \in \mathbb{R}^4. \tag{5.12}$$

In this controller, we made the choice to fix the desired angular velocity to zero, which cancels terms in the tracking error of the angular velocity, and also in the input torque. Another adaptation has been made to match the orientation of the frames which is different in this con-

tribution, see Fig. 3.3, with $\mathbf{z}_{\mathcal{B}} = \mathbf{R}(\boldsymbol{\rho},\, \mathrm{t}) \cdot \mathbf{z}_{\mathcal{W}}$ oriented up, and not down as in [T. Lee, Leok, et al., 2010a, 2010b].

5.2.3 Piece-wise Bézier curves

The controller is designed to let the quadrotor follow a reference trajectory $\mathbf{y}_{\mathrm{d}}(\mathbf{a},\, \mathrm{t})$, where $\mathbf{a}$ is the parameter vector for the chosen class of curve. In [Robuffo Giordano et al., 2018], plain polynomials were used, but not optimal since the influence of each coefficient was unbalanced. Thereby in Chapter 4 we switched to the use of Bézier curves, since, in this case, adjusting a control point in its admissible space during the optimisation is in general quite stable, from a numerical point of view.

In this work, we go one step further, and provide an even more intuitive and flexible trajectory representation. We adopted piece-wise Bézier curves, see Fig. 5.1, instead of only a single one of high degree as in Chapter 4. In this new implementation, the parameters are not the control points, but way-points along this curve, that set the limit conditions at the beginning and the end of the whole trajectory, as well as between two pieces. The trajectory passes through the way-points with desired velocities and accelerations, and even higher derivatives if needed.

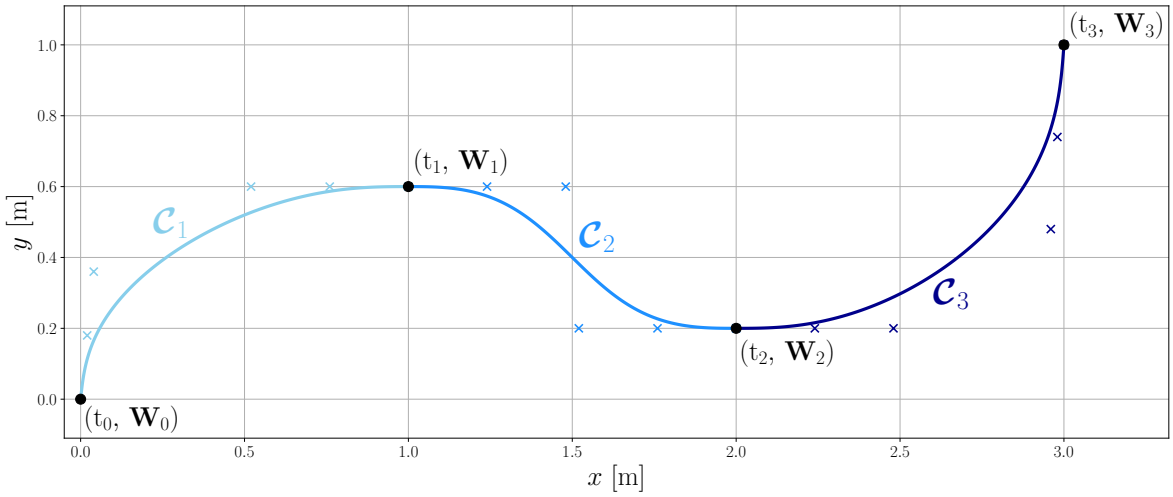

Figure 5.1 – Simple example of a trajectory represented by piece-wise Bézier curves. Here, the number of outputs is $\mathrm{n}_{\mathbf{y}} = 2$ (x and y), the curve has $\mathrm{n}_{\mathrm{p}} = 3$ pieces, each of degree $\mathrm{d}_{\mathcal{B}} = \mathrm{n}_{\mathrm{c.p.}} - 1 = 5$, thus the continuity is ensured in position, speed and acceleration ($\mathrm{n}_{\mathrm{j.c.}} = 3$).

We now give some details about this representation, and show how to translate the limit conditions into Bézier curves. Let $\boldsymbol{\mathcal{C}}_{i \in [\![1,\, \mathrm{n}_{\mathrm{p}}]\!]}$, with $\mathrm{n}_{\mathrm{p}} \geqslant 1$ the number of pieces, be the Bézier

curve of degree $\mathrm{d}_{\mathcal{B}} \geqslant 2$ shaping the trajectory, see Eq. (4.21), then $\mathcal{C}^{\mathrm{d}_{\mathcal{B}}-1}$ continuity is assured between each piece. In total, there are $\mathrm{n_p}+1$ way-points $(\mathrm{t}_i,\ \mathbf{W}_i)$, where t_i is the time associated to the point $\mathbf{W}_i \in \mathbb{R}^{\mathrm{n_y}} \times \mathrm{n_{j.c.}}$, with $\mathrm{n_y}$ the number of dimensions of the trajectory, *e.g.* for the space quadrotor, x, y, z, and the yaw/heading angle ψ, and $\mathrm{n_{j.c.}}$ the number of joining conditions, *e.g.*, $\mathrm{n_{j.c.}} = 1$ for position only, which means every Bézier curve piece would be a straight line segment. The degree $\mathrm{d}_{\mathcal{B}}$ of the Bézier curve, the number of joining conditions $\mathrm{n_{j.c.}}$ and the number of control points $\mathrm{n_{c.p.}}$ are linked by

$$\mathrm{d}_{\mathcal{B}} = 2\mathrm{n_{j.c.}} - 1 = \mathrm{n_{c.p.}} - 1. \tag{5.13}$$

With the conditions mentioned above, it is possible to use the derivatives of the Bézier curve, see *e.g.* the velocity in Eq. (4.22), to construct all the joining conditions. Then, one can define the linear system of equations which allows to obtain the control points of each piece from its limit conditions. The solution of this system outputs the trajectory as an object that contains the tensor of all way-points (of dimension $\mathrm{n_y n_{j.c.}}(\mathrm{n_p}+1)$) and the vector of the times of each way-point $\mathrm{t}_{i\in[\![0,\,\mathrm{n_p}]\!]}$. Therefore, with $\mathrm{n_a} = (\mathrm{n_y n_{j.c.}}+1)(\mathrm{n_p}+1)$, the parameter vector $\mathbf{a} \in \mathbb{R}^{\mathbf{a}}$ contains the information of way-points and the associate times, and shapes the reference to be tracked by the system at hand.

5.3 Control & observability-aware planning

In this section, we detail how to obtain trajectories that have minimal S/I-S and maximum observability.

5.3.1 Objectives for trajectory optimisation

A trajectory $\mathbf{y}_{\mathrm{d}}(\mathbf{a},\,\mathrm{t})$ parameterised by the coefficient vector $\mathbf{a}$, Subsection 5.2.3, can be optimised for different goals, by changing its shape. We represent this goal by the so-called utility function $\mathrm{U}(\mathbf{a})$ which, in our case, is a scalar cost to be minimised, subject to constraints

$$\begin{aligned} \underset{\mathbf{a}\in\mathcal{A}}{\text{minimise}} \quad & \mathrm{U}(\mathbf{a}) \\ \text{s.t.} \quad & \mathbf{C}_{\text{eq.}} = \mathbf{0}_{\mathrm{n}_{\mathbf{C}_{\text{eq.}}}} \\ & \mathbf{C}_{\text{ineq.}} \leqslant \mathbf{0}_{\mathrm{n}_{\mathbf{C}_{\text{ineq.}}}} \end{aligned} , \tag{5.14}$$

where $\mathcal{A}$ is the admissible set for the optimisation vector $\mathbf{a}$, $\mathbf{C}_{\text{eq.}}$ designs the equality constraints, and $\mathbf{C}_{\text{ineq.}}$ the inequality constraints.

5.3.1.1 Closed-loop state sensitivity objective

As the closed-loop state sensitivity has already been defined, and its computation/use has been fully detailed in Chapter 4, please refer to Sections 4.3 and 4.4 for more details.

In this contribution, we are interested in generating trajectories, whose tracking is most insensitive to deviations in the model parameters. In particular, we desire that the (possibly perturbed) state $\mathbf{q}(\mathrm{t}) \mid \mathrm{t} \in \mathbb{T}$ remains close to its nominal $\mathbf{q}_{\mathrm{c}}(\mathrm{t})$ throughout the navigation task, to ensure a better estimation of the state/parameters. Hence, we consider the integral norm of the state sensitivity over the whole time window $\mathbb{T}$, thus

$$\mathrm{U}_{\mathbf{\Pi}}(\mathbf{a}) = \int_0^{\mathrm{T}} \|\mathbf{\Pi}(\tau,\, \mathbf{a})\|_{\mathrm{F}}^2 \,\mathrm{d}\tau, \tag{5.15}$$

where $\|\bullet\|_{\mathrm{F}}^2$ designs the Frobenius norm, see Eq. (4.27).

5.3.1.2 Closed-loop input sensitivity objective

The input state sensitivity was also introduced and detailed in Chapter 4, again please refer to Sections 4.3 and 4.4.

As in Chapter 4, we are interested to improve the input prediction in the perturbed case $\mathbf{p} \neq \mathbf{p}_{\mathrm{c}}$. We take the integral of its norm, which yields the objective function

$$\mathrm{U}_{\mathbf{\Theta}}(\mathbf{a}) = \int_0^{\mathrm{T}} \|\mathbf{\Theta}(\tau,\, \mathbf{a})\|_{\mathrm{F}}^2 \,\mathrm{d}\tau, \tag{5.16}$$

that can be used to generate a trajectory, of which the tracking is intrinsically robust, in terms of control inputs, against uncertainties in the model parameters.

The evaluation of $\mathbf{\Pi}(\mathrm{t})$ and $\mathbf{\Theta}(\mathrm{t})$ is done by leveraging both the dynamical model of the robot $\mathbf{f}$ and the control law $(\mathbf{g},\, \mathbf{h})$, see Section 4.2, Eqs. (4.1) and (4.2), so that any strength/weakness of the chosen control action is correctly taken into account. Brought together for trajectory planning, both these objectives provide the ability to generate the so-called *control-aware* trajectories. We recall the reader that one of the main hypothesis in the sensitivities framework is to consider that the state of the actual system is fully known during the tracking task. Of course, this is not the case as we only have an approximation of the state through the estimators, *e.g.*

KFs. Thereby, ensuring that the state is best known is one of the key remaining aspects that needs to be solved for this kind of trajectory planning.

5.3.1.3 Observability objective

Online state estimation is one way of making the state $\mathbf{q}$ and parameters $\mathbf{p}$ available at runtime, see [Böhm, Scheiber, et al., 2021; Hausman, S. Weiss, et al., 2016; S. Weiss et al., 2012]. This is possible through the state-space model $\mathbf{f}$ and the comparison of sensor measurements with system state-based measurement models $\mathbf{m}(\mathbf{q},\, \mathbf{u},\, \mathbf{p}) \in \mathbb{R}^{\mathrm{n_m}}$.

How well such estimates perform depends on the accuracy of the system model and sensors used, but the reliability of the control inputs fed to the plant are equally important. The EELOG [Hausman, J. Preiss, et al., 2017; J. A. Preiss et al., 2018] works on the idea of the quality of observability, proposed in [Hermann et al., 1977; Krener et al., 2009], which evaluates it over a whole trajectory $\mathbf{y}_{\mathrm{d}}(\mathbf{a},\, \mathrm{t})$ defined on the time window $\mathbb{T} = [0,\, \mathrm{T}]$. It uses a n-th order Taylor expansion to approximate the jacobian matrix that models the sensitivity of the measurements with respect to the control inputs, the state and its changes on a small time horizon $\mathrm{H_t}$, see $\widetilde{\mathbf{W}}_{\mathrm{t_0},\,\mathrm{H_t}}(\mathbf{a})$ in [J. A. Preiss et al., 2018]. Summing all these quality measure segments along the trajectory $\mathbf{y}_{\mathrm{d}}(\mathbf{a},\, \mathrm{t})$ yields the EELOG

$$\widetilde{\mathbf{W}}_{\mathcal{O}}(\mathbf{a}) = \sum_{k=0}^{\mathrm{n_s}} \widetilde{\mathbf{W}}_{k\Delta\mathrm{t},\,\Delta\mathrm{t}}(\mathbf{a}) \in \mathbb{R}^{\mathrm{n_q}\times\mathrm{n_q}} \mid \Delta\mathrm{t} = \frac{\mathrm{T}}{\mathrm{n_s}} \quad , \tag{5.17}$$

where the fixed parameter $\Delta\mathrm{t}$ denotes the number of trajectory segments, and is usually set as high as possible, while being aware of the impact on the computation times. It is chosen empirically, such that it allows to observe the effects of the system dynamics, while maintaining a reasonable approximation error in the Taylor expansion.

The optimisation objective is to improve the observability of the least sensitive state, or combination of states, quantified through the smallest eigenvalue of $\widetilde{\mathbf{W}}_{\mathcal{O}}(\mathbf{a})$. Adding a minus to the smallest eigenvalue makes it usable in a minimisation problem as the objective function

$$\mathrm{U}_{\widetilde{\mathbf{W}}_{\mathcal{O}}}(\mathbf{a}) = -\min\left(\lambda_{\widetilde{\mathbf{W}}_{\mathcal{O},\,i}}\right) = -\lambda_{\widetilde{\mathbf{W}}_{\mathcal{O},\,\min}} \mid i \in [\![1,\, \mathrm{n_q}]\!]. \tag{5.18}$$

The result is a trajectory with optimum observability properties, which intrinsically improves the convergence for the states by lowering the uncertainties and increasing the overall accuracy.

The estimator and, therefore, the derivation of the observability-aware TO for the quadro-

tor UAV case study in Section 5.4 is based on the IEKF implementation presented in [Böhm, Scheiber, et al., 2021], and is new for rigid body dynamics based system models.

5.3.2 Multi-objectives optimisation problem

The problem presented in this work is an example of a Multi-Objective Optimisation Problem (MOOP) trying to optimise for different objectives with constraints.

5.3.2.1 Pareto optimality

Ideally, one would try to find the non-dominated set in the entire feasible set $\mathcal{A}$ for the parameter vector $\mathbf{a}$, a *Pareto optimal set*, or *Pareto front*, see, *e.g.*, [Dächert et al., 2010; Emmerich et al., 2018; Holzmann et al., 2018; Marler et al., 2004; Steuer et al., 1983; Wierzbicki, 1995]. The term *non-dominated* or *optimal* means that the current set does not improve one objective while worsening another. We compute only one point in the Pareto optimal set due to the complexity of the objective functions and the resulting computation times.

In Chapter 4, Linear Scalarisation Problem (LSP) was chosen as a method because it achieved good results in balancing the S/I-S of the plane quadrotor. LSP, see Section 4.4, combines objective functions to a single cost through a linear weighting of each objective, to allow for a single-objective constraint optimisation. This is only a feasible option if the set of all optimal solutions, namely the Pareto front, is convex. LSP comes with its drawbacks as well, often caused by the individual objective functions not building a convex front. To be more specific, in the case of concave Pareto fronts, LSP tends to converge towards extrema solutions, an optimal solution for only one of the objectives. Evenly distributed weights do not produce an evenly distributed representation of the Pareto optimal set. The addition of the observability-awareness through the EELOG as a third objective is the natural evolution of the approach to improve estimation performance, however, non-convexity or concavity, is possible with this addition.

5.3.2.2 Augmented weighted Chebyshev method

To solve the LSP issues, we use the augmented weighted Chebyshev method to balance all three objectives or any combination, compared to Chapter 4. This is done by computing each objective's distance between its current value $\mathrm{U}_i(\mathbf{a})$ and an aspiration point $\cup_i \in \mathbb{R}$, such that each objective is selected by $i \in \left\{\mathbf{\Pi},\, \mathbf{\Theta},\, \widetilde{\mathbf{W}}_{\mathcal{O}}\right\}$. In this framework, the aspiration point $\cup_i$ is the individual objective's minimal value from Eq. (5.14), also called the *utopia point*, therefore

defined as

$$\cup_{i\in\left\{\mathbf{\Pi},\mathbf{\Theta},\widetilde{\mathbf{W}}_{\mathcal{O}}\right\}} = \mathrm{U}_i\left(\mathbf{a}_{i_{\text{opt}}}\right) \leftarrow \begin{cases} \min\limits_{\mathbf{a}\in\mathcal{A}} & \mathrm{U}_i(\mathbf{a}) \\ \text{s.t.} & \mathbf{C}_{\text{eq.}} = \mathbf{0}_{\mathrm{n}_{\mathbf{C}_{\text{eq.}}}} \\ & \mathbf{C}_{\text{ineq.}} \leqslant \mathbf{0}_{\mathrm{n}_{\mathbf{C}_{\text{ineq.}}}} \end{cases} . \tag{5.19}$$

Basically, the utopia point memorises the smallest possible cost value for each objective, optimised as individual. Another important point in the Pareto set is the *nadir point*, defined for each objective as

$$\cap_i = \max_j \mathrm{U}_i\left(\mathbf{a}_{i_{\text{opt}}}\right) \in \mathbb{R} \mid (i,\, j) \in \left(\left\{\mathbf{\Pi},\, \mathbf{\Theta},\, \widetilde{\mathbf{W}}_{\mathcal{O}}\right\}\right)^2, \tag{5.20}$$

which memorises the largest cost of each objective w.r.t. the j-th utopia point. To facilitate comprehension, we schematise in Fig. 5.2 the main quantities of a bi-objective optimisation (the principles can be extended to more than two objectives).

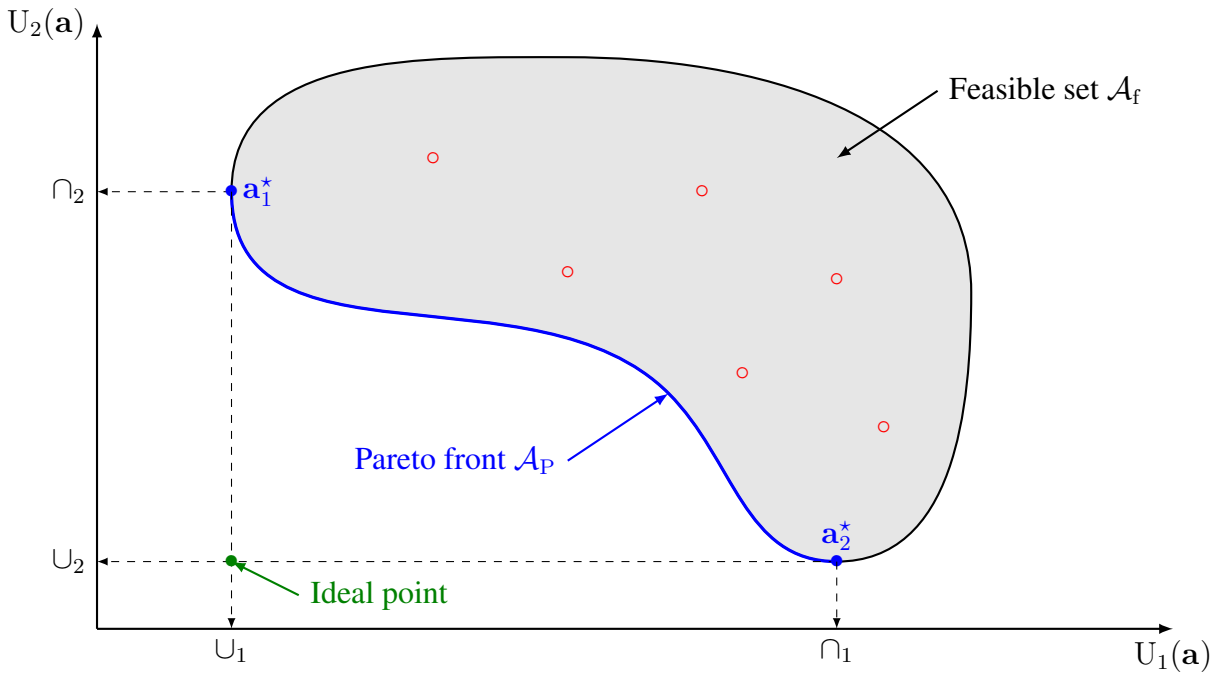

Figure 5.2 – Visualisation of a (possibly) non-convex Pareto front $\mathcal{A}_\mathrm{P}$ (in blue) of a bi-objective optimisation, where $\mathrm{U}_1(\mathbf{a})$ and $\mathrm{U}_2(\mathbf{a})$ are the two objectives (lower is better). The ideal point (in green), which minimises both objectives, is infeasible. The feasible set $\mathcal{A}_\mathrm{f} \subset \mathcal{A}$ is the subset of $\mathcal{A}$ for which the constraints are verified. $\mathbf{a}_1^\star \in \mathcal{A}_\mathrm{P}$ minimises the value of $\mathrm{U}_1(\mathbf{a})$ such that $\mathrm{U}_1(\mathbf{a}_1^\star) = \cup_1$, and defines the nadir point $\cap_2$. $\mathbf{a}_2^\star \in \mathcal{A}_\mathrm{P}$ minimises the value of $\mathrm{U}_2(\mathbf{a})$ such that $\mathrm{U}_2(\mathbf{a}_2^\star) = \cup_2$, and defines the nadir point $\cap_1$. The Pareto front $\mathcal{A}_\mathrm{P}$ dominates the other solutions within $\mathcal{A}_\mathrm{f}$ (in red).

Now let us define the COP utility function as

$$\mathrm{U}_{\mathrm{COP}}(\mathbf{a}) = \max_i \left\{ \frac{\mathrm{w}_i |\mathrm{U}_i(\mathbf{a}) - \cup_i|}{|\cap_i - \cup_i|} \right\} + \rho_{\mathrm{COP}} \sum_j |\mathrm{U}_j(\mathbf{a}) - \cup_j|, \tag{5.21}$$

where again, $(i,\, j) \in \left(\left\{\mathbf{\Pi},\, \mathbf{\Theta},\, \widetilde{\mathbf{W}}_{\mathcal{O}}\right\}\right)^2$ are used to include all objectives. Every point on a Pareto front is a minimum of the Chebyshev function of Eq. (5.21) and achieves Pareto optimality of the solution. Here, the left term contains the user preference weight w_i of one objective, which allows to select one biased solution from the possible set of Pareto optimal solutions, and of course, is such that

$$\sum_i \mathrm{w}_i = 1 \mid i \in \left\{\mathbf{\Pi},\, \mathbf{\Theta},\, \widetilde{\mathbf{W}}_{\mathcal{O}}\right\}. \tag{5.22}$$

This weight is multiplied to the distance between the actual value of the objective and the utopia point $\cup_i$. The denominator of the left term is the distance from the nadir point to the utopia point of each objective, and is used to normalise the Chebyshev function to the interval $[0,\, 1]$. According to [Steuer et al., 1983], ρ_{COP} values should be selected between 0.0001 and 0.01. We refer the reader to [Emmerich et al., 2018; Marler et al., 2004; Wierzbicki, 1995] for deeper information on value choice for this parameter.

The augmented weighted Chebyshev method allows to reduce the original MOOP into a SOOP under constraints. We further discuss how to use the utility function of Eq. (5.21) in Subsubsection 5.3.3.3 and Subsubsection 5.3.3.4.

5.3.3 Implementation

COP uses a multi-step approach to the trajectory optimisation, as the utopia point $\cup_i$ and nadir point $\cap_i$ of each objective $\left\{\mathbf{\Pi},\, \mathbf{\Theta},\, \widetilde{\mathbf{W}}_{\mathcal{O}}\right\}$ need to be known before combining objectives, see Subsubsection 5.3.2.2. The framework is implemented in Python and uses a local derivative-free optimisation routine, namely constrained optimisation by linear approximations (COBYLA) of the open-source library non-linear optimisation (NLOPT). The numerical integration method 'dopri5' of SciPy allows us to compute the individual costs of Eq. (5.15), Eq. (5.16), and Eq. (5.18), during each iteration of the optimisation. Fig. 5.3 gives a graphical overview of the COP method, and one can observe that our strategy returns all possible optimisation cases.

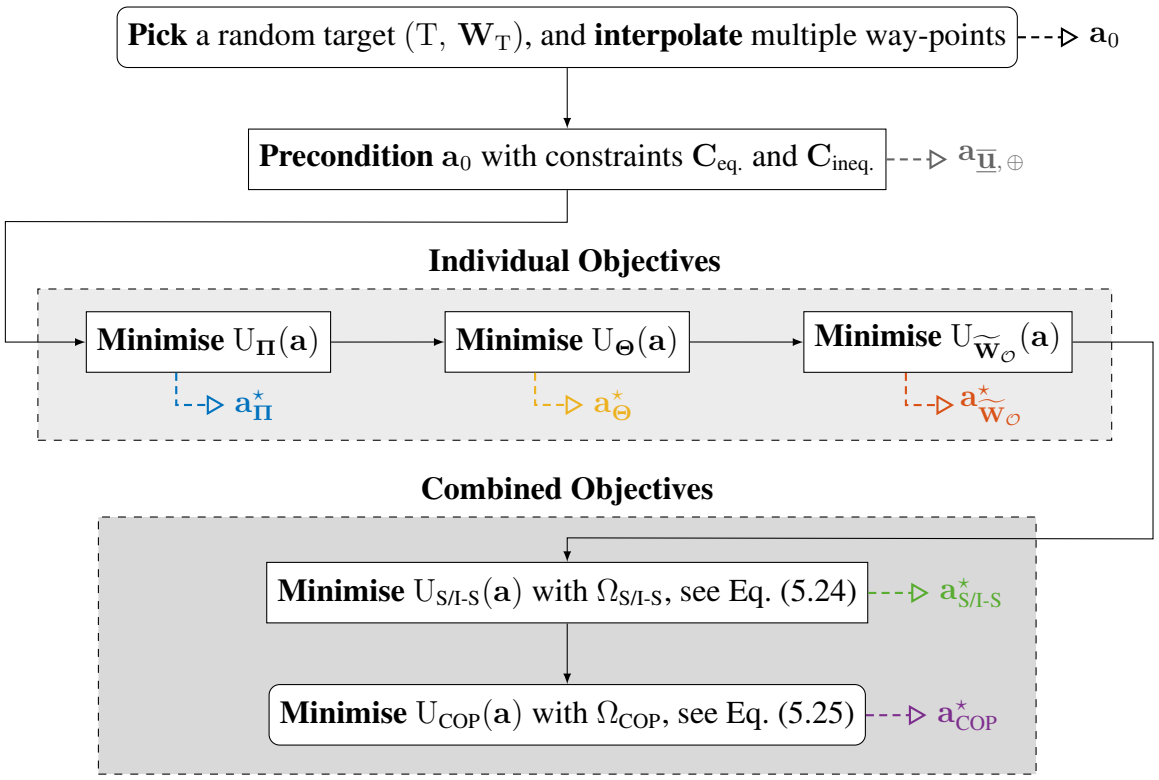

Figure 5.3 – Graphic overview of the multi-step SOOP solved with COP, where $\mathbf{a}_0$ is the initial interpolated trajectory, $\mathbf{a}_{\underline{\mathbf{u}},\oplus}$ is the preconditioned trajectory, *i.e.* it is dynamically feasible and reaches the target accurately in the nominal case $\mathbf{p} = \mathbf{p}_\mathrm{c}$. The optimisations for the individual objectives $\mathbf{\Pi}$, $\mathbf{\Theta}$ and $\widetilde{\mathbf{W}}_\mathcal{O}$ output respectively $\mathbf{a}^\star_{\mathbf{\Pi}}$, $\mathbf{a}^\star_{\mathbf{\Theta}}$, and $\mathbf{a}^\star_{\widetilde{\mathbf{W}}_\mathcal{O}}$. Then, the optimisations are achieved for the combined objectives. $\mathbf{a}^\star_{\text{S/I-S}}$ is the output parameter vector that shapes the S/I-S optimised reference, and lastly the COP method outputs the parameter vector $\mathbf{a}^\star_{\text{COP}}$. Note that, of course, all optimisations are done under constraints, *i.e.* $\mathbf{C}_{\text{eq.}}$ and $\mathbf{C}_{\text{ineq.}}$.

5.3.3.1 Preconditioning

Preconditioning is the first and necessary step in the trajectory generation framework, to ensure that the initial references are dynamically feasible, and let the system reach the final target accurately. Typically, it starts with the selection of a random output target $\mathbf{y}_\mathrm{d}(\mathrm{T})$, and its time derivatives $\dot{\mathbf{y}}_\mathrm{d}(\mathrm{T}) = \mathbf{0}_{\mathrm{n_y}}$ and $\ddot{\mathbf{y}}_\mathrm{d}(\mathrm{T}) = \mathbf{0}_{\mathrm{n_y}}$ (null for rest-to-rest motion), all stored in way-point $\mathbf{W}_\mathrm{T}$, at time T. Then, we simply interpolate additional way-points between the initial and the final, which gives us the initial set of way-points and times $\mathbf{a}_0$, see Subsection 5.2.3 for more precision. All way-points between the initial and target represent the decision variables of the optimisation, and $\mathbf{a} \in \mathbb{R}^{\mathrm{n_a}}$ can be chosen freely within the admissible set $\mathcal{A}$. Then, dynamical constraints are applied to the trajectory through a short optimisation, *i.e.* minimum and maximum propeller angular rates (equivalent to minimum and maximum thrust).

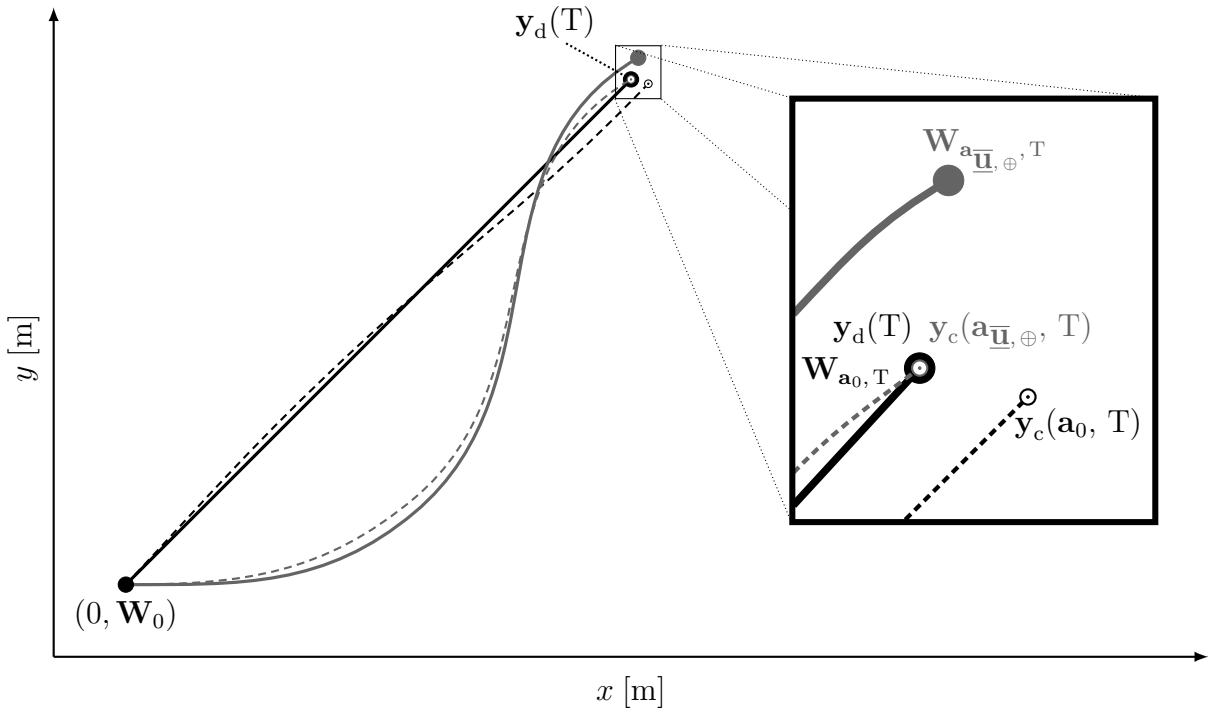

Figure 5.4 – Illustration of the target reach constraint during the preconditioning. The interpolated initial reference $\mathbf{a}_0$ goes to the desired target, such that $\mathbf{W}_\mathrm{T} = \mathbf{W}_{\mathbf{a}_0,\mathrm{T}}$. However, when looking at the zoomed part of the figure on the right, one can observe that the real motion of the system (dashed black line) differs from its reference $\mathbf{a}_0$ (as the controller does not ensure a perfect tracking in the nominal case). Therefore, $\mathbf{y}_\mathrm{c}(\mathbf{a}_0,\ \mathrm{T}) \neq \mathbf{y}_\mathrm{d}(\mathrm{T})$, the target is not accurately reached. After the preconditioning, the reference becomes $\mathbf{a}_{\underline{\mathbf{u}},\oplus}$, and one can observe that its last way-point $\mathbf{W}_{\mathbf{a}_{\underline{\mathbf{u}},\oplus},\mathrm{T}}$ is not centred at the target anymore. This is expected, as the *target reach* constraint was made such that the robot actually reaches the final target in the nominal case (grey dashed line), *i.e.* $\mathbf{y}_\mathrm{c}(\mathbf{a}_0,\ \mathrm{T}) = \mathbf{y}_\mathrm{d}(\mathrm{T})$.

As the geometric controller of Subsection 5.2.2 does not guarantee perfect tracking in the nominal case (as opposed to the DFL controller of Subsection 4.5.2), the preconditioning includes this consideration directly in the linear equality constraints. In fact, we do not want the last way-point to be the target, but rather we seek that the UAV actually ends its navigation task exactly at the desired target (in the nominal case $\mathbf{p} = \mathbf{p}_\mathrm{c}$). Now, $\mathbf{W}_0 = \mathbf{b}_0$ only contains the information of the beginning of the motion: indeed, as the system has not moved at initial time, even if the controller is not perfect, this equation works correctly for the first Bézier curve. Therefore, this small optimisation outputs a reference trajectory for which the last way-point may differ from the real target, see Fig. 5.4, thus, different from $\mathbf{W}_\mathrm{T}$. Thereby, the tracking of this new trajectory ensures that the system reaches the final target accurately when the parameters are correct.

The cost for the preconditioning is chosen to minimise the total curve length and its maximum curvature (or increase its minimum curvature radius), for regularisation of the curve. Hence, the complete optimisation problem can be written

$$\mathbf{a}_{\underline{\mathbf{u}},\oplus} = \arg\min_{\mathbf{a}\in\mathcal{A}} \quad \left(\kappa_{\ell_{\mathcal{C}}} \underbrace{\int_0^{\mathrm{T}} \|\dot{\mathbf{r}}_{\mathrm{c}}(\tau)\| \,\mathrm{d}\tau}_{\text{curve length}} + \kappa_{\mathcal{C}} \underbrace{\max_{\mathrm{t}\in\mathbb{T}} \left\{ \frac{\|[\dot{\mathbf{r}}_{\mathrm{c}}(\mathrm{t})]_{\times} \cdot \ddot{\mathbf{r}}_{\mathrm{c}}(\mathrm{t})\|}{\|\dot{\mathbf{r}}_{\mathrm{c}}(\mathrm{t})\|^3} \right\}}_{\text{maximum curvature}} \right)$$

$$\text{s.t.} \quad \begin{cases} \mathbf{W}_0 = \mathbf{b}_0 \\ \mathbf{y}_{\mathrm{c}}(\mathbf{a},\, \mathrm{T}) = \mathbf{y}_{\mathrm{d}}(\mathrm{T}) \\ \dot{\mathbf{y}}_{\mathrm{c}}(\mathbf{a},\, \mathrm{T}) = \dot{\mathbf{y}}_{\mathrm{d}}(\mathrm{T}) \\ \ddot{\mathbf{y}}_{\mathrm{c}}(\mathbf{a},\, \mathrm{T}) = \ddot{\mathbf{y}}_{\mathrm{d}}(\mathrm{T}) \\ \mathbf{u}_{\min} \leqslant \mathbf{u}_{\mathrm{c}}(\mathrm{t}) \leqslant \mathbf{u}_{\max}, \quad \forall \mathrm{t} \in \mathbb{T} = [0,\, \mathrm{T}] \end{cases}, \tag{5.23}$$

where $(\kappa_{\ell_{\mathcal{C}}},\, \kappa_{\mathcal{C}}) = (5,\, 0.01)$ are suitable gains for the preconditioning cost. The first line of the constraints concerns the linear kinematic constraints at the beginning of the motion, and are applied to the first way-point $\mathbf{W}_0$, with initial limit conditions $\mathbf{b}_0$.

The resulting preconditioned trajectory is parameterised by vector $\mathbf{a}_{\underline{\mathbf{u}},\oplus}$, and serves as a baseline in the latter evaluation in Section 5.4. Indeed, as the reaching of the target is not ensured by $\mathbf{a}_0$ (plus it is may be dynamically unfeasible), it would be unfair to do the comparison with the $\mathbf{\Pi}$, $\mathbf{\Theta}$ and $\widetilde{\mathbf{W}}_{\mathcal{O}}$ optimised references.

5.3.3.2 Optimising for individual objectives

As mentioned before, to allow the combination of multiple objectives, one needs to compute $\cup_i$ and $\cap_i$ of each objective first. We do the minimisation of each objective $\mathrm{U}_{\mathbf{\Pi}}$, $\mathrm{U}_{\mathbf{\Theta}}$, and $\mathrm{U}_{\widetilde{\mathbf{W}}_{\mathcal{O}}}$, defined in Eq. (5.15), Eq. (5.16), and Eq. (5.18) respectively, along with Eq. (5.14) to get those needed extrema points. Of course, the optimisations are conducted under the same constraints than for the preconditioning, Eq. (5.23). From each of these optimisations, we obtain the utopia point and resulting reference trajectory, *i.e.* $(\cup_{\mathbf{\Pi}},\, \mathbf{a}^{\star}_{\mathbf{\Pi}}) \in \mathbb{R} \times \mathbb{R}^{\mathrm{n_a}}$ for the state sensitivity minimisation problem, $(\cup_{\mathbf{\Theta}},\, \mathbf{a}^{\star}_{\mathbf{\Theta}}) \in \mathbb{R} \times \mathbb{R}^{\mathrm{n_a}}$ for the input sensitivity minimisation problem, and lastly $\left(\cup_{\widetilde{\mathbf{W}}_{\mathcal{O}}},\, \mathbf{a}^{\star}_{\widetilde{\mathbf{W}}_{\mathcal{O}}}\right) \in \mathbb{R} \times \mathbb{R}^{\mathrm{n_a}}$ for the EELOG maximisation problem. We further use the generated trajectories to compute the nadir points $\left(\cap_{\mathbf{\Pi}},\, \cap_{\mathbf{\Theta}},\, \cap_{\widetilde{\mathbf{W}}_{\mathcal{O}}}\right) \in \mathbb{R}^3$, see Eq. (5.20).

5.3.3.3 State and input sensitivities optimisation

A first example for the use of Eq. (5.21) is the computation of the S/I-S optimised reference motion. The balancing of state and input sensitivities, similar to Chapter 4, is possible by setting the weight

$$\Omega_{\text{S/I-S}} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^3, \tag{5.24}$$

and using the utopia and nadir points of $\mathrm{U}_{\mathbf{\Pi}}$ and $\mathrm{U}_{\mathbf{\Theta}}$ for normalisation. The rationale behind $\Omega_{\text{S/I-S}}$ is as follows:

— Eq. (5.15) aims at minimising the state sensitivity over the whole trajectory, so that the tracking accuracy of $\mathbf{y}_{\mathrm{d}}(\mathbf{a}^{\star}_{\text{S/I-S}}, \mathrm{t})$ is made most insensitive to uncertainties in the model parameters;

— Eq. (5.16) aims at minimising the input sensitivity over the whole trajectory, in order to obtain control inputs that are most insensitive, *i.e.*, with minimal deviations, against variations of the robot parameters.

For this optimisation, we chose $\rho_{\text{S/I-S}} = 0.0001$ in Eq. (5.21) and obtain the utopia point and reference trajectory $(\cup_{\text{S/I-S}}, \mathbf{a}^{\star}_{\text{S/I-S}}) \in \mathbb{R} \times \mathbb{R}^{\mathrm{n_a}}$. The reduction of both sensitivity indexes in this SOOP provides the ability to generate *control-aware* trajectories.

5.3.3.4 Control & observability-aware optimisation

The possible antagonistic nature of the S/I-S and EELOG needs utility functions like Eq. (5.21) to be balanced successfully. We weight all three objectives equally, thus we chose the weight

$$\Omega_{\text{COP}} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^3, \tag{5.25}$$

with $\rho_{\text{COP}} = 0.0001$. This makes the combination of state and input sensitivity-based with observability-aware trajectory planning possible. This final step gives $(\cup_{\text{COP}}, \mathbf{a}^{\star}_{\text{COP}}) \in \mathbb{R} \times \mathbb{R}^{\mathrm{n_a}}$. Note that in practice, the even distribution of weights might not result in equally improved objectives, due to the possible skewed normalisation from the approximation of nadir and utopia points. To ensure that we get proper trajectories that reduce all indexes, we apply a filtering based on the costs history available from each optimisation run with the *posterior* preferences $\mathrm{U}_{\text{S/I-S}}\left(\mathbf{a}^{\star}_{\text{COP}}\right) < \mathrm{U}_{\text{S/I-S}}\left(\mathbf{a}_{\underline{\overline{\mathbf{u}}},\oplus}\right)$ and $\mathrm{U}_{\widetilde{\mathbf{w}}_{\mathcal{O}}}\left(\mathbf{a}^{\star}_{\text{COP}}\right) < \mathrm{U}_{\widetilde{\mathbf{w}}_{\mathcal{O}}}\left(\mathbf{a}_{\underline{\overline{\mathbf{u}}},\oplus}\right)$. Therefore, the COP framework rejects trajectories that do not fulfil these requirements.

5.4 Statistical results and analyses

In this section, we show how the proposed COP approach, *i.e.* the combination of state and input sensitivities with observability-awareness, can be used to generate trajectories that are less sensitive to changes in the model parameters on the control side, and also favour certain states/parameters in the estimation. As a first, simple case study, we only focus on the thrust and torque coefficients k_f and k_τ of the space quadrotor, thus $\mathbf{p} = \begin{bmatrix} k_f & k_\tau \end{bmatrix}^\mathsf{T} \in \mathbb{R}^2$, see 5.2.1 for more details. These parameters were chosen because they have a significant impact on the tracking performance and are naturally hard to estimate. Furthermore, we only focus on the linear position $\mathbf{r} = \begin{bmatrix} x & y & z \end{bmatrix}^\mathsf{T} \in \mathbb{R}^3$ of the UAV as the state of interest, without considering the other states in the sensitivity evaluation.

5.4.1 Setup and evaluation method

The system model of Subsection 5.2.1, the geometric control law of Subsection 5.2.2, and the optimisation problem of Section 5.3 were implemented in Python. [Böhm, Scheiber, et al., 2021] shows that it is possible to estimate parameters k_f and k_τ. The estimation only needs propeller angular rates, position sensor and IMU measurements, as well as a few *a priori* measurements of the quadrotor. In this first try of fusing both S/I-S and observability in one single framework, remaining in simulation only allows for better repeatability of the experiments, and avoids the introduction of other artefacts due to uncertainties of the real system.

In this empirical evaluation, we compare four versions of the optimised trajectories, for each of all targets:

- the preconditioned trajectory, $\mathbf{a}_{\overline{\underline{\mathbf{u}}},\oplus}$, that is dynamically feasible and reaches the target accurately;
- the trajectory optimised for $\mathbf{\Pi}$ and $\mathbf{\Theta}$, $\mathbf{a}^\star_{\text{S/I-S}}$;
- the EELOG objective optimised trajectory $\mathbf{a}^\star_{\widetilde{\mathbf{W}}_\mathcal{O}}$;
- the new COP trajectory, $\mathbf{a}^\star_{\text{COP}}$, which is optimal in $\mathbf{\Pi}$, $\mathbf{\Theta}$ and $\widetilde{\mathbf{W}}_\mathcal{O}$.

As stated before, the preconditioned reference $\mathbf{a}_{\overline{\underline{\mathbf{u}}},\oplus}$ serves as baseline for all other trajectories. Each motion task is rest-to-rest, has a duration of $\mathrm{t} = 20\,\mathrm{s}$, and is represented by a a piece-wise Bézier curve of five pieces. Each final position target is generated randomly, such that

$$\mathbf{y}_\mathrm{d}(\mathbf{a},\,\mathrm{T}) = \begin{bmatrix} x_\oplus \\ y_\oplus \\ z_\oplus \end{bmatrix} \in [2,\,5] \times [2,\,5] \times \left[-\frac{1}{2},\,1\right], \tag{5.26}$$

where $x_{\oplus}$, $y_{\oplus}$ and $z_{\oplus}$ are in $[\mathrm{m}]$. These three values are drawn with uniform distribution, within their respective admissible intervals.

In total, the multi-steps optimisation approach has been completed for $\mathrm{n}_{\oplus} = 20$ targets, in order to make the results statistically meaningful. The individual cost functions evolutions, positional mean integral error norms, and estimation uncertainties are evaluated from this set of trajectories. It needs to be mentioned that for the positional mean integral error norm evaluation, we chose to randomly perturb the parameters k_f and k_τ, in the ranges of $\delta_{\text{per},\,1\,\%} = \pm 1\,\%$ and $\delta_{\text{per},\,5\,\%} = \pm 5\,\%$, and fulfil $\mathrm{n}_{\text{per}} = 30$ closed-loop perturbed flights, where k_f and k_τ are randomly drawn within their respective possible intervals, with uniform distribution. Perturbations above $5\,\%$ are not considered as such deviations from the nominal values might hint at problems at the parameter identification/estimation. The estimation uncertainty results are based on $\mathrm{n}_{\text{est}} = 10$ runs, for each trajectory, and with randomly chosen incorrect initial guesses at the beginning of the motion, in the range $\delta_{\text{est},\,30\,\%} = \pm 30\,\%$, from ground truths $k_{f_\mathrm{c}} = 3.375 \times 10^{-4}\,[\mathrm{N.rad^2.s^{-2}}]$ and $k_{\tau_\mathrm{c}} = 0.016\,[\mathrm{m}]$.

5.4.2 Discussion

In this subsection, we discuss the performance of the framework in terms of costs evolutions during the different optimisation steps, tracking errors, and parameter estimation errors.

5.4.2.1 Cost functions behaviour

We recorded each objective cost evolution, *i.e.* $\mathrm{U}_{\text{S/I-S}}(\mathbf{a})$, $\mathrm{U}_{\widetilde{\mathbf{W}}_{\mathcal{O}}}(\mathbf{a})$, and $\mathrm{U}_{\text{COP}}(\mathbf{a})$, at all iteration steps, during each optimisation run. We then evaluate the behaviour of the costs by averaging all runs.

As EELOG and S/I-S have different orders of magnitudes, a normalisation was performed by dividing the cost evolution by its initial value, *i.e.* the first cost of the optimisation process. Therefore, we note the normalised cost evolutions

$$\widehat{\mathrm{U}}_i(\mathbf{a}) = \frac{\mathrm{U}_i(\mathbf{a}_\mathrm{n})}{\mathrm{U}_i(\mathbf{a}_{\mathrm{n}=0})} \in \mathbb{R} \mid i \in \left\{\widetilde{\mathbf{W}}_{\mathcal{O}},\ \text{S/I-S},\ \text{COP}\right\}, \tag{5.27}$$

where n is the step of the current optimisation process. In addition, to ensure that the values of the EELOG, potentially growing unbound, remain in a comparable range, we depict its inverse value. A decrease (< 1) means an improvement, while an increase (> 1) indicates a decline in the performance of the respective objective. As the optimisation routine uses a gradient-free

method, the average shows some spikes: this could be addressed in future works, by adopting a gradient-based approach.

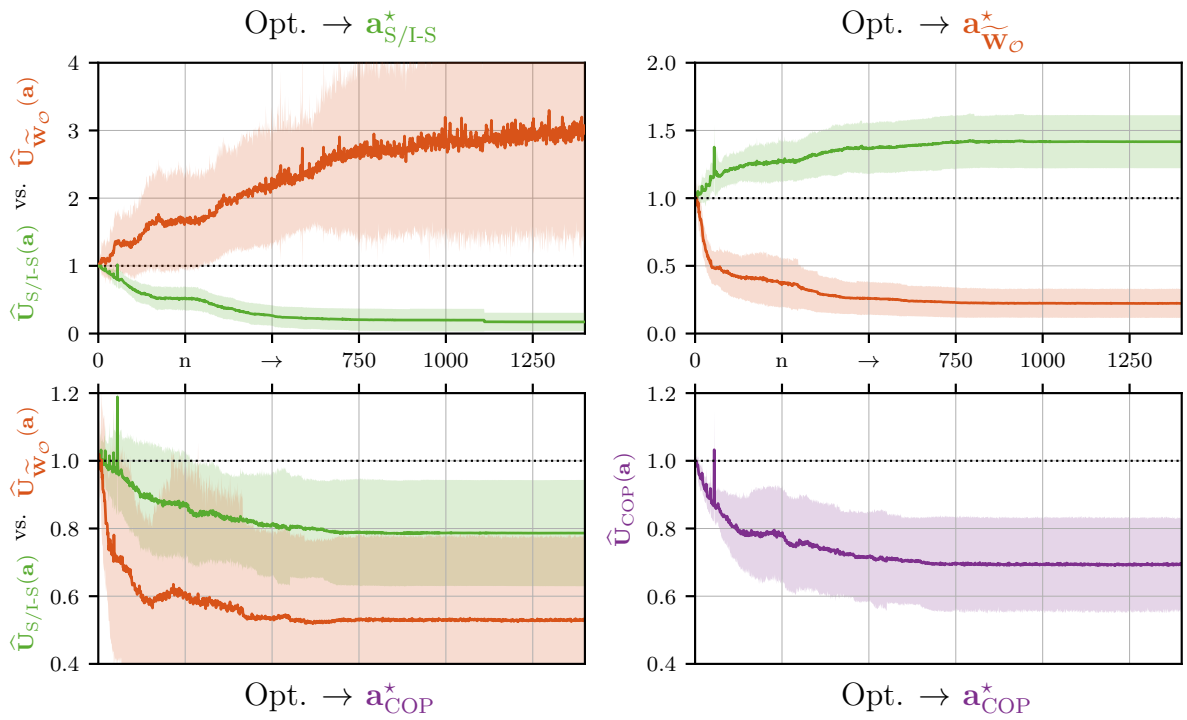

Figure 5.5 – Overview of the normalised cost functions evolutions during the optimisations, depicted by averages and 1σ standard deviations (stds) over $\mathrm{n}_\oplus = 20$ different initial targets. At the top row, the left graphic is obtained when minimising the S/I-S, which outputs $\mathbf{a}^\star_{\text{S/I-S}}$, and the right graphic is obtained when optimising for the EELOG, which outputs $\mathbf{a}^\star_{\widetilde{\mathbf{W}}_\mathcal{O}}$. The bottom row graphics are obtained during the last optimisation, which outputs $\mathbf{a}^\star_{\text{COP}}$. The normalised S/I-S cost $\widehat{\mathrm{U}}_{\text{S/I-S}}(\mathbf{a})$ is in light green, the EELOG $\widehat{\mathrm{U}}_{\widetilde{\mathbf{W}}_\mathcal{O}}(\mathbf{a})$ in dark orange and the COP $\widehat{\mathrm{U}}_{\text{COP}}(\mathbf{a})$ cost is in purple. All optimisations minimise their respective cost function, therefore, a decrease below 1, highlighted by the dotted line, is an improvement of the respective objective.

Fig. 5.5 presents the results of the $\mathrm{n}_\oplus = 20$ individual COP runs. The top left plot shows the average and 1σ standard deviation of the S/I-S-based optimisation, Subsubsection 5.3.3.3, with its cost in light green and the EELOG's cost in dark orange. One can see the decrease in the sensitivity cost, meaning that the state and input sensitivities are minimised, as expected. This comes, however, at the expense of the overall quality of observability, indicated by the increase of the inverse EELOG cost. Therefore, these results seem to indicate that S/I-S and EELOG can be two conflicting objectives. On the top right are both costs, depicted for the quality of observability optimisation, Subsubsection 5.3.3.2. As we chose the inverse here, a decrease is equal to an improvement of the EELOG. Once again, we see the behaviour of the left plot

reflected in this optimisation as well. From those two plots, one can infer that if one objective improves, it is most likely that the other gets worse.

The two plots at the bottom of Fig. 5.5 are the results of the final optimisation runs, based on the COP objective, see Subsubsection 5.3.3.4. We can see that the even distributed weights can still result in slightly skewed solutions, caused by the approximation of the individual utopia and nadir points, respectively $\cup_i$ and $\cap_i$. The solutions themselves are Pareto optimal, and the straightforward filtering ensures an overall decrease of all considered objectives. This indicates that COP can balance, and decrease all objectives.

5.4.2.2 Output tracking error

The evaluation of the tracking performance of the system with its controller is based on the aforementioned set of trajectories, and is done by considering the integral error norm of position, between the nominal and the perturbed simulations, defined as

$$\mathbf{E}_\mathbf{r} = \int_0^\mathrm{T} \|\mathbf{r}(\tau) - \mathbf{r}_\mathrm{c}(\tau)\|^2 \, \mathrm{d}\tau. \tag{5.28}$$

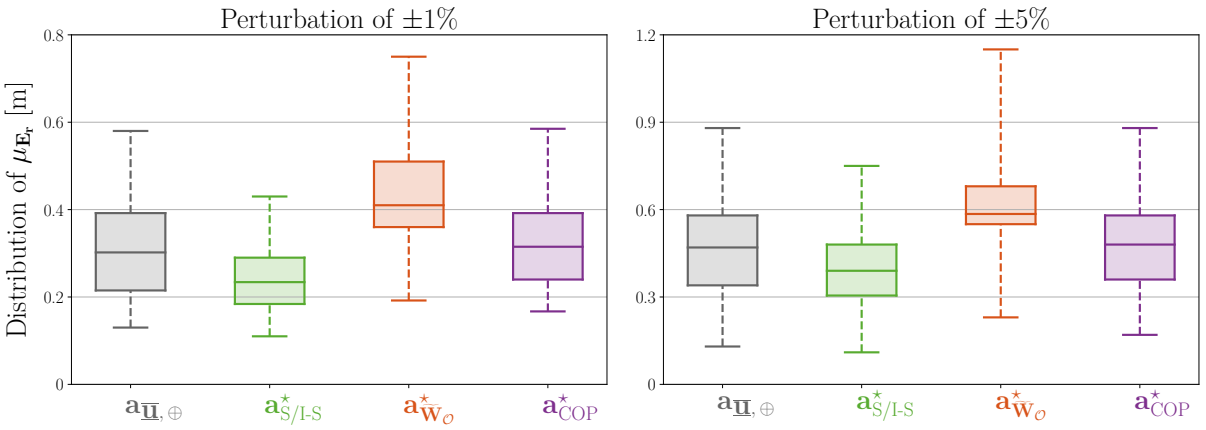

Figure 5.6 – Quartile box plots showing the positional mean integral error norm over the whole trajectory, average over $\mathrm{n}_\oplus = 20$ targets, with $\mathrm{n}_\mathrm{per} = 30$ perturbed closed-loop flights for each resulting reference. The two plots show the influence of different perturbation amplitudes on k_f and k_τ. Note that since the sensitivity is evaluated at $\mathbf{p} = \mathbf{p}_\mathrm{c}$, S/I-S and COP are most effective with small deviations around the nominal values.

The results are depicted in Fig. 5.6, where we can observe the box plots from the statistical data. As already mentioned, each trajectory is flown in simulation $\mathrm{n}_\mathrm{per} = 30$ times with a perturbed set of parameters $\mathbf{p}$ for each flight.

The quartile boxplots of Fig. 5.6 show the average tracking performance of each optimisation case, with different amplitudes of perturbations, represented by the mean of the integral of the squared error norm. The median of the S/I-S optimised trajectory performs the best and the EELOG one the worst, with COP performing between those two, which confirms the costs of Fig. 5.5, and therefore our expectations. Of course, COP optimised shapes cannot reach the same performance level as the S/I-S optimised ones, because of the balancing in Eq. (5.21). However, we can improve the estimation performance at the same time, see Fig. 5.7. The results for $\mathbf{a}_{\overline{\mathbf{u}},\oplus}$ highlights the fact that the initial, not optimised trajectories can lead to greater tracking errors, if the parameters of the UAV deviate too much from the values that are implanted inside the controller. Looking at the comparison between the left figure of boxplots, where $\delta_{\text{per},\,1\,\%} = \pm 1\,\%$, and the one on the right, where $\delta_{\text{per},\,5\,\%} = \pm 5\,\%$, one can see that the farther away $\mathbf{p}$ gets from it nominal value $\mathbf{p}_{\text{c}}$, the less difference is between each objective. This is also expected, as the optimisation evaluates the state and input sensitivities, $\mathbf{\Pi}$ and $\mathbf{\Theta}$ respectively, at the nominal values of the model parameters, see Eq. (4.3) and Eq. (4.4). According to these insights, one might start with estimation-aware trajectories, in order to get $\mathbf{p}$ as close as possible to $\mathbf{p}$, and then maybe switch to control-aware ones, for improved tracking performance, as well as feasibility/security.

5.4.2.3 Estimation error

We used the IEKF implementation of [Böhm, Scheiber, et al., 2021], in Matlab, in order to evaluate the influence of each optimisation case on the estimation of k_f and k_τ, since it has already proven to be good at estimating these parameters. The trajectory optimisation is used to generate artificial position sensor and IMU measurements together, with propeller angular rates for each trajectory. As stated before, each of these recordings has been tested using the IEKF, with initial guesses of values k_f and k_τ, perturbed randomly by $\delta_{\text{est},\,30\,\%} = \pm 30\,\%$, $\mathrm{n}_{\text{est}} = 10$ times. Note that both parameters are poorly observable. To visualise the influence of each optimisation case on the estimation performance, we once again use quartile box plots. Fig. 5.7 shows on the left plot the quartile box plot of k_f and on the right for k_τ. The estimation performance is evaluated by the reduction of uncertainty, represented by the std $\sigma_{\{k_f,\,k_\tau\}}$ at the end of the trajectory, *i.e.* when $\mathrm{t} = \mathrm{T}$.

On Fig. 5.7, one can observe that the S/I-S optimised references perform slightly better than the initial ones. This is probably because we already gain more motion and excitation from the optimisation objective. As expected, based on the optimisation data, the EELOG optimised trajectories perform best, and the results of the COP optimised motions are in between. This could

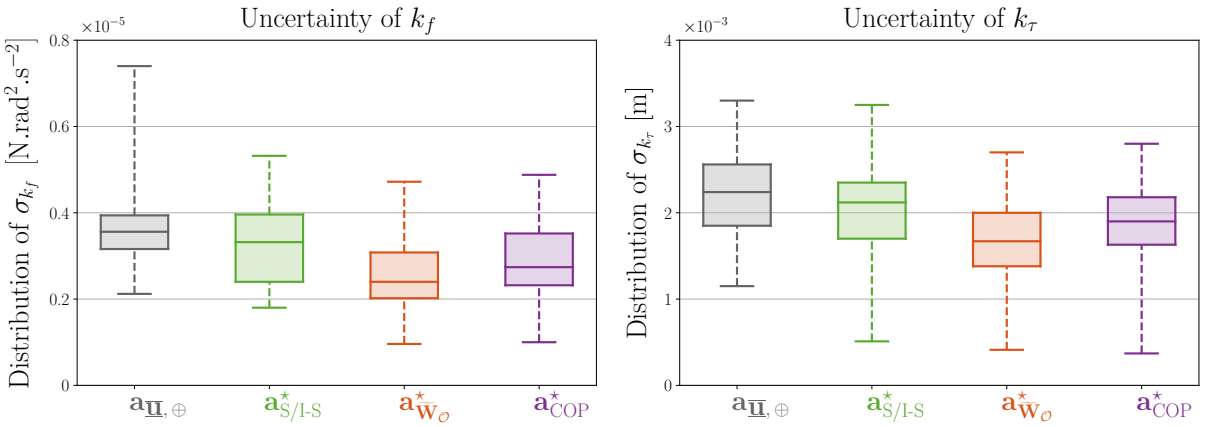

Figure 5.7 – Quartile box plots showing the IEKF's uncertainty based on the state's stds at the end of the trajectory, for a total of $\mathrm{n}_\oplus = 20$ different targets. For each optimisation case, the closed-loop flights are ran with $\mathrm{n}_{\text{est}} = 10$ different initial guesses at the start. The thrust force coefficient k_f and torque coefficient k_τ box plots are depicted respectively on the left and right. As expected, the preconditioned trajectories have the worst estimation performance, and the EELOG ones have the best. COP optimised reference trajectories show a good compromise for the estimation performance.

have been anticipated: due to the balancing of two objectives, we are not able to perform as well as a single objective optimised trajectory. These results were already indicated in Fig. 5.5.

All the presented results support our claim that our proposed COP objective can balance two possibly opposing objectives, while maintaining good performance in estimation, tracking accuracy, and security.

5.5 Conclusion

In this chapter, the COP framework proposes an attempt to combine the closed-loop state and input sensitivities with the observability metrics, for robust and informative trajectory planning. We have made an evaluation of our strategy in a statistical evaluation of tracking/estimation performances.

Intuitively, taking state and input sensitivities into account during the trajectory generation might result in non-informative trajectories for state/parameter estimation. However, informative motions for the estimation process are often difficult to control and likely cause higher tracking errors. Our statistical case study of the space quadrotor UAV, equipped with a modified version of the geometric controller, focused on parameters that have a significant impact on the tracking error, and are poorly observable. It provided insights into the negative correlation between the control-aware and observability-aware trajectory planning. Indeed, we were

able to show that both objectives are often conflicting, meaning that while one can improve, the other has high probability to get worse. Applying the augmented weighted Chebyshev method in our multi-step approach to such a MOOP allows to balance both in a SOOP, resulting in optimal reference motions, for trajectory tracking, security, and at the same time, estimation performances.

To summarise, we have successfully shown that it is important to consider both objectives as they correlate with each other. The insights and results give a motivation to go forward with more sophisticated MOOP approaches. Further research should be conducted towards a real-world closed-loop flight using the estimation in the optimisation as feedback.

Our novel COP framework is the first that uses these possibly opposing objectives in a SOOP based on the augmented weighted Chebyshev method to perform the balancing of them and generation of piece-wise Bézier curve-based trajectories. Statistically relevant simulations for a space quadrotor UAV case study produce results that support our claims and show the negative correlation between both objectives. We were able to reduce the positional mean integral error norm as well as the estimation uncertainty with the same trajectory to comparable levels of the trajectories optimised with individual objectives.

CHAPTER 6

TUBE-BASED TRAJECTORY OPTIMISATION FOR ROBOTS WITH PARAMETRIC UNCERTAINTIES

6.1 Introduction

In this last contribution, we detail a new version of our robust trajectory generation framework, which extends the approaches of [Robuffo Giordano et al., 2018], Chapters 4 and 5, by providing a new possible norm for the sensitivity metrics, which, from our point of view, ensures even more security than with the previously used Frobenius matrix norm. Thanks to this upgrade, we now only exploit the input sensitivity in the constraints of the optimisation problem, in a way which forces the inputs inside the actuation limits, even with the worst perturbations. Therefore, full space is available in the optimisation cost for the state/output sensitivity. As in [Böhm, Brault, et al., 2022], we also use piece-wise Bézier curves to represent the reference trajectory to be tracked. For a space quadrotor model with a shift in the CoG, and equipped with a geometric tracking controller, we optimised a set of trajectories, and conducted an extensive campaign of simulations, that aims at evaluating the relevance of this new norm. The key elements of this contribution are:

— we define the first-order closed-loop output sensitivity, detail its relationship to the state sensitivity, and show how to obtain it by forward integration (as in Chapter 4);

— we use the knowledge of the parameters ranges of uncertainty in order to construct a new norm for the sensitivities, that has a meaning in the state/input/output spaces, and further use it to formulate a new optimisation problem, that aims at minimising this new norm of the state/output sensitivities;

— thanks to this new norm formulation, we now use the input sensitivity to construct the *worst deviation tubes* around the nominal inputs, which, compared to Chapters 4 and 5, allows us to remove it from the cost: now, we rather express the input saturation con-

straints with the worst deviation tubes, instead of the nominal inputs: this guarantees the dynamic feasibility of the motion task, even with the most disadvantageous deviations of the parameters (provided they remain within the uncertainty ranges);

— for the case study, we now consider a space quadrotor with a possible shift in the CoM, still oriented through unit quaternions;

— we validate the approach through a statistical campaign of perturbed simulations.

This chapter is structured as follows: in Section 6.2 we first refer to the main notions and equations needed to compute the different sensitivity metrics. The upgrades on the sensitivity norm and the updated optimisation problem that are presented in Section 6.3 and Section 6.4, are then tested in an extensive campaign of perturbed simulations Section 6.6, for an upgraded space quadrotor model, see Section 6.5: the analysis of the results demonstrates the improvement in closed-loop performance when minimising the sensitivity indexes. Section 6.7 concludes this contribution, and opens to further perspectives.

This chapter will be submitted soon in the research community:

Pascal Brault and Paolo Robuffo Giordano [2023], « Tube-Based Trajectory Optimisation for Robots with Parametric Uncertainty », *in*: In preparation for the IEEE Robotics and Automation Letters (RA-L)

6.2 Essentials

This section serves the purpose of building upon the main notions that were already presented earlier in Chapter 4. In particular, we define the novel *closed-loop output sensitivity* and show how to obtain it by numerical integration.

As the fundamental generic problem that motivates this thesis has already been defined in Section 4.2, we refer the reader to this section for complete information. In the same manner, the metrics that we use to tackle this problem (namely the state and input sensitivities) were already defined in Section 4.3, please refer to this section for full details.

As already established in Chapters 4 and 5, the reference $\mathbf{y}_{\mathrm{d}}(\mathbf{a},\, \mathrm{t}) \in \mathbb{R}^{\mathrm{n_y}}$ is to be followed by some quantity of interest of the system at hand, that is often, its output $\mathbf{y}(\mathbf{q}) \in \mathbb{R}^{\mathrm{n_y}}$. Therefore, it is fairly common that we do not desire the whole state to be more predictable, but rather only the output: *e.g.* in Chapter 4 we only considered the linear position in the plane (of the plane quadrotor), and in Chapter 5 the linear position in space (of the space quadrotor). We now wish to take into account the whole output in the optimisation process, thus we introduce the

closed-loop output sensitivity

$$\boldsymbol{\Upsilon}(\mathrm{t}) = \left.\frac{\partial \mathbf{y}\left(\mathbf{q}(\mathrm{t})\right)}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_\mathrm{c}} \in \mathbb{R}^{\mathrm{n}_\mathbf{y} \times \mathrm{n}_\mathbf{p}} \tag{6.1}$$

which, like the other sensitivities, is evaluated at the nominal $\mathbf{p} = \mathbf{p}_\mathrm{c}$, along the closed-loop trajectories of Eqs. (4.1)-(4.2). This new metric can be easily obtained from the state sensitivity $\boldsymbol{\Pi}$, as $\mathbf{y}(\mathbf{q})$ is a function of the state. Hence, it is more appropriate to write $\boldsymbol{\Upsilon}$ as

$$\boldsymbol{\Upsilon}(\mathrm{t}) = \frac{\partial \mathbf{y}\left(\mathbf{q}\right)}{\partial \mathbf{q}} \cdot \frac{\partial \mathbf{q}(\mathrm{t})}{\partial \mathbf{p}} = \frac{\partial \mathbf{y}}{\partial \mathbf{q}} \cdot \boldsymbol{\Pi}(\mathrm{t}), \tag{6.2}$$

where the jacobian in front of $\boldsymbol{\Pi}$ is a new quantity to be computed: if a variable of the output is not contained in the state vector (*e.g.* for a space quadrotor oriented with unit quaternions, the yaw ψ is not present in the state, but is the last component of the output), then this jacobian contains the information that links it to the state vector $\mathbf{q}$; otherwise, the output is completely contained in the state, meaning that the jacobian is trivial, and simply selects the appropriate rows of $\boldsymbol{\Pi}(\mathrm{t})$. As for the state sensitivity, we recall that generating an optimised reference $\mathbf{y}_\mathrm{d}(\mathbf{a}^\star,\, \mathrm{t})$ for which some norm of $\boldsymbol{\Upsilon}(\mathrm{t})$ is minimised should render the closed-loop output $\mathbf{y}(\mathrm{t})$ as close as possible to its nominal $\mathbf{y}_\mathrm{c}(\mathrm{t})$, and result in a statistically enhanced performance of the system.

The evaluation of the state/input/output sensitivities is done by forward integration of

$$\begin{cases} \boldsymbol{\Pi}(0) = \boldsymbol{\Pi}_0 = \mathbf{0}_{\mathrm{n}_\mathbf{q} \times \mathrm{n}_\mathbf{p}} \\ \boldsymbol{\Pi}_{\boldsymbol{\xi}}(0) = \boldsymbol{\Pi}_{\boldsymbol{\xi}_0} = \mathbf{0}_{\mathrm{n}_{\boldsymbol{\xi}} \times \mathrm{n}_\mathbf{p}} \\ \dot{\boldsymbol{\Pi}}(\mathrm{t}) = \mathbf{f}_{,\mathbf{q}} \cdot \boldsymbol{\Pi} + \mathbf{f}_{,\mathbf{u}} \cdot \boldsymbol{\Theta} + \mathbf{f}_{,\mathbf{p}} \\ \dot{\boldsymbol{\Pi}}_{\boldsymbol{\xi}}(\mathrm{t}) = \mathbf{g}_{,\mathbf{q}} \cdot \boldsymbol{\Pi} + \mathbf{g}_{,\boldsymbol{\xi}} \boldsymbol{\Pi}_{\boldsymbol{\xi}} \\ \boldsymbol{\Theta}(\mathrm{t}) = \mathbf{h}_{,\mathbf{q}} \cdot \boldsymbol{\Pi} + \mathbf{h}_{,\boldsymbol{\xi}} \boldsymbol{\Pi}_{\boldsymbol{\xi}} \\ \boldsymbol{\Upsilon}(\mathrm{t}) = \mathbf{y}_{,\mathbf{q}} \cdot \boldsymbol{\Pi} \end{cases}, \tag{6.3}$$

along the reference trajectory $\mathbf{y}_\mathrm{d}(\mathbf{a},\, \mathrm{t})$, and for the chosen robot/controller pair of Eqs. (4.1)-(4.2). From these metrics, one can formulate an optimisation problem under constraints, *e.g.* limit conditions and actuator limits, that aims at minimising a norm of $\boldsymbol{\Pi}$ and/or $\boldsymbol{\Upsilon}$, and outputs the optimal shape parameter $\mathbf{a}^\star$. This way, the tracking of the optimised reference $\mathbf{y}_\mathrm{d}(\mathbf{a}^\star,\, \mathrm{t})$ will be *intrinsically robust* w.r.t. uncertainties in the parameters $\mathbf{p}$, at least in the neighbourhood of

the nominal $\mathbf{p}_\mathrm{c}$.

6.3 Evaluation of uncertainty tubes

We now discuss how one can exploit the notions of state/input/output sensitivities for evaluating the uncertainty tubes from a given model of parametric uncertainty. The theory presented in this section is detailed at a specific time point $\mathrm{t} \in \mathbb{T} = [0,\ \mathrm{T}]$. Of course, when applying this theory for all time points, we obtain the desired *uncertainty tubes* along the whole trajectory.

6.3.1 State, input and output ellipsoids

Let us assume that each parameter $\mathrm{p}_{i \in [\![1,\, \mathrm{n}_\mathbf{p}]\!]}$ can take values in a given interval defined by the range $\delta\mathrm{p}_i \in \mathbb{R}$, centred at the nominal $\mathrm{p}_{\mathrm{c}_i}$, defined by

$$\forall i \in [\![1,\, \mathrm{n}_\mathbf{p}]\!],\ \ \mathrm{p}_i \in [\mathrm{p}_{\mathrm{c}_i} - \delta\mathrm{p}_i,\, \mathrm{p}_{\mathrm{c}_i} + \delta\mathrm{p}_i]\,. \tag{6.4}$$

For instance, these uncertainty ranges can be defined if we know the measurement capabilities (and therefore the uncertainties) of the different parameters involved in our dynamic model. We then use these ranges to define the diagonal weight matrix

$$\mathbf{W}_\delta = \begin{bmatrix} \delta^2_{\mathrm{p}_1} & & \\ & \ddots & \\ & & \delta^2_{\mathrm{p}_{\mathrm{n}_\mathbf{p}}} \end{bmatrix} \in \mathbb{R}^{\mathrm{n}_\mathbf{p} \times \mathrm{n}_\mathbf{p}}. \tag{6.5}$$

Now, letting $\Delta\mathbf{p} = \mathbf{p} - \mathbf{p}_\mathrm{c}$, the equation

$$\Delta\mathbf{p}^\mathsf{T} \cdot \mathbf{W}_\delta^{-1} \cdot \Delta\mathbf{p} = 1 \tag{6.6}$$

represents an ellipsoid in the parameters space, centred at the nominal $\mathbf{p}_\mathrm{c}$, and with semi-axes of lengths $\delta_{\mathrm{p}_i} \mid i \in [\![1,\, \mathrm{n}_\mathbf{p}]\!]$.

Then, by taking advantage of the closed-loop state sensitivity definition of Eq. (4.3), one can obtain the corresponding ellipsoid in the state space, by following a procedure similar to the derivation of the manipulability ellipsoid for robot manipulators, see, *e.g.*, [Sciavicco et al., 2001]. To this end, let $\Delta\mathbf{q}(\mathrm{t}) = \mathbf{q}(\mathrm{t}) - \mathbf{q}_\mathrm{c}(\mathrm{t})$, where $\mathbf{q}_\mathrm{c}(\mathrm{t})$ represents the nominal state evolution of the closed-loop system for the robot/controller of Eqs. (4.1)-(4.2), therefore at $\mathbf{p} = \mathbf{p}_\mathrm{c}$. For a

small enough $\Delta\mathbf{p}$, one has

$$\Delta\mathbf{q}(\mathrm{t}) \approx \mathbf{\Pi}(\mathrm{t}) \cdot \Delta\mathbf{p}. \tag{6.7}$$

We recall that as written just above, the discrepancy from $\mathbf{p}$ to $\mathbf{p}_\mathrm{c}$ has to remain small: indeed, as the closed-loop state sensitivity is a first-order jacobian of the state $\mathbf{q}$ w.r.t. the parameters $\mathbf{p}$, and is evaluated at the nominal $\mathbf{p}_\mathrm{c}$, if $\mathbf{p}$ deviates too much from $\mathbf{p}_\mathrm{c}$, it may be the case that the approximation of Eq. (6.7) becomes wrong.

Let us now invert Eq. (6.7), and plug it into Eq. (6.6), this yields

$$\Delta\mathbf{q}^\mathsf{T}(\mathrm{t}) \cdot \mathbf{\Pi}^{+\mathsf{T}}(\mathrm{t}) \cdot \mathbf{W}_\delta^{-1} \cdot \mathbf{\Pi}^{+}(\mathrm{t}) \cdot \Delta\mathbf{q}(\mathrm{t}) = 1, \tag{6.8a}$$

$$\text{(6.8a)} \Longleftrightarrow \Delta\mathbf{q}^\mathsf{T}(\mathrm{t}) \cdot \left[\mathbf{\Pi}(\mathrm{t}) \cdot \mathbf{W}_\delta \cdot \mathbf{\Pi}^\mathsf{T}(\mathrm{t})\right]^{-1} \cdot \Delta\mathbf{q}(\mathrm{t}) = 1, \tag{6.8b}$$

where $\mathbf{M}^+$ denotes the Moore–Penrose generalised inverse of matrix $\mathbf{M} \in \mathbb{R}^{m\times n}$, see [Penrose, 1955]. Eqs. (6.8a) and (6.8b) represent an ellipsoid in state space, centred at $\mathbf{q}_\mathrm{c}(\mathrm{t})$, and with a symmetric and positive semi-definite state sensitivity kernel matrix

$$\mathbf{K}_{\mathbf{\Pi},\delta}(\mathrm{t}) = \mathbf{\Pi}(\mathrm{t}) \cdot \mathbf{W}_\delta \cdot \mathbf{\Pi}^\mathsf{T}(\mathrm{t}) \in \mathbb{R}^{\mathrm{n_q}\times\mathrm{n_q}}. \tag{6.9}$$

We then inject Eq. (6.9) into Eq. (6.8b), to rewrite the ellipsoid in terms of $\mathbf{K}_{\mathbf{\Pi},\delta}(\mathrm{t})$, that yields

$$\text{(6.8b)} \Longleftrightarrow \Delta\mathbf{q}^\mathsf{T}(\mathrm{t}) \cdot \mathbf{K}_{\mathbf{\Pi},\delta}^{-1}(\mathrm{t}) \cdot \Delta\mathbf{q}(\mathrm{t}) = 1. \tag{6.10}$$

Let $\Delta\mathbf{u}(\mathrm{t}) = \mathbf{u}(\mathrm{t}) - \mathbf{u}_\mathrm{c}(\mathrm{t})$ and $\Delta\mathbf{y}(\mathrm{t}) = \mathbf{y}(\mathrm{t}) - \mathbf{y}_\mathrm{c}(\mathrm{t})$, where $\mathbf{u}_\mathrm{c}(\mathrm{t})$ and $\mathbf{y}_\mathrm{c}(\mathrm{t})$ represent respectively the nominal input and output evolutions of the closed-loop system. We take advantage of the closed-loop input sensitivity definition of Eq. (4.4), and closed-loop output sensitivity definition of Eq. (6.1), to define the symmetric and positive semi-definite input and output sensitivities kernel matrices

$$\mathbf{K}_{\mathbf{\Theta},\delta}(\mathrm{t}) = \mathbf{\Theta}(\mathrm{t}) \cdot \mathbf{W}_\delta \cdot \mathbf{\Theta}^\mathsf{T}(\mathrm{t}) \in \mathbb{R}^{\mathrm{n_u}\times\mathrm{n_u}}, \tag{6.11}$$

and

$$\mathbf{K}_{\mathbf{\Upsilon},\delta}(\mathrm{t}) = \mathbf{y}(\mathrm{t}) \cdot \mathbf{W}_\delta \cdot \mathbf{y}^\mathsf{T}(\mathrm{t}) \in \mathbb{R}^{\mathrm{n_y}\times\mathrm{n_y}}, \tag{6.12}$$

and use analogous arguments to those that allowed us to write Eqs. (6.4)-(6.10), but replace $\Delta\mathbf{q}(\mathrm{t})$ by $\Delta\mathbf{u}(\mathrm{t})$ and $\Delta\mathbf{y}(\mathrm{t})$, $\mathbf{\Pi}(\mathrm{t})$ by $\mathbf{\Theta}(\mathrm{t})$ and $\mathbf{\Upsilon}(\mathrm{t})$, and $\mathbf{K}_{\mathbf{\Pi},\delta}(\mathrm{t})$ by $\mathbf{K}_{\mathbf{\Theta},\delta}(\mathrm{t})$ and $\mathbf{K}_{\mathbf{\Upsilon},\delta}(\mathrm{t})$.

Therefore, we also obtain two ellipsoids, in the input space

$$\Delta\mathbf{u}^{\mathsf{T}}(\mathrm{t}) \cdot \mathbf{K}^{-1}_{\mathbf{\Theta},\delta}(\mathrm{t}) \cdot \Delta\mathbf{u}(\mathrm{t}) = 1 \tag{6.13}$$

and output space

$$\Delta\mathbf{y}^{\mathsf{T}}(\mathrm{t}) \cdot \mathbf{K}^{-1}_{\mathbf{\Upsilon},\delta}(\mathrm{t}) \cdot \Delta\mathbf{y}(\mathrm{t}) = 1 \tag{6.14}$$

respectively.

The ellipsoids of Eq. (6.10), Eq. (6.13) and Eq. (6.14) represent the tube, or envelope, of respectively the most perturbed state, input and output trajectories. Indeed, if the parameter $\mathbf{p}$ lie inside the volume of the ellipsoid of Eq. (6.6), with semi-axes of lengths $\delta \mathrm{p}_i \mid i \in [\![1,\, \mathrm{n}_{\mathbf{p}}]\!]$, the state $\mathbf{q}(\mathrm{t})$ will lie inside the ellipsoid of Eq. (6.10), whose semi-axes are of lengths $\sqrt{\lambda_{\mathbf{K}_{\mathbf{\Pi}},i}(\mathrm{t})} \mid i \in [\![1,\, \mathrm{n}_{\mathbf{q}}]\!]$, where $\lambda_{\mathbf{K}_{\mathbf{\Pi}},i}(\mathrm{t}) \in \mathbb{R}$ are the eigenvalues of $\mathbf{K}_{\mathbf{\Pi},\delta}(\mathrm{t})$. Analogously, for the same conditions on $\mathbf{p}$, the input/output remain inside the ellipsoids of Eq. (6.13) and Eq. (6.14) respectively.

Therefore, the eigenvalues $\lambda_{\mathbf{K}_{\mathbf{\Pi}},i}$ of $\mathbf{K}_{\mathbf{\Pi},\delta}$, $\lambda_{\mathbf{K}_{\mathbf{\Theta}},i}$ of $\mathbf{K}_{\mathbf{\Theta},\delta}$ and $\lambda_{\mathbf{K}_{\mathbf{\Upsilon}},i}$ of $\mathbf{K}_{\mathbf{\Upsilon},\delta}$, quantify respectively the largest state, input and output deviations, along the principal axes of the corresponding ellipsoids, see, Eq. (6.10), Eq. (6.13) and Eq. (6.14). However, one may also be interested in obtaining the largest deviation in the state/input/output space, along any direction of interest, thus not necessarily aligned with one of the principal axes of the different ellipsoids.

6.3.2 Largest deviation along any direction

We now detail a general procedure for addressing this latter point, of computing the largest deviation in the state/input space, along any direction of potential interest. Let us consider, for the sake of illustration, the state space ellipsoid of Eq. (6.10) (the input/output cases are equivalent), and let $\mathbf{n} \in \mathbb{R}^{\mathrm{n}_{\mathbf{q}}}$ be a unit-norm direction in the state space. We are interested in finding the radius $\mathrm{r}_{\mathbf{n}}(\mathrm{t}) \in \mathbb{R}^{+}$ of the ellipsoid of Eq. (6.10), which allows to obtain the bound for the perturbed states along the given direction $\mathbf{n}$

$$\mathbf{n}^{\mathsf{T}} \cdot \mathbf{q}(\mathrm{t}) \in \left[\mathbf{n}^{\mathsf{T}} \cdot \mathbf{q}_{\mathbf{p}_{\mathrm{c}}}(\mathrm{t}) - \mathrm{r}_{\mathbf{n}}(\mathrm{t}),\, \mathbf{n}^{\mathsf{T}} \cdot \mathbf{q}_{\mathbf{p}_{\mathrm{c}}}(\mathrm{t}) + \mathrm{r}_{\mathbf{n}}(\mathrm{t})\right]. \tag{6.15}$$

We recall the reader that in Eq. (6.15), $\mathbf{n}^{\mathsf{T}} \cdot \mathbf{q}(\mathrm{t}) = \langle \mathbf{n}\ ,\ \mathbf{q}(\mathrm{t}) \rangle$ is a scalar product.

As illustrated later in Section 6.4, this can be relevant when, for instance, $\mathbf{n}$ represents one of the state-space Cartesian directions, so that Eq. (6.15) can be used to bound individual components of the state vector $\mathbf{q}(\mathrm{t})$.

Plugging vector $\mathrm{r}_{\mathbf{n}}(\mathrm{t})\mathbf{n}$ in Eq. (6.10) results in

$$\mathrm{r}_{\mathbf{n}}^2(\mathrm{t})\mathbf{n}^{\mathsf{T}} \cdot \mathbf{K}_{\mathbf{\Pi},\delta}^{-1}(\mathrm{t}) \cdot \mathbf{n} = 1 \Longrightarrow \mathrm{r}_{\mathbf{n}}(\mathrm{t}) = \frac{1}{\sqrt{\mathbf{n}^{\mathsf{T}} \cdot \mathbf{K}_{\mathbf{\Pi},\delta}^{-1}(\mathrm{t}) \cdot \mathbf{n}}}. \tag{6.16}$$

Eq. (6.16) is, unfortunately, valid only if the direction $\mathbf{n}$ belongs to the range space of the kernel matrix $\mathbf{K}_{\mathbf{\Pi},\delta}(\mathrm{t})$ (we recall that $\mathbf{K}_{\mathbf{\Pi},\delta}(\mathrm{t})$ and $\mathbf{K}_{\mathbf{\Pi},\delta}^{-1}(\mathrm{t})$ have the same range space). This is never an issue if $\mathbf{K}_{\mathbf{\Pi},\delta}(\mathrm{t})$ is full rank but, in our context, $\mathbf{K}_{\mathbf{\Pi},\delta}(\mathrm{t})$ can easily be rank-deficient. For example, if the number of states is larger than the number of parameters, *i.e.* $\mathrm{n}_{\mathbf{q}} > \mathrm{n}_{\mathbf{p}}$, then the state sensitivity $\mathbf{\Pi}(\mathrm{t})$ is rectangular high and, thus, $\mathbf{K}_{\mathbf{\Pi},\delta}(\mathrm{t}) = \mathbf{\Pi}(\mathrm{t}) \cdot \mathbf{W}_{\delta} \cdot \mathbf{\Pi}^{\mathsf{T}}(\mathrm{t})$ results rank deficient by construction. Consequently, we now have to discuss a generalisation of Eq. (6.16), that remains valid also when the direction $\mathbf{n}$ does not belong to the range of $\mathbf{K}_{\mathbf{\Pi},\delta}(\mathrm{t})$.

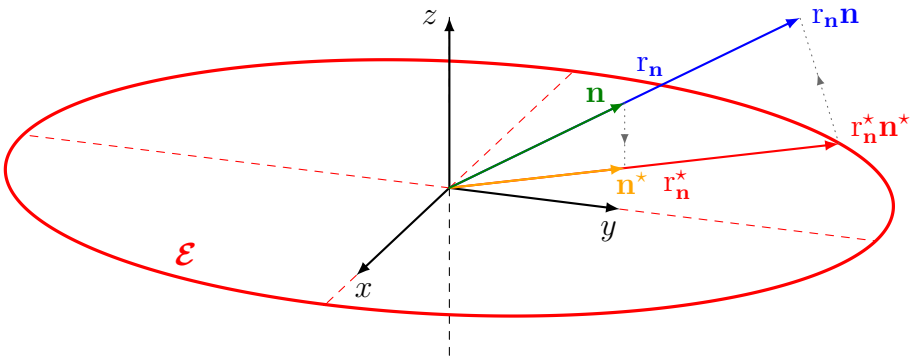

Figure 6.1 – Illustration of how to obtain the radius $\mathrm{r}_{\mathbf{n}}(\mathrm{t})$ along any direction $\mathbf{n}$.

As illustration, one could think of this analogy: let $\mathbf{K}_{\mathbf{\Pi},\delta}(\mathrm{t})$ be a 3×3 matrix with rank 2, thus representing a 2-dimensional ellipse $\boldsymbol{\mathcal{E}}$, oriented arbitrarily in a 3-dimensional space. The notion of radius $\mathrm{r}_{\mathbf{n}}$ of the 2-dimensional ellipse along the direction $\mathbf{n}$ is well-defined only if $\mathbf{n}$ to the plane of $\boldsymbol{\mathcal{E}}$. Otherwise, see Fig. 6.1, one should:

(i) project the direction $\mathbf{n}$ onto the plane of the 2-dimensional ellipse $\boldsymbol{\mathcal{E}}$, in order to obtain a *feasible* direction $\mathbf{n}^{\star}(\mathrm{t})$;

(ii) compute the radius $\mathrm{r}_{\mathbf{n}}^{\star}(\mathrm{t})$ of the 2-dimensional ellipse $\boldsymbol{\mathcal{E}}$, along the feasible direction $\mathbf{n}^{\star}(\mathrm{t})$;

(iii) re-project orthogonally the vector $\mathrm{r}_{\mathbf{n}}^{\star}(\mathrm{t})\mathbf{n}^{\star}(\mathrm{t})$ on $\mathbf{n}$, in order to obtain the resulting length $\mathrm{r}_{\mathbf{n}}(\mathrm{t})$ along the original direction.

Finally, this procedure should, of course, yield the same result as Eq. (6.16) when $\mathbf{K}_{\mathbf{\Pi},\delta}(\mathrm{t})$ is full rank, or when $\mathbf{n}$ happens to belong to the range space of $\mathbf{K}_{\mathbf{\Pi},\delta}(\mathrm{t})$.

We now detail how to implement this procedure, in the general case. Let

$$\mathbf{U}(\mathrm{t}) \cdot \mathbf{\Lambda}(\mathrm{t}) \cdot \mathbf{U}^{\mathsf{T}}(\mathrm{t}) = \mathbf{K}_{\mathbf{\Pi},\,\delta}(\mathrm{t}) \tag{6.17}$$

be the eigenvector decomposition of $\mathbf{K}_{\mathbf{\Pi},\,\delta}(\mathrm{t})$, with

$$\mathbf{\Lambda}(\mathrm{t}) = \begin{bmatrix} \lambda_{\mathbf{K}_{\mathbf{\Pi}},\,1}(\mathrm{t}) & & \\ & \ddots & \\ & & \lambda_{\mathbf{K}_{\mathbf{\Pi}},\,\mathrm{n}_{\mathbf{q}}}(\mathrm{t}) \end{bmatrix} \in \mathbb{R}^{\mathrm{n}_{\mathbf{q}} \times \mathrm{n}_{\mathbf{q}}}. \tag{6.18}$$

We emphasise that in the general case, some $\lambda_{\mathbf{K}_{\mathbf{\Pi}},\,i}(\mathrm{t})$ could be zero, if $\mathbf{K}_{\mathbf{\Pi},\,\delta}(\mathrm{t})$ is not full rank. Therefore, the generalised inverse of $\mathbf{K}_{\mathbf{\Pi},\,\delta}(\mathrm{t})$ is

$$\mathbf{K}^{\dagger}_{\mathbf{\Pi},\,\delta}(\mathrm{t}) = \mathbf{U}(\mathrm{t}) \cdot \mathbf{\Lambda}^{\dagger}(\mathrm{t}) \cdot \mathbf{U}^{\mathsf{T}}(\mathrm{t}), \tag{6.19}$$

in which

$$\mathbf{\Lambda}^{\dagger}(\mathrm{t}) = \begin{bmatrix} \ell_1(\mathrm{t}) & & \\ & \ddots & \\ & & \ell_{\mathrm{n}_{\mathbf{q}}}(\mathrm{t}) \end{bmatrix} \in \mathbb{R}^{\mathrm{n}_{\mathbf{q}} \times \mathrm{n}_{\mathbf{q}}}, \Bigg| \begin{cases} \ell_i(\mathrm{t}) = \dfrac{1}{\lambda_{\mathbf{K}_{\mathbf{\Pi}},\,i}(\mathrm{t})} & \text{if } \lambda_{\mathbf{K}_{\mathbf{\Pi}},\,i}(\mathrm{t}) > \varepsilon, \\ \ell_i(\mathrm{t}) = 0 & \text{otherwise} \end{cases}, \tag{6.20}$$

where ε is a user-defined small threshold, for deciding whether an eigenvalue is considered null or not. Now, let $\mathrm{d}_{\lambda}(\mathrm{t}) \in [\![1,\, \mathrm{n}_{\mathbf{q}}]\!]$ be the number of non-zero $\lambda_{\mathbf{K}_{\mathbf{\Pi}},\,i}(\mathrm{t})$. We exclude the case where $\mathrm{d}_{\lambda}(\mathrm{t}) = 0$, since it would mean that $\mathbf{K}^{\dagger}_{\mathbf{\Pi},\,\delta}(\mathrm{t}) = \mathbf{0}_{\mathrm{n}_{\mathbf{q}}}$. Also, note that as $\mathrm{d}_{\lambda}(\mathrm{t})$ is a function of $\mathbf{\Pi}(\mathrm{t})$, it is time-dependant, *i.e.* at each time $\mathrm{t} \in \mathbb{T}$, by construction, the number of non-zero eigenvalues can change. Then, let $\mathbf{v}_{\lambda,\,i}(\mathrm{t}) \in \mathbb{R}^{\mathrm{n}_{\mathbf{q}}}$ be the eigenvector that corresponds to the non-zero eigenvalue $\lambda_{\mathbf{K}_{\mathbf{\Pi}},\,i}(\mathrm{t})$, that is the i-th column of $\mathbf{U}(\mathrm{t})$, see Eqs. (6.17) and (6.19). Hence, the d_{λ}-dimensional range space of $\mathbf{K}_{\mathbf{\Pi},\,\delta}(\mathrm{t})$ is spanned by the vectors $\left\{\mathbf{v}_{\lambda,\,1}(\mathrm{t}),\ \ \ldots\ \ ,\, \mathbf{v}_{\lambda,\,\mathrm{d}_{\lambda}}(\mathrm{t})\right\}$, and thus

$$\mathbf{P}_{\perp}(\mathrm{t}) = \begin{bmatrix} \mathbf{v}_{\lambda,\,1}(\mathrm{t}) & \cdots & \mathbf{v}_{\lambda,\,\mathrm{d}_{\lambda}}(\mathrm{t}) \end{bmatrix} \cdot \begin{bmatrix} \mathbf{v}_{\lambda,\,1}(\mathrm{t}) & \cdots & \mathbf{v}_{\lambda,\,\mathrm{d}_{\lambda}}(\mathrm{t}) \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^{\mathrm{n}_{\mathbf{q}} \times \mathrm{n}_{\mathbf{q}}} \tag{6.21}$$

is an orthogonal projector onto this range space, such that $\mathbf{P}_{\perp}(\mathrm{t}) = \mathbf{P}^{\mathsf{T}}_{\perp}(\mathrm{t})$, and also projecting

twice is the same as projecting once, *i.e.* $\mathbf{P}_{\perp}(\mathrm{t}) \cdot \mathbf{P}_{\perp}(\mathrm{t}) = \mathbf{P}_{\perp}(\mathrm{t})$. We can then define

$$\mathbf{n}^{\star}(\mathrm{t}) = \mathbf{P}_{\perp}(\mathrm{t}) \cdot \mathbf{n} \in \mathbb{R}^{\mathrm{n_q}} \tag{6.22}$$

as the orthogonal projection of the direction $\mathbf{n}$ onto the range of $\mathbf{K}_{\mathbf{\Pi},\,\delta}(\mathrm{t})$, which allows using Eq. (6.16) to obtain the radius $\mathrm{r}^{\star}_{\mathbf{n}}(\mathrm{t})$ of the ellipsoid along the feasible direction $\mathbf{n}^{\star}(\mathrm{t})$, that is

$$\mathrm{r}^{\star}_{\mathbf{n}}(\mathrm{t}) = \frac{1}{\sqrt{\mathbf{n}^{\star\mathsf{T}}(\mathrm{t}) \cdot \mathbf{K}^{\dagger}_{\mathbf{\Pi},\,\delta}(\mathrm{t}) \cdot \mathbf{n}^{\star}(\mathrm{t})}}. \tag{6.23}$$

Since $\mathbf{K}^{\dagger}_{\mathbf{\Pi},\,\delta}(\mathrm{t})$ can be expanded from Eq. (6.19) as

$$\mathbf{K}^{\dagger}_{\mathbf{\Pi},\,\delta}(\mathrm{t}) = \sum_{i=1}^{\mathrm{d}_{\lambda}(t)} \frac{\mathbf{v}_{\lambda,\,i}(\mathrm{t}) \cdot \mathbf{v}^{\mathsf{T}}_{\lambda,\,i}(\mathrm{t})}{\lambda_{\mathbf{K}_{\mathbf{\Pi}},\,i}(\mathrm{t})}, \tag{6.24}$$

it follows that

$$\mathbf{n}^{\star\mathsf{T}}(\mathrm{t}) \cdot \mathbf{K}^{\dagger}_{\mathbf{\Pi},\,\delta}(\mathrm{t}) \cdot \mathbf{n}^{\star}(\mathrm{t}) = \sum_{i=1}^{\mathrm{d}_{\lambda}(t)} \frac{\left[\mathbf{v}^{\mathsf{T}}_{\lambda,\,i}(\mathrm{t}) \cdot \mathbf{n}^{\star}(\mathrm{t})\right]^2}{\lambda_{\mathbf{K}_{\mathbf{\Pi}},\,i}(\mathrm{t})} \tag{6.25a}$$

$$\text{(6.25a)} \iff \mathbf{n}^{\star\mathsf{T}}(\mathrm{t}) \cdot \mathbf{K}^{\dagger}_{\mathbf{\Pi},\,\delta}(\mathrm{t}) \cdot \mathbf{n}^{\star}(\mathrm{t}) = \sum_{i=1}^{\mathrm{d}_{\lambda}(t)} \frac{\left[\mathbf{v}^{\mathsf{T}}_{\lambda,\,i}(\mathrm{t}) \cdot \mathbf{P}_{\perp}(\mathrm{t}) \cdot \mathbf{n}\right]^2}{\lambda_{\mathbf{K}_{\mathbf{\Pi}},\,i}(\mathrm{t})} \tag{6.25b}$$

$$\text{(6.25b)} \iff \mathbf{n}^{\star\mathsf{T}}(\mathrm{t}) \cdot \mathbf{K}^{\dagger}_{\mathbf{\Pi},\,\delta}(\mathrm{t}) \cdot \mathbf{n}^{\star}(\mathrm{t}) = \sum_{i=1}^{\mathrm{d}_{\lambda}(t)} \frac{\left[\mathbf{v}^{\mathsf{T}}_{\lambda,\,i}(\mathrm{t}) \cdot \mathbf{n}\right]^2}{\lambda_{\mathbf{K}_{\mathbf{\Pi}},\,i}(\mathrm{t})}, \tag{6.25c}$$

where we used the fact that $\mathbf{P}_{\perp}(\mathrm{t}) \cdot \mathbf{v}_{\lambda,\,i}(\mathrm{t}) = \mathbf{v}_{\lambda,\,i}(\mathrm{t})$, since by construction, $\mathbf{v}_{\lambda,\,i}(\mathrm{t})$ belongs to the range space of $\mathbf{K}_{\mathbf{\Pi},\,\delta}(\mathrm{t})$. Therefore, we finally have that

$$\mathrm{r}^{\star}_{\mathbf{n}}(\mathrm{t}) = \frac{1}{\sqrt{\sum_{i=1}^{\mathrm{d}_{\lambda}(t)} \frac{\left[\mathbf{v}^{\mathsf{T}}_{\lambda,\,i}(\mathrm{t}) \cdot \mathbf{n}\right]^2}{\lambda_{\mathbf{K}_{\mathbf{\Pi}},\,i}(\mathrm{t})}}}. \tag{6.26}$$

The last step consists of obtaining the corresponding length $\mathrm{r}_{\mathbf{n}}(\mathrm{t})$ along the original direction

$\mathbf{n}$, which can be done by projecting vector $\mathrm{r}^{\star}_{\mathbf{n}}(\mathrm{t})\mathbf{n}$ onto $\mathbf{n}$, *i.e.*,

$$\mathrm{r}_{\mathbf{n}}(\mathrm{t}) = \mathrm{r}^{\star}_{\mathbf{n}}(\mathrm{t})\mathbf{n}^{\star\mathsf{T}}(\mathrm{t}) \cdot \mathbf{n} \tag{6.27a}$$

$$(6.27a) \iff \mathrm{r}_{\mathbf{n}}(\mathrm{t}) = \mathrm{r}^{\star}_{\mathbf{n}}(\mathrm{t})\mathbf{n}^{\mathsf{T}} \cdot \mathbf{P}_{\perp}(\mathrm{t}) \cdot \mathbf{n} \tag{6.27b}$$

$$(6.27b) \iff \mathrm{r}_{\mathbf{n}}(\mathrm{t}) = \frac{\mathbf{n}^{\mathsf{T}} \cdot \mathbf{P}_{\perp}(\mathrm{t}) \cdot \mathbf{n}}{\sqrt{\displaystyle\sum_{i=1}^{\mathrm{d}_{\lambda}(t)} \frac{\left[\mathbf{v}^{\mathsf{T}}_{\lambda,\,i}(\mathrm{t}) \cdot \mathbf{n}\right]^2}{\lambda_{\mathbf{K}_{\mathbf{\Pi}},\,i}(\mathrm{t})}}}. \tag{6.27c}$$

Therefore, Eq. (6.27c) is a generalisation of Eq. (6.16), for the arbitrary unit-norm direction $\mathbf{n}$ in the state space, which does not necessarily belong to the range of $\mathbf{K}_{\mathbf{\Pi},\,\delta}(\mathrm{t})$. Furthermore, we highlight that:

— if $\mathbf{K}_{\mathbf{\Pi},\,\delta}(\mathrm{t})$ is full rank, then the projector $\mathbf{P}_{\perp}(\mathrm{t})$ of Eq. (6.21) is just the identity matrix, $\mathbf{P}_{\perp}(\mathrm{t}) = \mathbf{I}_{\mathrm{n}_{\mathbf{q}}}$;

— if $\mathbf{K}_{\mathbf{\Pi},\,\delta}(\mathrm{t})$ is not full rank, but the direction $\mathbf{n}$ happens to belong to its range space, then $\mathbf{P}_{\perp}(\mathrm{t}) \cdot \mathbf{n} = \mathbf{n}$.

As such, in both of these cases, Eq. (6.27c) reduces to Eq. (6.16), as expected.

We now briefly recall the main steps of the preceding theory, to maximise the ease of understanding. The method exploits the sensitivities definitions, in order to construct the maximum *uncertainty tubes* $\mathrm{r}_{\mathbf{n}}(\mathrm{t})$ of the state/input/output, along a reference trajectory. The complete procedure for the state (starting at $\mathrm{t} = 0$) can be summarised by the following points:

(i) consider a possible range of deviation $\delta\mathrm{p}_i$ for each uncertain parameter p_i, see Eq. (6.4);

(ii) construct the associated weight matrix $\mathbf{W}_{\delta}$ of all ranges, see Eq. (6.5);

(iii) let $\Delta\mathbf{p} = \mathbf{p} - \mathbf{p}_{\mathrm{c}}$, and construct the ellipsoid of the largest possible deviation on $\mathbf{p}$ (thus in the parameter space), centred at $\mathbf{p}_{\mathrm{c}}$, see Eq. (6.6);

(iv) let $\Delta\mathbf{q}(\mathrm{t}) = \mathbf{q}(\mathrm{t}) - \mathbf{q}_{\mathrm{c}}(\mathrm{t})$, and then exploit the state sensitivity definition, see Eq. (4.3), such that for small enough $\Delta\mathbf{q}(\mathrm{t})$ one can construct the corresponding ellipsoid of the largest possible deviation on $\mathbf{q}(\mathrm{t})$ (thus in the state space), centred at $\mathbf{q}_{\mathrm{c}}(\mathrm{t})$, see Eq. (6.10);

(v) therefore, if the parameter $\mathbf{p}$ lies into the parameter ellipsoid of Eq. (6.6), then the state $\mathbf{q}(\mathrm{t})$ lies within the state ellipsoid of Eq. (6.10);

(vi) as one might want to compute the largest deviations along any direction $\mathbf{n}$ of interest, and not only along the principal axes of the state ellipsoid, Eq. (6.27c) gives a generalisation of Eq. (6.16), to compute the radius $\mathrm{r}_{\mathbf{n}}(\mathrm{t})$ along this direction.

(vii) one can now proceed through all these steps, for the state $\mathbf{q}(\mathrm{t})$, for the input $\mathbf{u}(\mathrm{t})$ and for the output $\mathbf{y}(\mathrm{t})$, from $\mathrm{t} = 0$ to $\mathrm{t} = \mathrm{T}$, so as to obtain the complete evolution of the maximum state/input/output uncertainty tubes $\mathrm{r}_{\mathbf{n}}(\mathrm{t})$, along any direction $\mathbf{n}$.

Thanks to this theory, and by exploiting the ranges of parametric uncertainties as well as the state/input/output sensitivities, we now have the capacity to compute the worst deviations that might occur to the state/input/output (given that the parameters remain in their ranges of deviations), along a given reference trajectory. With that in mind, we further seek to provide an upgraded version of our sensitivity minimisation problems.

6.4 A novel trajectory optimisation problem

In this section, we discuss how one can exploit the results of the previous section, in order to define a new optimisation problem for the planning of trajectories that are robust to parametric uncertainties. First, we recall the formulation used until then, and we continue with the definition of the new TO problem.

Reminders of Chapter 4, Section 4.4: robust TO problem.

In our previous works so far, *i.e.* [Robuffo Giordano et al., 2018] and Chapter 4, see Section 4.4, we have considered the following general formulation (modulo minor variations) for the state and input sensitivities TO problem

$$\mathbf{a}^{\star} = \arg\min_{\mathbf{a}\in\mathcal{A}} \quad \left(\mathrm{w}_1 \|\mathbf{\Pi}(\mathrm{T})\|_{\mathrm{F}}^2 + \mathrm{w}_2 \int_0^{\mathrm{T}} \|\mathbf{\Theta}(\tau)\|_{\mathrm{F}}^2 \, \mathrm{d}\tau \right) \quad \text{s.t.} \quad \begin{cases} \mathbf{M}_{\mathcal{C}} \cdot \mathbf{a} = \mathbf{b} \\ \mathbf{u}_{\min} \leqslant \mathbf{u}_{\mathrm{c}}(\mathrm{t}) \leqslant \mathbf{u}_{\max}, \quad \forall \mathrm{t} \in \mathbb{T} = [0,\, \mathrm{T}] \end{cases}, \tag{6.28}$$

where $\mathbf{a}$ designs the optimisation vector which shapes the reference trajectory to be tracked by the robot/controller pair, $\mathcal{A}$ is its admissible set, $(\mathrm{w}_1,\, \mathrm{w}_2) \in ([0,\, 1])^2$ are suitable weights (again, see the possible values for these in Section 4.4), the first line of constraints represents given initial/final values for the trajectory $\mathbf{y}_{\mathrm{d}}(\mathbf{a},\, \mathrm{t})$ and its derivatives (linear equality constraints), and the second line imposes bounds on the inputs (nonlinear inequality constraints). So far, we used the Frobenius matrix norm to define our TO

problems, *i.e.*

$$\forall \mathrm{t} \in \mathbb{T}, \quad \|\mathbf{\Pi}(\mathrm{t})\|_{\mathrm{F}} = \sqrt{\mathrm{Tr}\left(\mathbf{\Pi}^{\mathsf{T}}(\mathrm{t}) \cdot \mathbf{\Pi}(\mathrm{t})\right)} = \sqrt{\sum_{i=1}^{\mathrm{n_q}} \sum_{j=1}^{\mathrm{n_p}} \Pi(\mathrm{t})_{i,j}^2} \quad , \tag{6.29}$$

and analogously for $\|\mathbf{\Theta}(\mathrm{t})\|_{\mathrm{F}}$. The rationale of the chosen cost is to minimise a norm of the sensitivity of the state at the final time T, for accuracy in reaching a desired final state, and a norm of the sensitivity of the inputs over the whole trajectory, for minimising the deviations in the perturbed inputs.
We now discuss how one can exploit the results of the previous section, to define an improved TO algorithm, w.r.t. Eq. (6.28).

First of all, we note that the Frobenius norm of Eq. (6.29) does not take advantage of the possible prior knowledge of the uncertainty ranges δ_{p_i} for the parameter vector $\mathbf{p} \in \mathbb{R}^{\mathrm{n_P}}$, an information that is typically available in many cases. Instead, one can exploit these quantities for defining a more *informative* matrix norm: in particular, in this work, we have made the choice to replace the Frobenius norm with a norm of the kernel matrix $\mathbf{K}_{\mathbf{\Upsilon},\delta}(\mathrm{t})$ such as, for instance, its largest eigenvalue $\max\left(\lambda_{\mathbf{K}_{\mathbf{\Upsilon}},i}(\mathrm{t})\right) = \lambda_{\max,\mathbf{y}}(\mathrm{t}) \mid i \in [\![1,\, \mathrm{n_q}]\!]$, which represents the largest (worst-case) deviation in the output space, given the parametric uncertainty model of Eq. (6.4). We then define a norm of $\mathbf{\Upsilon}(\mathrm{t})$ as

$$\|\mathbf{y}(\mathrm{t})\|_{\delta} = \lambda_{\max,\mathbf{\Upsilon}}(\mathrm{t}) \tag{6.30a}$$

$$\text{Eq. (6.30a)} \iff \|\mathbf{\Upsilon}(\mathrm{t})\|_{\delta} \approx \sqrt[p]{\sum_{i=1}^{\mathrm{n_y}} \lambda_{\mathbf{K}_{\mathbf{\Upsilon}},i}^{p}(\mathrm{t})} \tag{6.30b}$$

where, for numerical reasons, the $\max(\bullet)$ operation is replaced by a smooth approximation, here, namely the p-norm, with a large enough p. As such, minimising $\|\mathbf{\Upsilon}(\mathrm{t})\|_{\delta}$ is equivalent to approximately minimising the length of the largest semi-axis of the ellipsoid of Eq. (6.14), which explicitly takes into account the uncertainty model, Eq. (6.4). Other metrics could have been considered, such as the trace $\mathrm{Tr}\left(\mathbf{K}_{\mathbf{\Upsilon},\delta}(\mathrm{t})\right) = \prod_{i=1}^{\mathrm{n_q}} \lambda_{\mathbf{K}_{\mathbf{\Upsilon}},i}(\mathrm{t})$, of the kernel matrix $\mathbf{K}_{\mathbf{\Upsilon},\delta}(\mathrm{t})$, that is proportional to the volume of the ellipsoid in the output space. However, this metric is not ideal, since it may happen that all the eigenvalues $\lambda_{\mathbf{K}_{\mathbf{\Upsilon}},i}(\mathrm{t})$ become very small while one remains large: consequently, this norm could decrease to very low values during the optimisation, while the bound in the direction of the (still) large $\lambda_{\mathbf{K}_{\mathbf{\Upsilon}},i}(\mathrm{t})$ could remain high, which does not

guarantee that the perturbed output $\mathbf{y}(\mathrm{t})$ is getting constrained in an enough shrinking space. On the contrary, the norm of Eq. (6.30b) does, since one $\lambda_{\mathbf{K}_{\boldsymbol{\Upsilon}},i}(\mathrm{t})$ remains large compared to the others, the optimisation should decrease it, until it get smaller than another. This ensures that the volume of the ellipsoid that bounds the perturbed state $\mathbf{y}(\mathrm{t})$ always decreases when the cost objective decreases, but in a more selective, hence preferred way, *i.e.* by reducing only the largest semi-axis of the ellipsoid.

Concerning the inputs, we can exploit Eq. (6.27c), so as to obtain the uncertainty bounds on each component of the inputs $\mathbf{u}(\mathrm{t})$, by letting the direction $\mathbf{n}$ be any of the Cartesian directions in $\mathbb{R}^{\mathrm{n_u}}$. Letting $\mathrm{r}_{\mathbf{n}_{\mathrm{u}_i}}(\mathrm{t}) \mid i \in [\![1,\, \mathrm{n}_{\mathbf{u}}]\!]$ be the radius obtained from Eq. (6.27c), for the i-th Cartesian direction, then, one has that the envelope of perturbed inputs lies in the interval $\Big[\mathrm{u}_{\mathrm{c},\,i}(\mathrm{t}) - \mathrm{r}_{\mathbf{n}_{\mathrm{u}_i}}(\mathrm{t}),\, \mathrm{u}_{\mathrm{c},\,i}(\mathrm{t}) + \mathrm{r}_{\mathbf{n}_{\mathrm{u}_i}}(\mathrm{t})\Big]$. This can be exploited to replace the input constraints in Eq. (6.28), by the following two constraints

$$\forall\,(i,\,\mathrm{t}) \in [\![1,\,\mathrm{n}_{\mathbf{u}}]\!] \times [0,\,\mathrm{T}] \begin{cases} \mathrm{u}_{\min} \leqslant \mathrm{u}_{\mathrm{c},\,i}(\mathrm{t}) - \mathrm{r}_{\mathbf{n}_{\mathrm{u}_i}}\left(\mathbf{K}_{\boldsymbol{\Theta},\,\delta}(\mathrm{t})\right) \\ \mathrm{u}_{\mathrm{c},\,i}(\mathrm{t}) + \mathrm{r}_{\mathbf{n}_{\mathrm{u}_i}}\left(\mathbf{K}_{\boldsymbol{\Theta},\,\delta}(\mathrm{t})\right) \leqslant \mathrm{u}_{\max} \end{cases}, \tag{6.31}$$

where we explicitly write that the computed radius $\mathrm{r}_{\mathbf{n}_{\mathrm{u}_i}}\left(\mathbf{K}_{\boldsymbol{\Theta},\,\delta}(\mathrm{t})\right)$ is a function of $\mathbf{K}_{\boldsymbol{\Theta},\,\delta}(\mathrm{t})$, therefore it is also a function of the input sensitivity $\boldsymbol{\Theta}(\mathrm{t})$, as well as the maximum ranges of deviation $\delta\mathrm{p}_i \mid i \in [\![1,\,\mathrm{n}_{\mathbf{p}}]\!]$. Here, $\mathrm{u}_{\mathrm{c},\,i}(\mathrm{t})$ denotes the i-th component of the nominal input vector $\mathbf{u}_{\mathrm{c}}(\mathrm{t})$, thus when $\mathbf{p} = \mathbf{p}_{\mathrm{c}}$. Eq. (6.31) formulates new input constraints, that are considering the worst case deviation of the parameter $\mathbf{p}$, when it is at the limit of the ellipsoid in the parameter space, Eq. (6.6). The advantage of this choice is that enforcing Eq. (6.31) will guarantee the *feasibility* of the tube of perturbed inputs for *any* value of the parameters within the ellipsoid of Eq. (6.6), whereas the original formulation, see Eq. (6.28), could only guarantee feasibility in the nominal/unperturbed case $\mathbf{p} = \mathbf{p}_{\mathrm{c}}$. Note that in Subsection 6.3.1, we highlighted the fact that since the state/input/output sensitivities are defined as the first order jacobians of the state/input w.r.t. the parameter, the approximation of Eq. (6.7) is only usable when $\Delta\mathbf{p}$ remains small enough. Otherwise, the approximation becomes wrong, and therefore the feasibility, even with the tubes of maximum perturbations, may be compromised.

We can now formulate a new optimisation problem with the constraints of the tubes of

perturbed inputs, that is

$$\mathbf{a}^{\star}_{\boldsymbol{\Upsilon}} = \arg\min_{\mathbf{a}\in\mathcal{A}} \quad \left(\mathrm{w}_1 \|\boldsymbol{\Upsilon}(\mathrm{T})\|_\delta + \mathrm{w}_2 \int_0^{\mathrm{T}} \|\boldsymbol{\Upsilon}(\tau)\|_\delta \, \mathrm{d}\tau \right)$$
$$\text{s.t.} \quad \begin{cases} \mathbf{C}_{\text{eq.}} = \mathbf{0}_{\mathrm{n}_{\mathbf{C}_{\text{eq.}}}} \\ \left.\begin{array}{l} \mathrm{u}_{\min} \leqslant \mathrm{u}_{\mathrm{c},i}(\mathrm{t}) - \mathrm{r}_{\mathbf{n}_{\mathrm{u}_i}}\left(\mathbf{K}_{\boldsymbol{\Theta},\delta}(\mathrm{t})\right) \\ \mathrm{u}_{\mathrm{c},i}(\mathrm{t}) + \mathrm{r}_{\mathbf{n}_{\mathrm{u}_i}}\left(\mathbf{K}_{\boldsymbol{\Theta},\delta}(\mathrm{t})\right) \leqslant \mathrm{u}_{\max} \end{array}\right\} \forall\, (i,\, \mathrm{t}) \in [\![1,\, \mathrm{n}_{\mathbf{u}}]\!] \times [0,\, \mathrm{T}] \end{cases} , \tag{6.32}$$

where we do not write explicitly $\mathbf{C}_{\text{eq.}}$ that might change depending on what controller is used: as seen in Chapter 5, when using the geometric controller, we had to relax the kinematic constraints at the end of the motion, to replace them by the target reach constraint, *i.e*. the system has to go to the target (and the desired might go a bit off the real target for this to happen).

In Eq. (6.32), we aim at minimising the tube of perturbed outputs $\mathbf{y}(\mathrm{T})$ at the final time T and/or during the trajectory (depending on the values chosen for the weights w_1 and w_2), with the (usual) initial/final constraints on the states and its derivatives, and the (new) feasibility constraints on the envelope of perturbed inputs. Note that, in this problem formulation, the input sensitivity $\boldsymbol{\Theta}(\mathrm{t})$ is now used in the constraints for evaluating the radius $\mathrm{r}_{\mathbf{n}_{\mathrm{u}_i}}\left(\mathbf{K}_{\boldsymbol{\Theta},\delta}(\mathrm{t})\right)$, for each component of the inputs. Consequently, this upgrades grants full room to any combination of the final/integral output sensitivity in the cost to be decreased. Of course, if relevant for a task, one could choose to replace the output sensitivity $\boldsymbol{\Upsilon}$ by the state sensitivity $\boldsymbol{\Pi}$, in the cost of Eq. (6.32). We also wish to emphasise that, conversely to Eq. (6.28), where we sought to minimise some norm of the input sensitivity, Eq. (6.32) does not consider this reduction at all, and could even raise it. Nonetheless, even if that occurs, the new constraints formulation ensures that the reference trajectory remains dynamically feasible.

6.5 3D quadrotor model

6.5.1 Dynamics with a shift in the centre of gravity

As case study for validating our approach we consider the tracking task for a space quadrotor, for which the CoG is not aligned with the geometric centre of the UAV. In this contribution, we build our space quadrotor model upon the one that was already detailed in Chapter 3, see Section 3.4. The state and inputs have already been defined there, thus we refer the reader to this section for more details.

We recall that by construction, a space quadrotor is a trivially under-actuated platform, since it has only four (non-tilted) propellers, while moving in a six DoFs space. In reality, it is pretty obvious that it must tilt in order to move in $\mathbf{x}_{\mathcal{W}}$ or $\mathbf{y}_{\mathcal{W}}$ directions, and can only hover if the pitch and roll angles are null. Therefore, the output $\mathbf{y}(\mathbf{q})$ consists of the linear position $\mathbf{r}$ and the heading/yaw angle ψ, thus $\mathbf{y}(\mathbf{q}) = \begin{bmatrix}\mathbf{r}^{\mathsf{T}} & \psi\end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^3 \times \mathbb{R}$. Since the orientation of the drone is not directly given by Euler angles, but rather by the quaternion $\boldsymbol{\rho}$, we recall the relation

$$\psi(\mathbf{q}) = \text{atan2}\left(2\rho_w\rho_z + 2\rho_x\rho_y,\ 2\rho_w^2 + 2\rho_x^2 - 1\right), \tag{6.33}$$

which is needed to get the jacobian in Eq. (6.2), and compute the closed-loop output sensitivity $\boldsymbol{\Upsilon}(\mathrm{t})$.

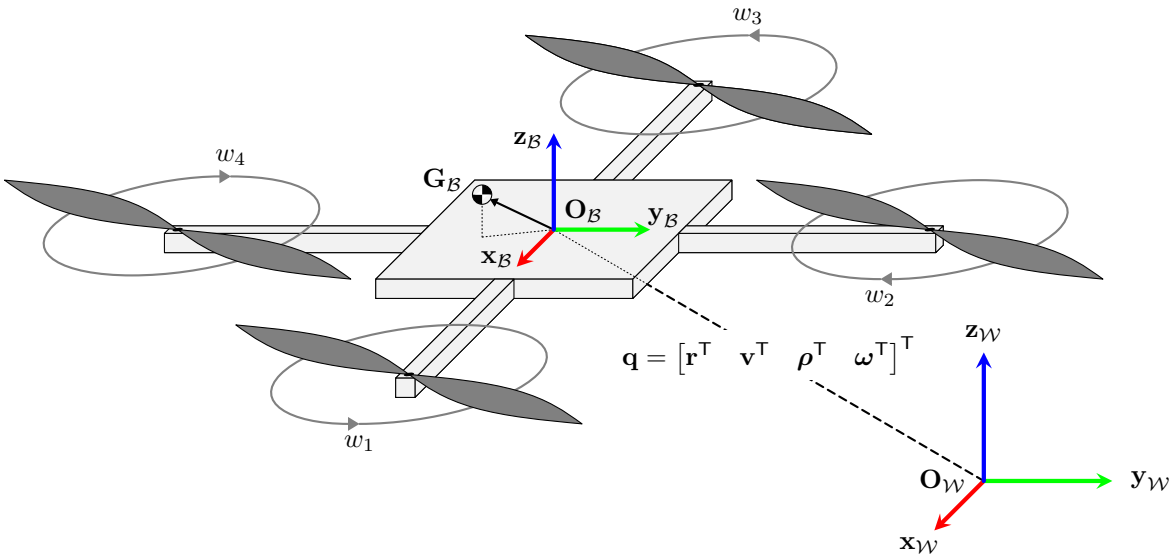

Figure 6.2 – Schematic view of the space quadrotor, oriented by quaternions. Note that in this contribution, there is a shift in the CoM, such that $\mathbf{O}_{\mathcal{B}}\mathbf{G}_{\mathcal{B}} = g_x\mathbf{x}_{\mathcal{B}} + g_y\mathbf{y}_{\mathcal{B}} + g_z\mathbf{z}_{\mathcal{B}}$.

In this contribution, we also consider that the CoG $\mathbf{G}_{\mathcal{B}}$ is not coincident with the geometric centre $\mathbf{O}_{\mathcal{B}}$, but it is displaced at an offset $\mathbf{r}_{\mathrm{C}} = \mathbf{O}_{\mathcal{B}}\mathbf{G}_{\mathcal{B}} = \begin{bmatrix}g_x & g_y & g_z\end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^3$, expressed in $\boldsymbol{\mathcal{F}}_{\mathcal{B}}$, see Fig. 6.2. Indeed, this can often happen because of unavoidable asymmetries in the quadrotor mass distribution, especially when considering onboard sensors, *e.g.* cameras, batteries or processing units. The parameters considered uncertain for the space quadrotor model are then the two aerodynamic coefficients k_f, k_τ, and the location of the CoG $\mathbf{r}_{\mathrm{C}}$. Therefore, we set our uncertain parameters vector as $\mathbf{p} = \begin{bmatrix}k_f & k_\tau & g_x & g_y & g_z\end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^5$.

We then now detail how one can derive the quadrotor dynamics under this assumption. In

the case under consideration of a non-negligible displacement between the geometric centre and the CoM, the equations of motion can be obtained as follows, see, *e.g.* [Featherstone, 2014]. Let g be the Earth acceleration magnitude, m be the mass of the UAV and $\mathbf{J}$ its inertia tensor: since we consider the shift in the CoG, the inertia at $\mathbf{O}_{\mathcal{B}}$, namely $\mathbf{J}_{\mathbf{O}_{\mathcal{B}}}$, can be linked to the (always minimal) inertia $\mathbf{J}_{\mathbf{G}_{\mathcal{B}}}$ by the Huygens-Steiner theorem, that is, in its generalised second-order tensor form,

$$\mathbf{J}_{\mathbf{O}_{\mathcal{B}}} = \underbrace{\begin{bmatrix} \mathrm{J}_{xx} & 0 & 0 \\ 0 & \mathrm{J}_{yy} & 0 \\ 0 & 0 & \mathrm{J}_{zz} \end{bmatrix}}_{\mathbf{J}_{\mathbf{G}_{\mathcal{B}}}} + \underbrace{m \begin{bmatrix} g_y^2 + g_z^2 & -g_x g_y & -g_x g_z \\ -g_x g_y & g_x^2 + g_z^2 & -g_y g_z \\ -g_x g_z & -g_y g_z & g_x^2 + g_y^2 \end{bmatrix}}_{\text{additional inertia, due to } \mathbf{r}_{\mathrm{C}}}. \tag{6.34}$$

However, for the (later) considered ranges $\delta_{\{g_x,\, g_y,\, g_z\}} = 0.03\,[\mathrm{m}]$ of the CoM shift, and the original values of $\mathbf{J}_{\mathbf{G}_{\mathcal{B}}}$, the changes in the new inertia $\mathbf{J}_{\mathbf{O}_{\mathcal{B}}}$ are negligible. Hence, for computation purposes, we do not take this phenomena into account, since it would require three additional parameters in the sensitivity derivations, and an even more complex model. Therefore, here we take $\mathbf{J} = \mathbf{J}_{\mathbf{O}_{\mathcal{B}}} \approx \mathbf{J}_{\mathbf{G}_{\mathcal{B}}}$.

Now, the external forces and torques applied to the body of the quadrotor are its weight, plus the total acting thrust and torques exerted by the four AUs. Because of the displacement $\mathbf{r}_{\mathrm{C}}$, the Newton-Euler equations of motion differ from the usual, see *e.g.* [Mellinger, Lindsey, et al., 2011; Palunko et al., 2011]: there is now a coupling between the linear and angular accelerations. Therefore, we can write the Newton-Euler equations of motion at $\mathbf{O}_{\mathcal{B}}$, in the matrix form

$$\underbrace{\begin{bmatrix} f(\mathbf{u})\mathbf{z}_{\mathcal{W}} - mg\mathbf{R}^{\mathsf{T}}(\boldsymbol{\rho}) \cdot \mathbf{z}_{\mathcal{W}} \\ \boldsymbol{\tau}(\mathbf{u}) - mg\,[\mathbf{r}_{\mathrm{C}}]_{\times} \cdot \mathbf{R}^{\mathsf{T}}(\boldsymbol{\rho}) \cdot \mathbf{z}_{\mathcal{W}} \end{bmatrix}}_{\mathbf{F}_{\text{ext}}} = \underbrace{\begin{bmatrix} m\mathbf{I}_3 & -m\,[\mathbf{r}_{\mathrm{C}}]_{\times} \\ m\,[\mathbf{r}_{\mathrm{C}}]_{\times} & \mathbf{J} \end{bmatrix}}_{\mathbf{S}_{m,\,\mathbf{J}}} \cdot \begin{bmatrix} \boldsymbol{\gamma}(\mathrm{t}) \\ \boldsymbol{\alpha}(\mathrm{t}) \end{bmatrix} + \underbrace{\begin{bmatrix} m\,[\boldsymbol{\omega}]_{\times} \cdot [\boldsymbol{\omega}]_{\times} \cdot \mathbf{r}_{\mathrm{C}} \\ [\boldsymbol{\omega}]_{\times} \cdot \mathbf{J} \cdot \boldsymbol{\omega} \end{bmatrix}}_{\mathbf{F}_{\text{fict}}}, \tag{6.35}$$

where $\mathbf{F}_{\text{ext}} \in \mathbb{R}^6$ denotes the external forces and torques, $\mathbf{S}_{m,\,\mathbf{J}} \in \mathbb{R}^{6\times 6}$ denotes the so-called *spatial inertia matrix*, and $\mathbf{F}_{\text{fict}} \in \mathbb{R}^6$ denote the external *fictitious forces and torques*. We recall that $\boldsymbol{\gamma}(\mathrm{t}) \in \mathbb{R}^3$ and $\boldsymbol{\alpha}(\mathrm{t}) \in \mathbb{R}^3$ are respectively the UAV linear and angular accelerations. We also emphasise that in Eq. (6.35), the forces are expressed in the body frame $\boldsymbol{\mathcal{F}}_{\mathcal{B}}$ this time, as opposed to what was done in the two previous contributions. We can now invert the latter

equation, and express $\boldsymbol{\gamma}(\mathrm{t})$, $\boldsymbol{\alpha}(\mathrm{t})$ as functions of the other terms, yielding

$$\begin{bmatrix} \boldsymbol{\gamma}(\mathbf{q},\,\mathbf{u},\,\mathbf{p}) \\ \boldsymbol{\alpha}(\mathbf{q},\,\mathbf{u},\,\mathbf{p}) \end{bmatrix} = \mathbf{S}^{-1}_{m,\mathbf{J}}(\mathbf{p}) \cdot \left[\mathbf{F}_{\text{ext}}(\mathbf{q},\,\mathbf{u},\,\mathbf{p}) - \mathbf{F}_{\text{fict}}(\mathbf{q},\,\mathbf{u},\,\mathbf{p})\right]. \tag{6.36}$$

With all the settings and conditions described above, one can now obtain the state-space model of the space quadrotor

$$\dot{\mathbf{q}}(\mathrm{t}) = \begin{cases} \dot{\mathbf{r}}(\mathrm{t}) = \mathbf{v}(\mathrm{t}) \\ \dot{\mathbf{v}}(\mathrm{t}) = \boldsymbol{\gamma}(\mathbf{q},\,\mathbf{u},\,\mathbf{p}) \\ \dot{\boldsymbol{\rho}}(\mathrm{t}) = \dfrac{1}{2}\,\boldsymbol{\rho} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \\ \dot{\boldsymbol{\omega}}(\mathrm{t}) = \boldsymbol{\alpha}(\mathbf{q},\,\mathbf{u},\,\mathbf{p}) \end{cases}, \tag{6.37}$$

which gives us the sought dynamics $\dot{\mathbf{q}}(\mathrm{t}) = \mathbf{f}(\mathbf{q},\,\mathbf{u},\,\mathbf{p})$, further exploited to derive the state/output sensitivities, for the TO problem.

6.5.2 Geometric tracking controller

As motion task, we consider the tracking of reference trajectory for the well-known quadrotor flat outputs. We then seek that the output function $\mathbf{y}(\mathbf{q})$ of the space quadrotor, defined in the previous section, tracks a desired trajectory $\mathbf{y}_{\mathrm{d}}(\mathbf{a},\,\mathrm{t})$. To this end, as in Chapter 5, we make use of the popular geometric controller, see [T. Lee, Leok, et al., 2010a, 2010b], with an integral term on the position. Again, we refer the reader to Subsection 5.2.2 for full details.

6.5.3 Trajectory representation

The chosen class of curves for representing our trajectories is the same as in Chapter 5. Once again, we refer the reader to Subsection 5.2.3 for more insights on piece-wise Bézier curves.

6.6 Extensive statistical campaign of simulations

In this section we discuss the results of an extensive statistical analysis, in order to validate the theory presented beforehand, *i.e.* to assess the capabilities of problem Eq. (6.32), with the weights $\mathrm{w}_1 = 1$ and $\mathrm{w}_2 = 0$, when aiming to reach accurately the final output target.

6.6.1 Settings and procedure

The procedure consists of generating a set of $\mathrm{n}_{\oplus} = 30$ non-optimised trajectories $\mathbf{a}_{\overline{\mathbf{u}},\oplus}$ and corresponding $\mathbf{a}^{\star}_{\boldsymbol{\Upsilon}}$ to compare their performances against $\mathrm{n}_{\text{per}} = 30$ parameter perturbations, when tracked by the geometric controller of the space quadrotor. For this campaign, the nominal parameters (implanted inside the controller) are:

— $k_{f_{\mathrm{c}}} = 6.5 \times 10^{-4}\,[\mathrm{N.rad^2.s^{-2}}]$; $k_{\tau_{\mathrm{c}}} = 7.7 \times 10^{-2}\,[\mathrm{m}]$;
— $g_{x_{\mathrm{c}}} = 0\,[\mathrm{m}]$; $g_{y_{\mathrm{c}}} = 0\,[\mathrm{m}]$; $g_{z_{\mathrm{c}}} = 0\,[\mathrm{m}]$.

Each output target $\mathbf{y}_{\mathrm{d}}(\mathrm{T})$ is randomly selected according to a uniform distribution pattern where the bounds are

$$\mathbf{y}_{\mathrm{d}}(\mathrm{T}) = \begin{bmatrix} x_{\oplus} \\ y_{\oplus} \\ z_{\oplus} \\ \psi_{\oplus} \end{bmatrix} \in [1,\,4] \times \left[\frac{1}{2},\,2\right] \times \left[-\frac{1}{2},\,\frac{1}{2}\right] \times [-\pi,\,\pi]\,. \tag{6.38}$$

The final desired output velocities and accelerations are set to zero, since we desire a rest-to-rest motion to the output target. As this conditions are quite hard to reach in reality, because the geometric controller does not guarantee a perfect tracking of the reference, as in Chapter 5, Subsubsection 5.3.3.1, see Eq. (5.23) and Fig. 5.4, we relaxed the equality constraints on the trajectory itself, and replaced them with constraints that force the system to go to the desired target in the nominal case (by tweaking the reference motion as needed). We consider actuation constraints on the total thrust f exerted by the quadrotor, such that $0 < f < 2mg$, translated into propellers rotation rates with the inverse mapping of Eq. (3.24). We recall to the reader that these constraints are constructed with the tubes in the input space, and fed to the optimiser, to ensure that the actuation limits are not reached, even in the most disadvantageous perturbed case, *i.e.* when the deviation $\Delta\mathbf{p}$ is at the boundary of the parameter ellipsoid of Eq. (6.6). In this contribution, we also switched the preconditioning cost to the well known minimum snap algorithm of [Mellinger and Kumar, 2011], only with the linear snap in the cost. As such, the

preconditioning optimisation problem can be written

$$\mathbf{a}_{\underline{\overline{\mathbf{u}}},\oplus} = \arg\min_{\mathbf{a}\in\mathcal{A}} \quad \left(\int_0^{\mathrm{T}} \|\mathbf{s}_{\mathrm{c}}(\tau)\|\,\mathrm{d}\tau\right)$$
$$\text{s.t.} \quad \left\{\begin{array}{l} \mathbf{C}_{\text{eq.}} = \mathbf{0}_{\mathrm{n}_{\mathbf{C}_{\text{eq.}}}} \\ \mathrm{u}_{\min} \leqslant \mathrm{u}_{\mathrm{c},i}(\mathrm{t}) - \mathrm{r}_{\mathbf{n}_{\mathrm{u}_i}}\left(\mathbf{K}_{\boldsymbol{\Theta},\delta}(\mathrm{t})\right) \\ \mathrm{u}_{\mathrm{c},i}(\mathrm{t}) + \mathrm{r}_{\mathbf{n}_{\mathrm{u}_i}}\left(\mathbf{K}_{\boldsymbol{\Theta},\delta}(\mathrm{t})\right) \leqslant \mathrm{u}_{\max} \end{array}\right\} \forall\,(i,\,\mathrm{t}) \in [\![1,\,\mathrm{n}_{\mathbf{u}}]\!] \times [0,\,\mathrm{T}] \tag{6.39}$$

where $\mathbf{s}$ is the snap (fourth derivative of the linear position w.r.t. time) and $\mathbf{C}_{\text{eq.}}$ are the same equality constraints as the ones in Eq. (5.23), but now for the complete output $\mathbf{y}$. As $\mathbf{a}_{\underline{\overline{\mathbf{u}}},\oplus}$ satisfies all the constraints and reaches the target accurately, we do the statistical evaluation by comparing $\mathbf{a}_{\underline{\overline{\mathbf{u}}},\oplus}$ and $\mathbf{a}^{\star}_{\boldsymbol{\Upsilon}}$ (obtained with the cost of Eq. (6.32)).

For each target $\mathbf{y}_{\mathrm{d}}(\mathrm{T})$ and associated trajectories $\mathbf{a}_{\underline{\overline{\mathbf{u}}},\oplus}$ and $\mathbf{a}^{\star}_{\boldsymbol{\Upsilon}}$, we achieved n_{per} simulations, while drawing $\Delta\mathbf{p}$ according to a uniform distribution, see Eq. (6.4). As such, in general there is no guarantee that the drawn perturbation ensures that $\mathbf{p}$ remains inside the bounds of the parameter ellipsoid of Eq. (6.6). Let

$$\Delta^{\mathsf{T}}\mathbf{p}\mathbf{W}_{\delta}^{-1}\Delta\mathbf{p} = \mathrm{r} \tag{6.40}$$

for a given perturbation vector $\Delta\mathbf{p}$. If $\mathrm{r} > 1$ then $\Delta\mathbf{p}$ lies outside of the ellipsoid, we address this problem by re-scaling it

$$\Delta\mathbf{p} \leftarrow \frac{\Delta\mathbf{p}}{\sqrt{\mathrm{r}}}. \tag{6.41}$$

Then, for k_f and k_τ which have non-zero nominal values, the range is $\delta_{\{k_f,\,k_\tau\}} = \pm 10\,\%$. For g_x, g_y and g_z, for which the nominal value is null, $\delta_{\{g_x,\,g_y,\,g_z\}} = \pm 3 \times 10^{-2}\,[\mathrm{m}]$ is the considered uncertainty range. With these settings, we can randomly draw the $\mathrm{n}_{\text{per}} = 30$ perturbations, and re-scale with Eq. (6.41) if necessary.

From each run of all perturbed simulation runs, for $\mathbf{a}_{\underline{\overline{\mathbf{u}}},\oplus}$ and $\mathbf{a}^{\star}_{\boldsymbol{\Upsilon}}$, we measure each final linear position error norm to the desired final position target

$$\mathbf{E}_{\mathbf{r}} = \|\mathbf{r}(\mathrm{T}) - \mathbf{r}_{\mathrm{d}}(\mathrm{T})\| \tag{6.42}$$

and each yaw error norm to the desired final yaw

$$\mathbf{E}_{\psi} = \|\psi(\mathrm{T}) - \psi_{\mathrm{d}}(\mathrm{T})\|. \tag{6.43}$$

Then, on each of these four (two for $\mathbf{a}_{\underline{\overline{\mathbf{u}}},\oplus}$ and two for $\mathbf{a}^{\star}_{\boldsymbol{\Upsilon}}$) resulting sets of error values, we compute the mean and the standard deviation. As a synthesis, starting from a single target, we end up with eight numbers, namely the means $\mu_{\left\{\mathbf{E}_{\mathbf{r}},\,\mathbf{E}_{\psi}\right\}}$ and the standard deviations $\sigma_{\left\{\mathbf{E}_{\mathbf{r}},\,\mathbf{E}_{\psi}\right\}}$ of the final error norms for $\mathbf{a}_{\underline{\overline{\mathbf{u}}},\oplus}$ and $\mathbf{a}^{\star}_{\boldsymbol{\Upsilon}}$.

Finally, we aggregate these numbers over the whole set of $\mathrm{n}_{\oplus}$ final output targets, in order to compute the box-plot characteristics of these means and standard deviations. More precisely, for each set of values, we compute the median, the first and third quartiles, and the first and last centiles of the $\mathrm{n}_{\oplus}$ output error means and standard deviations.

6.6.2 Results for a single target

Before discussing the statistical results for all targets, we study the results for one single target. In particular, we focus on the tubes on the inputs and output, and then we also analyse the final output trust ellipsoid for this target.

6.6.2.1 Tubes analyses

On Fig. 6.3, we display one of the inputs that was obtained by forward integration along the reference $\mathbf{a}^{\star}_{\boldsymbol{\Upsilon}}$. We chose the second direction of the inputs $\mathbf{u}$, thus taking $\mathbf{n} = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}^{\mathsf{T}}$ for computing the tube. Therefore, the resulting lower and upper tubes are centred at the nominal. To show that our theory works correctly, we also display twenty perturbed inputs, and we verify that they remain inside the tubes. The input saturations are also displayed (in red), and one can now visualise the importance of considering the tubes in our optimisation problem: since the tubes correspond to the worst deviation within the parameter ellipsoid (*i.e.* at the edge of it), using these quantities for the saturation constraints ensure that we remain inside the bounds of the inputs for any considered deviation.

Even if we do not take advantage of it in this chapter, we display two of the state/output variables in Fig. 6.4, namely y and z, respectively on the left and right. Again, for each of these variables, the resulting lower and upper tubes are centred at the nominal. Now, imagine that one needs to plan a motion for a system, that safely avoids collisions, but under parametric uncertainties: with the uncertainty ranges knowledge, one can now exploit the state/output sensitivities to generate motion that will ensure a safe, collision-free behaviour of the robot, even for the worst case scenarios.

In our opinion, with the graphics that we display, the new theory is clearly justified in the context of safe motion planning under parametric uncertainties, and had great potential for

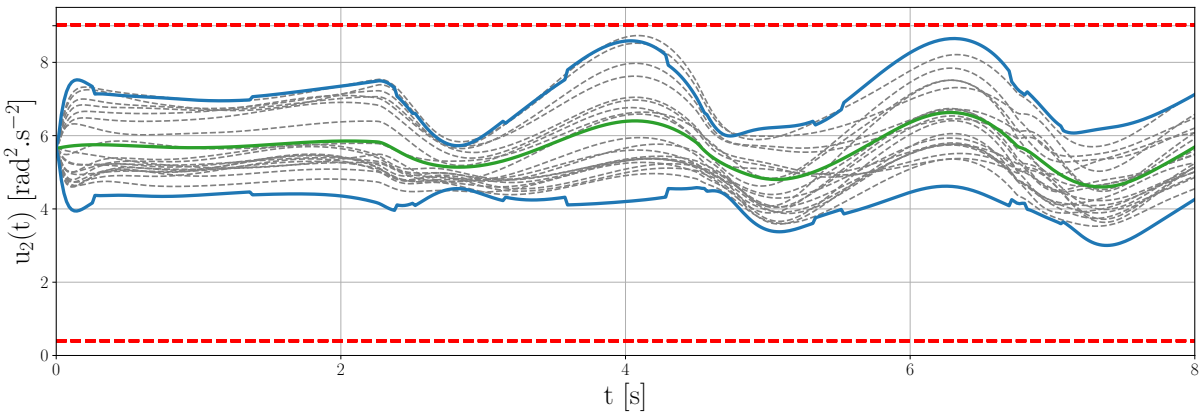

Figure 6.3 – Input tube along a reference trajectory $\mathbf{a}^{\star}_{\boldsymbol{\Upsilon}}$ of $\mathrm{T} = 8$ [s]. The nominal behaviour is in green, the twenty perturbed behaviours are in dashed grey, the upper and lower tubes are in blue, and the input saturations are in dashed red.

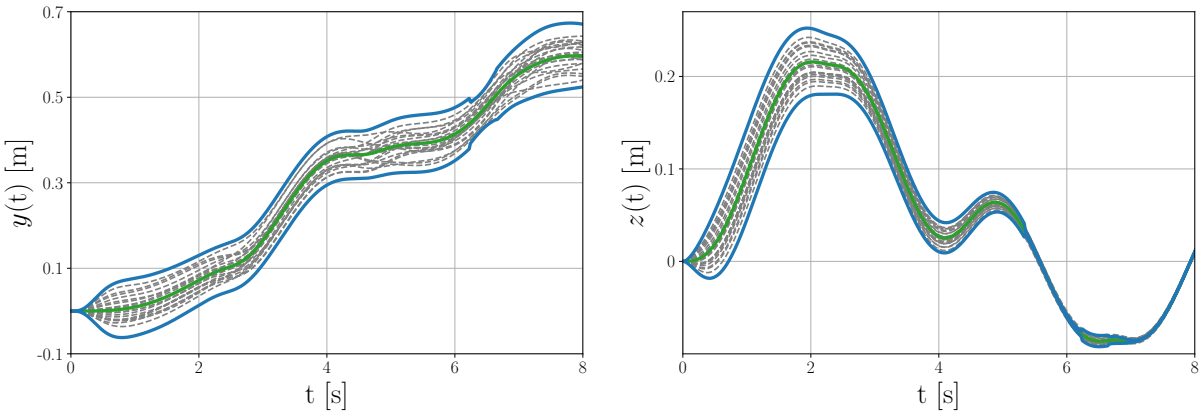

Figure 6.4 – y (left) and z (right) tubes along a reference trajectory $\mathbf{a}^{\star}_{\boldsymbol{\Upsilon}}$ of $\mathrm{T} = 8$ [s]. The nominal behaviour is in green, the twenty perturbed behaviours are in dashed grey, and the upper and lower tubes are in blue.

future robotic applications.

6.6.2.2 Final output confidence ellipsoid

We now display the results for one specific target: from the $\mathrm{n_{per}} = 30$ perturbed runs for $\mathbf{a}_{\underline{\overline{\mathbf{u}}},\oplus}$ and $\mathbf{a}^{\star}_{\boldsymbol{\Upsilon}}$, we stored the final outputs of the quadrotor in four different clouds, each of size $\mathrm{n_{per}} \times \mathrm{n_{\mathbf{y}}}$. Of these latter, we have computed the two corresponding 4-dimensional $90\,\%$ confidence ellipsoids.

On Fig. 6.5, one can observe the corresponding 2-dimensional confidence ellipses, in plane $(x,\, y)$ on the left, and in plane $(z,\, \psi)$ on the right. With this two subplots, one can now assess the

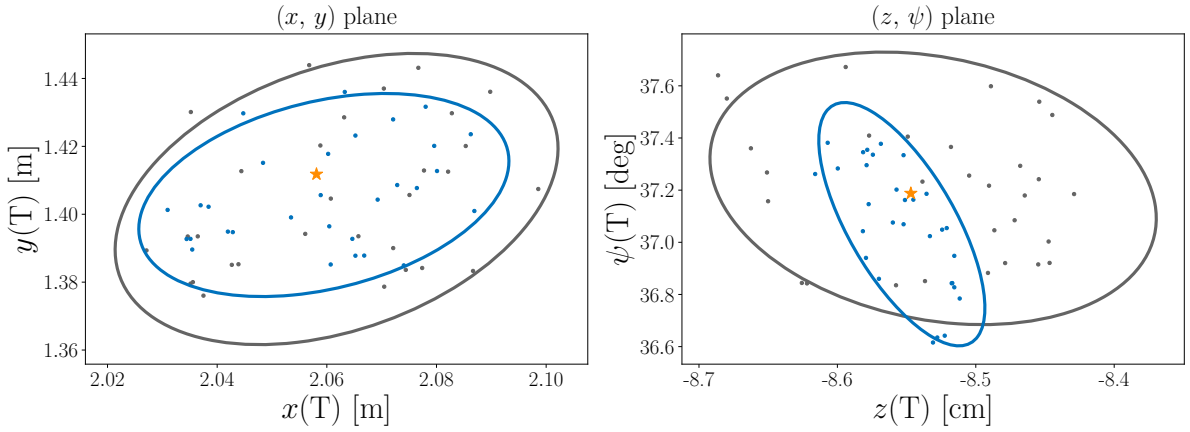

Figure 6.5 – Display of the two 90 % confidence ellipsoids, for one target only, and divided in two planes: $(x,\, y)$ on the left, and $(z,\, \psi)$ on the right. On each subplot, one can observe the 2-dimensional final confidence ellipses resulting from the perturbed simulations of trajectories $\mathbf{a}_{\underline{\mathbf{u}},\oplus}$ (in grey) and $\mathbf{a}^\star_{\boldsymbol{\Upsilon}}$ (in blue). We also displayed the final output target $\mathbf{y}_\mathrm{d}(\mathrm{T})$ (in orange).

tracking performance improvements, when reducing the final output sensitivity, for one target alone. On both planes, we can see that the ellipse corresponding to the optimised $\mathbf{a}^\star_{\boldsymbol{\Upsilon}}$ is smaller than the preconditioned one, $\mathbf{a}_{\underline{\mathbf{u}},\oplus}$. This means that the statistical spread of the perturbed final output is smaller after optimising the reference motion.

6.6.3 Statistics

We now show the complete statistical results, thus for all targets.

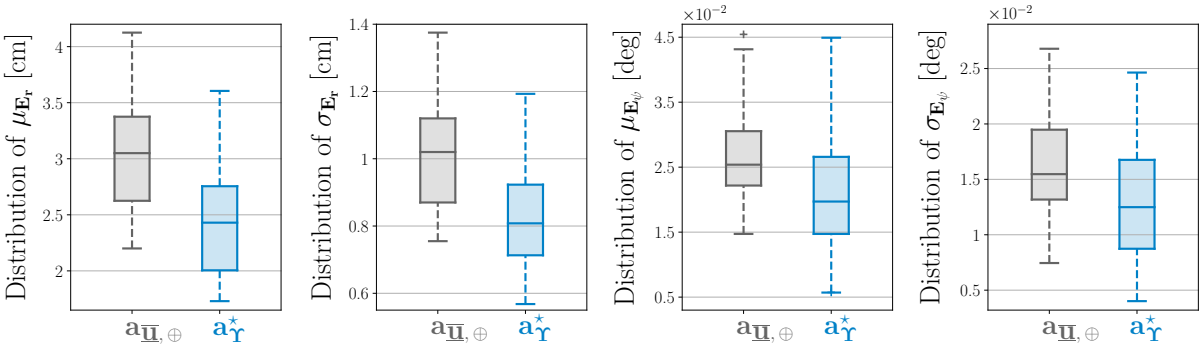

Figure 6.6 – Box-plots of the evaluated performances for the conducted statistical campaign when comparing all the preconditioned $\mathbf{a}_{\underline{\mathbf{u}},\oplus}$ (in grey) to the optimised trajectories $\mathbf{a}^\star_{\boldsymbol{\Upsilon}}$ (in blue) resulting from problem Eq. (6.32). From left to right, we display respectively the distribution of the means and stds of the position error $\mathbf{E}_\mathbf{r}$, and then for the yaw error $\mathbf{E}_\psi$.

On Fig. 6.6, we display four graphics that show the distributions (from left to right) of respectively the distribution of the means and stds of the position error $\mathbf{E}_{\mathbf{r}}$, and then the same for the yaw error $\mathbf{E}_{\psi}$. These are the results of a full campaign of $\mathrm{n}_{\text{per}} = 30$ simulation runs for $\mathrm{n}_{\oplus} = 30$ random output targets. As all graphics highlight a reduction of the means or stds for the linear position error norm, and the yaw error norm when comparing $\mathbf{a}^{\star}_{\boldsymbol{\Upsilon}}$ to $\mathbf{a}_{\underline{\overline{\mathbf{u}}},\oplus}$, we draw the conclusion that with the cost formulation, with the new output sensitivity norm allows for statistical reduction of the error (in linear and angular spaces).

From these results, we strongly believe that the effectiveness of Eq. (6.32) is proven, at least in the conditions we described, *i.e*. for the space quadrotor with a shift in the CoG equipped with the geometric tracking controller. Consequently, one can understand that generating optimised trajectories for which the output sensitivity norm is smaller than for non-optimised shapes allows to statistically increase the output predictability, and therefore results in a less hazardous dynamical behaviour when $\mathbf{p} \neq \mathbf{p}_{\mathrm{c}}$: the performance of the system is enhanced overall, and this ensures an improvement in security during the control task.

6.7 Conclusion

As a conclusion, in this chapter we have provided a new version of our robust trajectory planning framework. This new, tube-based approach, allows to leverage the information contained in the state/input/output sensitivities, in order to compute the worst case deviation tubes around the nominal state/input/output. Thanks to the uncertainty ranges, we were able to provide an improved, informative norm, that is directly linked to the deviations in the state/input/output spaces.

We have applied this new problem for an improved space quadrotor model, that may have a shift in its CoM. From the analyses and observations made in the result section, the benefits of this framework are numerous: this formulation allows for to generate robust trajectories w.r.t. parametric uncertainties, that statistically enhance the performance of our systems, and which do that in a very safe and reliable way. We can mention the ability to plan motions that will ensure the dynamic feasibility even with the most disadvantageous deviations (thanks to the input tubes), and that can also avoid collisions efficiently (thanks to the state/output tubes).

Part III

General conclusion and perspectives

CONCLUSION

In this thesis, we defended the idea that the generation of robust, *control-aware* trajectories is a promising tool to plan robotic tasks under parametric uncertainties. We showed that an adequate use of our 'closed-loop sensitivities framework' yields better performance than trivial, non-optimised motion plans. Throughout our contributions, we highlighted the fact that each sensitivity (state, input and output) has its own function, and that one can exploit any to enhance the performance of the system at a specific time, or during all the robotic task, so as to minimise the gap of behaviour (in the space of interest) between the *nominal* and *perturbed* cases. In particular, in the context of free-flying aerial robotics with uncertain parameters, we have shown multiple times, with different versions of our framework, that tracking our robust motion plans leads to statistical performance improvements, and ensures a better predictability of a quadrotor UAV behaviour.

6.8 Our contribution

In Chapter 4 we proposed a first extension of the already existing sensitivity framework, with the addition of the newly introduced input sensitivity, which, if minimised, leads to increased input predictability. After deriving the state and input sensitivities, we have shown how to obtain them and their gradients (w.r.t. the optimisation variables) along any reference motion. We then formulate an optimisation problem that combines both the state and input sensitivities in a single cost objective, and seeks to minimise the state error at the end of our motion, plus that guarantees a better prediction of our inputs. This problem is solved under constraints, *i.e.* kinematic limit conditions and the (newly introduced) input saturations. In this chapter, compared to what was done beforehand, we replaced the trajectory representation by Bézier curves, which offer more numerical stability than 'plain polynomials'. We then assessed the soundness of the proposed strategy, with the realisation of an extensive statistical campaign of perturbed simulations (to simulate the uncertain behaviour of our robot) applied to trajectories that were optimised for a planar quadrotor, equipped with a 'perfect' (in the nominal case) DFL controller. The campaign has shown performance improvements for two versions of the controller. The results demonstrate that both the state and input sensitivities prove useful to reduce the

errors in their respective spaces. We also highlight that both objectives have been combined in a single optimisation. The corresponding trajectories have shown to be a good trade-off between those minimising a single index. However, we recall that our method relies on the hypothesis that the state is fully known during the control task (to perform the sensitivity computations), which, of course, is wrong: as the state (continuous) is an estimate that gets computed from all the (discrete) measurements, by nature, it is uncertain.

Motivated by this last point, we collaborated with the research group 'Control of Networked Systems' (University of Klagenfurt), which studies the planning of trajectories whose tracking renders the system's state and/or parameters well observable. This lead to our 'control & observability-aware planning framework': in Chapter 5, we propose an attempt to combine the closed-loop state and input sensitivities with the observability metric, for robust and *informative* motion plans. Intuitively, the objectives are opposite. Thanks to the augmented weighted Chebyshev method, we were able to balance both objectives in a single problem. In this chapter, the trajectory representation has been extended to piece-wise Bézier curves, as it allows for dimension reduction. Our multi-step approach was then assessed for a space quadrotor equipped with a geometric controller (imperfect in the nominal case): in a simplified case study, we considered the thrust and drag coefficients of the propellers as uncertain. Indeed, a more thorough study would require to consider other uncertain parameters, and of course, the state as uncertain (as it is linked to the 'state knowledge' hypothesis). Anyhow, a comparison was done between the preconditioned, and the S/I-S, the EELOG, the COP optimised references. As our controller is not perfect in the nominal case, we had to relax the constraints at the end of the trajectories, and replace them by constraints that ensure that our system actually goes to the desired target (in the nominal case). This allows to fairly compare the non-optimised references to the optimised ones. In the results section, we have shown that the sensitivity and observability are often conflicting, which completely justifies our approach. In the described conditions, the COP optimised motions have shown to be a good compromise between the control-aware and observability-aware trajectories.

In Chapter 6, we have been concerned with finding a good norm for our cost objective: indeed, the Frobenius matrix norm that we exploited in the two preceding chapters, does not take advantage of a possible knowledge of the parametric uncertainty ranges. One can agree that for each parameter of our models, we always have a good idea of the measure uncertainty. With this information, it is possible to construct the largest deviation ellipsoid in the parameter space. We then show how to exploit the state/input/output sensitivities to obtain the corresponding ellipsoids in their respective spaces. From this theory, we now exploit the 'worst-case deviation

tubes'. These are of utmost importance as they allows us to now consider new constraints for the input saturation: we replace the old constraints by the tubes, and ensure that the trajectory remains dynamically feasible (even with the worst deviation). Moreover, this grants us the ability to remove the input sensitivity from the cost, and now gives full room to any combination of the final/integral state/output sensitivities. Besides, we updated the cost with a new matrix norm of the output sensitivity kernel matrix (*i.e.*, its largest eigenvalue). This new norm represents the worst-case deviation in the output space, given the parametric uncertainty model. In this chapter, we extended our quadrotor model to one that has a shift in its CoG, thus adding new uncertainties to our tube-based sensitivity framework. The results section leads one to believe that the tubes work correctly on the state/input/output. In the context of motion planning, these can be exploited to generate motions which we ensure dynamically feasible and collision-free. The approach is validated, since we are able to enhance the final quadrotor output precision.

The research track that we pursued in this thesis provides good insights about the potential of our robust and control-aware trajectory planning approach. The concept of sensitivity is already quite old, but its consideration for a closed-loop system/controller pair is recent. We showed that the field of robust trajectory planning can benefit from our metrics, and the novel ways to exploit them for relevant optimisation problems. We believe that we have only scratched the surface of this topic, and it should greatly benefit from further exploration.

6.9 Limits and research perspectives

Three years (almost four) is a short period on the timescale of research. We limited ourselves to "simple" tasks, and applied them on a unique quadrotor. However, in order to convince other researchers and engineers in the field of robotics that our contribution is worth it, further validation is required.

Overall, the framework is very slow. Even if it has been improved a lot (*e.g.*, the compilation of the sensitivity and gradients took weeks for a space quadrotor, equipped with a geometric controller using 'matlabfunction' of Matlab) by switching to Python (using 'JiTCODE', which allowed to reduce the time to minutes, then seconds), the optimisation still require too much time, and we are quite far from generating at real-time. As such, we advice to explore the use of learning NNs to provide estimates of the sensitivity cost itself, to fasten the process. The potential of NNs is already proven in the scientific community, and could, if well used in the context of our sensitivity framework, enable high-speed, real-time generation/planning/replanning.

The optimisation routines that we have chosen were always gradient-descent algorithms (or close). We think that this is very limiting, as it does not allow for efficient exploration of the trajectory space. Therefore, we suggest to try implementing population-based algorithms (*e.g.* GAs) for the planning process.

Throughout the manuscript, we find an optimal shape of the trajectory, and leave the system/controller pair unchanged. Well, we believe that besides the curve parameterisation, one could formulate an optimisation problem that automatically tunes the controller gains for each trajectory shape, which would certainly lead to even better sensitivity reduction. One could also compare the benefits from our method for control policies with different levels of robustness: can we improve the performance of a closed-loop system that includes an already robust controller?

We also mention that in the statistical campaigns of Chapters 4 and 6, we searched accurate positioning at the end of the navigation. This is quite limiting in terms of results interpretation, as we never tried to enhance the performance while the system is moving. Besides in our evaluations the motion tasks were limited to rest-to-rest. We believe that in these chapters, the improvements that we see are the hardest to obtain (compared to what could arise from a moving system), since when the motion goes back to rest, then the controller has "time" to correct for all the errors, *i.e.* the tracking error, and the one that is caused by parametric uncertainties. Therefore, it would be really interesting to see what happens when optimising at a time point for which the system is at great speed. We expect that the difference in performance when tracking a non-optimised motion and a robust one would be even larger. If this hypothesis is verified, optimising for precision at a specific time point could, *e.g.*, have potential for planned drone racing.

As stated in Chapter 1, we think that our robust motion planning algorithms are promising for a wide range of applications, and would like to see how our research can be used for other objectives than contact-free flying. We believe that APhI, and other fields of research could benefit from our ideas (and their future extensions). For instance, one could imagine testing our method with other systems, such as aerial manipulators, *e.g.* a fully actuated tilted hexarotor, equipped with a relevant end effector. Our approach should be tested for standard manipulators, *e.g.* for tasks such as tossing an object with good precision (where the object to be tossed is uncertain).

In our opinion, another limiting factor for evaluating the benefits of our method is that we always test it on already quite sophisticated systems. It could be really interesting to see the outcomes of trying our strategy with very simple, small (or micro) robots, that, by nature, have

limited hardware and are inherently subject to parametric uncertainties.

A series of real experiments has already been conducted in the past for our framework (with positive results) on a unicycle robot. We would like to perform these experiments on a real-world quadrotor, since this validation lacks as of today. Then, if the results are also positive, this validation should be extended with specific tasks runs, mimicking relevant possible applications.

To close this thesis, we firmly believe that our work is a stepping stone on the path to robust, control-aware trajectory planning for robotic tasks under parametric uncertainty. We hope that we have convinced other researchers in the field to invest time and energy on our topic, so that novel, efficient closed-loop sensitivity planning algorithms can see the light in the near future.

BIBLIOGRAPHY

Abas, Norafizah, Ari Legowo, and Rini Akmeliawati (2011), « Parameter identification of an autonomous quadrotor », *in*: *2011 4th international conference on mechatronics (ICOM)*, IEEE, pp. 1–8 (cit. on p. 32).

Abdi, Hervé, Dominique Valentin, and Betty Edelman (1999), *Neural networks*, 124, Sage (cit. on p. 27).

Afifi, Amr, Mark van Holland, and Antonio Franchi (2022), « Toward physical human-robot interaction control with aerial manipulators: Compliance, redundancy resolution, and input limits », *in*: *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 4855–4861 (cit. on p. 7).

Alexis, Kostas et al. (2016), « Aerial robotic contact-based inspection: planning and control », *in*: *Autonomous Robots* 40, pp. 631–655 (cit. on p. 7).

Allgöwer, Frank and Alex Zheng (2012), *Nonlinear model predictive control*, vol. 26, Birkhäuser (cit. on p. 25).

Ansari, Alex and Todd Murphey (2013a), « Minimal parametric sensitivity trajectories for nonlinear systems », *in*: *2013 American Control Conference*, IEEE, pp. 5011–5016 (cit. on p. 40).

— (2013b), « Minimal sensitivity control for hybrid environments », *in*: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, pp. 3023–3028 (cit. on p. 40).

— (2016), « Minimum sensitivity control for planning with parametric and hybrid uncertainty », *in*: *The International Journal of Robotics Research* 35.7, pp. 823–839 (cit. on pp. 40, 86).

Araar, Oualid and Nabil Aouf (2014), « Full linear control of a quadrotor UAV, LQ vs H-infinity », *in*: *2014 UKACC International Conference on Control (CONTROL)*, IEEE, pp. 133–138 (cit. on p. 22).

Argentim, Lucas M et al. (2013), « PID, LQR and LQR-PID on a quadcopter platform », *in*: *2013 International Conference on Informatics, Electronics and Vision (ICIEV)*, IEEE, pp. 1–6 (cit. on p. 21).

Ascorti, Leonardo (2013), « An application of the extended Kalman filter to the attitude control of a quadrotor », *in*: (cit. on p. 30).

Åström, Karl J and Björn Wittenmark (2013), *Adaptive control*, Courier Corporation (cit. on p. 32).

Atheer, L Salih, AF Mohamed Haider, and Sallom Gaeid Khalaf (2010), « Flight PID controller design for a UAV quadrotor », *in*: *Scientific research and essays* 5.*23*, pp. 3660–3667 (cit. on p. 20).

Badr, Sherif, Omar Mehrez, and AE Kabeel (2016), « A novel modification for a quadrotor design », *in*: *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, pp. 702–710 (cit. on p. 18).

Bar-Shalom, Yaakov, X Rong Li, and Thiagalingam Kirubarajan (2004), *Estimation with applications to tracking and navigation: theory algorithms and software*, John Wiley & Sons (cit. on p. 32).

Baskaya, Elgiz et al. (2021), « A novel robust hexarotor capable of static hovering in presence of propeller failure », *in*: *IEEE Robotics and Automation Letters* 6.*2*, pp. 4001–4008 (cit. on p. 16).

Benallegue, Abdelaziz, Abdellah Mokhtari, and Leonid Fridman (2008), « High-order sliding-mode observer for a quadrotor UAV », *in*: *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* 18.*4-5*, pp. 427–440 (cit. on p. 31).

Bensalah, Choukri, Nacer K M'Sirdi, and Aziz Naamane (2019), « Full modelling and sliding mode control for a quadrotor uav in visual servoing task », *in*: *IMAACA2019* (cit. on p. 25).

Bishop, Chris M (1994), « Neural networks and their applications », *in*: *Review of scientific instruments* 65.*6*, pp. 1803–1832 (cit. on p. 27).

Böhm, Christoph, Pascal Brault, et al. (2022), « COP: Control & Observability-aware Planning », *in*: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3364–3370 (cit. on pp. 12, 87, 109).

Böhm, Christoph, Guanrui Li, et al. (2020), « Observability-Aware Trajectories for Geometric and Inertial Self-Calibration », *in*: *Power On and Go Robots* (cit. on pp. 33, 85, 86).

Böhm, Christoph, Martin Scheiber, and Stephan Weiss (2021), « Filter-Based Online System-Parameter Estimation for Multicopter UAVs », *in*: *Robotics: Science and Systems* (cit. on pp. 33, 86, 94, 95, 102, 106).

Bouabdallah, Samir, Pierpaolo Murrieri, and Roland Siegwart (2005), « Towards autonomous indoor micro VTOL », *in*: *Autonomous robots* 18.*2*, pp. 171–183 (cit. on p. 20).

Bouabdallah, Samir, Andre Noth, and Roland Siegwart (2004), « PID vs LQ control techniques applied to an indoor micro quadrotor », *in*: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3, IEEE, pp. 2451–2456 (cit. on pp. 15, 20, 21).

Bouabdallah, Samir and Roland Siegwart (2005), « Backstepping and sliding-mode techniques applied to an indoor micro quadrotor », *in*: *Proceedings of the 2005 IEEE international conference on robotics and automation*, IEEE, pp. 2247–2252 (cit. on pp. 15, 23, 25).

Bouzid, Yasser, Yasmina Bestaoui, and Houria Siguerdidjane (2017), « Quadrotor-UAV optimal coverage path planning in cluttered environment with a limited onboard energy », *in*: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 979–984 (cit. on p. 35).

Brault, Pascal, Quentin Delamare, and Paolo Robuffo Giordano (2021), « Robust Trajectory Planning with Parametric Uncertainties », *in*: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11095–11101 (cit. on pp. 12, 62).

Brault, Pascal and Paolo Robuffo Giordano (2023), « Tube-Based Trajectory Optimisation for Robots with Parametric Uncertainty », *in*: In preparation for the IEEE Robotics and Automation Letters (RA-L) (cit. on pp. 12, 110).

Bristeau, Pierre-Jean et al. (2009), « The role of propeller aerodynamics in the model of a quadrotor UAV », *in*: *2009 European control conference (ECC)*, IEEE, pp. 683–688 (cit. on p. 14).

Byrne, P and M Burke (1976), « Optimization with trajectory sensitivity considerations », *in*: *IEEE Transactions on Automatic Control* 21.*2*, pp. 282–283 (cit. on p. 39).

Camacho, Eduardo F and Carlos Bordons Alba (2013), *Model predictive control*, Springer science & business media (cit. on p. 24).

Candido, Salvatore and Seth Hutchinson (2010), « Minimum uncertainty robot path planning using a POMDP approach », *in*: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, pp. 1408–1413 (cit. on p. 39).

— (2011), « Minimum uncertainty robot navigation using information-guided POMDP planning », *in*: *2011 IEEE International Conference on Robotics and Automation*, IEEE, pp. 6102–6108 (cit. on p. 39).

Carino, Jossué, Hernan Abaunza, and P Castillo (2015), « Quadrotor quaternion control », *in*: *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, pp. 825–831 (cit. on p. 53).

Chamseddine, Abbas et al. (2012), « Flatness-based trajectory planning/replanning for a quadrotor unmanned aerial vehicle », *in*: *IEEE Transactions on Aerospace and Electronic Systems* 48.*4*, pp. 2832–2848 (cit. on p. 38).

Chen, Jinbao et al. (2019), « An improved probabilistic roadmap algorithm with potential field function for path planning of quadrotor », *in*: *2019 Chinese Control Conference (CCC)*, IEEE, pp. 3248–3253 (cit. on p. 34).

Cheng, Chi-Tsun et al. (2009), « Cooperative path planner for UAVs using ACO algorithm with Gaussian distribution functions », *in*: *2009 IEEE International Symposium on Circuits and Systems*, IEEE, pp. 173–176 (cit. on p. 35).

Christodoulou, Konstantinos et al. (2019), « Aerodynamic analysis of a quadcopter drone propeller with the use of computational fluid dynamics », *in*: *Chemical Engineering Transactions* 76, pp. 181–186 (cit. on p. 14).

Cowling, Ian D et al. (2007), « A prototype of an autonomous controller for a quadrotor UAV », *in*: *2007 European Control Conference (ECC)*, IEEE, pp. 4001–4008 (cit. on p. 21).

Coza, C et al. (2011), « Adaptive fuzzy control for a quadrotor helicopter robust to wind buffeting », *in*: *Journal of Intelligent & Fuzzy Systems* 22.*5-6*, pp. 267–283 (cit. on p. 27).

Dächert, Kerstin, Jochen Gorski, and Kathrin Klamroth (2010), « An adaptive augmented weighted Tchebycheff method to solve discrete, integer-valued bicriteria optimization problems », *in*: *University of Wuppertal, Tech. Rep. BUWAMNA-OPAP* 10.*06* (cit. on p. 95).

Devlin, Trieste et al. (2018), « Elbowquad: Thrust vectoring quadcopter », *in*: *2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, p. 0893 (cit. on p. 18).

Dhaybi, Mohamad and Naseem Daher (2020), « Accurate real-time estimation of the inertia tensor of package delivery quadrotors », *in*: *2020 American control conference (ACC)*, IEEE, pp. 1520–1525 (cit. on p. 33).

Diebel, James (2006), « Representing attitude: Euler angles, unit quaternions, and rotation vectors », *in*: *Matrix* 58.*15-16*, pp. 1–35 (cit. on p. 45).

Dierks, Travis and Sarangapani Jagannathan (2009), « Output feedback control of a quadrotor UAV using neural networks », *in*: *IEEE transactions on neural networks* 21.*1*, pp. 50–66 (cit. on p. 27).

Dijkstra, Edsger W (1959), « A note on two problems in connexion with graphs », *in*: *Numerische mathematik* 1.*1*, pp. 269–271 (cit. on p. 35).

de-Dios, Jose Ramiro Martínez et al. (2020), « GRVC-CATEC: Aerial robot co-worker in plant servicing (ARCOW) », *in*: *Bringing Innovative Robotic Technologies from Research Labs*

to Industrial End-users: The Experience of the European Robotics Challenges, pp. 211–242 (cit. on p. 7).

Edwards, Christopher and Sarah Spurgeon (1998), *Sliding mode control: theory and applications*, Crc Press (cit. on p. 25).

Efe, Mehmet Onder (2007), « Robust low altitude behavior control of a quadrotor rotorcraft through sliding modes », *in*: *2007 Mediterranean Conference on Control & Automation*, IEEE, pp. 1–6 (cit. on p. 25).

Eltayeb, Ahmed, Mohd Fua'ad Rahmat, and Mohd Ariffanan Mohd Basri (2020), « Sliding mode control design for the attitude and altitude of the quadrotor UAV », *in*: *International Journal on Smart Sensing and Intelligent Systems* 13.*1*, p. 1 (cit. on p. 25).

Emmerich, Michael and André H Deutz (2018), « A tutorial on multiobjective optimization: fundamentals and evolutionary methods », *in*: *Natural computing* 17.*3*, pp. 585–609 (cit. on pp. 95, 97).

Fantoni, Isabelle, Rogelio Lozano, and Rogelio Lozano (2002), *Non-linear control for underactuated mechanical systems*, Springer Science & Business Media (cit. on p. 23).

Featherstone, Roy (2014), *Rigid body dynamics algorithms*, Springer (cit. on p. 124).

Feron, Eric and Eric N Johnson (2008), *Aerial Robotics.* (Cit. on p. 3).

Fresk, Emil and George Nikolakopoulos (2013), « Full quaternion based attitude control for a quadrotor », *in*: *2013 European control conference (ECC)*, IEEE, pp. 3864–3869 (cit. on p. 53).

Gabriel, Darren Lance, Johan Meyer, and Francois Du Plessis (2011), « Brushless DC motor characterisation and selection for a fixed wing UAV », *in*: *IEEE Africon'11*, IEEE, pp. 1–6 (cit. on p. 14).

Gamagedara, Kanishke et al. (2019), « Geometric controls of a quadrotor UAV with decoupled yaw control », *in*: *2019 American Control Conference (ACC)*, IEEE, pp. 3285–3290 (cit. on p. 24).

Gao, Fei et al. (2018), « Optimal time allocation for quadrotor trajectory generation », *in*: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 4715–4722 (cit. on p. 37).

Gautam, Deepak and Cheolkeun Ha (2013), « Control of a quadrotor using a smart self-tuning fuzzy PID controller », *in*: *International Journal of Advanced Robotic Systems* 10.*11*, p. 380 (cit. on p. 35).

Goodarzi, Farhad A and Taeyoung Lee (2017), « Global formulation of an extended Kalman filter on SE(3) for geometric control of a quadrotor UAV », *in*: *Journal of Intelligent & Robotic Systems* 88.*2*, pp. 395–413 (cit. on p. 30).

Hamandi, Mahmoud et al. (2021), « Design of multirotor aerial vehicles: A taxonomy based on input allocation », *in*: *The International Journal of Robotics Research* 40.*8-9*, pp. 1015–1044 (cit. on pp. 13, 18).

Hamza, Abdelaziz, Abdallah H Mohamed, and Ayman El-Badawy (2022), « Robust H-infinity Control for a Quadrotor UAV », *in*: *AIAA SCITECH 2022 Forum*, p. 2033 (cit. on p. 22).

Han, Bo et al. (2018), « A review of control algorithms for quadrotor », *in*: *2018 IEEE International Conference on Information and Automation (ICIA)*, IEEE, pp. 951–956 (cit. on p. 20).

Hasircioglu, Isil, Haluk Rahmi Topcuoglu, and Murat Ermis (2008), « 3-D path planning for the navigation of unmanned aerial vehicles by using evolutionary algorithms », *in*: *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pp. 1499–1506 (cit. on p. 35).

Hausman, Karol, James Preiss, et al. (2017), « Observability-aware trajectory optimization for self-calibration with application to uavs », *in*: *IEEE Robotics and Automation Letters* 2.*3*, pp. 1770–1777 (cit. on pp. 31, 33, 85, 86, 94).

Hausman, Karol, Stephan Weiss, et al. (2016), « Self-calibrating multi-sensor fusion with probabilistic measurement validation for seamless sensor switching on a UAV », *in*: *2016 IEEE international conference on robotics and automation (ICRA)*, IEEE, pp. 4289–4296 (cit. on p. 94).

Hehn, Markus and Raffaello D'Andrea (2011), « Quadrocopter trajectory generation and control », *in*: *IFAC proceedings Volumes* 44.*1*, pp. 1485–1491 (cit. on p. 36).

Hermann, Robert and Arthur Krener (1977), « Nonlinear controllability and observability », *in*: *IEEE Transactions on automatic control* 22.*5*, pp. 728–740 (cit. on p. 94).

Hoffmann, Gabriel et al. (2007), « Quadrotor helicopter flight dynamics and control: Theory and experiment », *in*: *AIAA guidance, navigation and control conference and exhibit*, p. 6461 (cit. on p. 21).

Holzmann, Tim and J Cole Smith (2018), « Solving discrete multi-objective optimization problems using modified augmented weighted Tchebychev scalarizations », *in*: *European Journal of Operational Research* 271.*2*, pp. 436–449 (cit. on p. 95).

Hsu, David et al. (2002), « Randomized kinodynamic motion planning with moving obstacles », *in*: *The International Journal of Robotics Research* 21.*3*, pp. 233–255 (cit. on p. 35).

Jimenez-Cano, Antonio E et al. (2013), « Control of an aerial robot with multi-link arm for assembly tasks », *in*: *2013 IEEE International Conference on Robotics and Automation*, IEEE, pp. 4916–4921 (cit. on p. 7).

Johnson, Michael A and Mohammad H Moradi (2005), *PID control*, Springer (cit. on p. 20).

Kamel, Mina, Michael Burri, and Roland Siegwart (2017), « Linear vs nonlinear MPC for trajectory tracking applied to rotary wing micro aerial vehicles », *in*: *IFAC-PapersOnLine* 50.*1*, pp. 3463–3469 (cit. on p. 25).

Kamel, Mina, Thomas Stastny, et al. (2017), « Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system », *in*: *Robot operating system (ROS)*, Springer, pp. 3–39 (cit. on p. 86).

Kang, Taesam et al. (2015), « H-infinity control system design for a quad-rotor », *in*: *Journal of Institute of Control, Robotics and Systems* 21.*1*, pp. 14–20 (cit. on p. 22).

Kataoka, Yasuyuki, Kazuma Sekiguchi, and Mitsuji Sampei (2011), « Nonlinear control and model analysis of trirotor UAV model », *in*: *IFAC Proceedings Volumes* 44.*1*, pp. 10391–10396 (cit. on p. 15).

Kavraki, Lydia E, Mihail N Kolountzakis, and J-C Latombe (1998), « Analysis of probabilistic roadmaps for path planning », *in*: *IEEE Transactions on Robotics and automation* 14.*1*, pp. 166–171 (cit. on p. 34).

Kavraki, Lydia E, Petr Svestka, et al. (1996), « Probabilistic roadmaps for path planning in high-dimensional configuration spaces », *in*: *IEEE transactions on Robotics and Automation* 12.*4*, pp. 566–580 (cit. on p. 34).

Keane, John F and Stephen S Carr (2013), « A brief history of early unmanned aircraft », *in*: *Johns Hopkins APL Technical Digest* 32.*3*, pp. 558–571 (cit. on p. 4).

Khatib, Oussama (1986), « Real-time obstacle avoidance for manipulators and mobile robots », *in*: *Autonomous robot vehicles*, Springer, pp. 396–404 (cit. on p. 34).

Khatoon, Shahida, Dhiraj Gupta, and LK Das (2014), « PID & LQR control for a quadrotor: Modeling and simulation », *in*: *2014 international conference on advances in computing, communications and informatics (ICACCI)*, IEEE, pp. 796–802 (cit. on p. 21).

Kim, Il Yong and Oliver L De Weck (2005), « Adaptive weighted-sum method for bi-objective optimization: Pareto front generation », *in*: *Structural and multidisciplinary optimization* 29.*2*, pp. 149–158 (cit. on p. 72).

Kreindler, Eliezer (1969), « Formulation of the minimum trajectory sensitivity problem », *in*: *IEEE Transactions on Automatic Control* 14.*2*, pp. 206–207 (cit. on p. 39).

Krener, Arthur J and Kayo Ide (2009), « Measures of unobservability », *in*: *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, IEEE, pp. 6401–6406 (cit. on p. 94).

Kuffner, James J and Steven M LaValle (2000), « RRT-connect: An efficient approach to single-query path planning », *in*: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2, IEEE, pp. 995–1001 (cit. on p. 35).

Labbadi, Moussa and Mohamed Cherkaoui (2019), « Robust integral terminal sliding mode control for quadrotor UAV with external disturbances », *in*: *International Journal of Aerospace Engineering* 2019 (cit. on p. 25).

LaValle, Steven M (2006), *Planning algorithms*, Cambridge university press (cit. on p. 34).

Lee, Daewon, H Jin Kim, and Shankar Sastry (2009), « Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter », *in*: *International Journal of control, Automation and systems* 7.*3*, pp. 419–428 (cit. on p. 23).

Lee, Taeyoung and Youdan Kim (2001), « Nonlinear adaptive flight control using backstepping and neural networks controller », *in*: *Journal of Guidance, Control, and Dynamics* 24.*4*, pp. 675–682 (cit. on p. 27).

Lee, Taeyoung, Melvin Leok, and N Harris McClamroch (2010a), « Control of complex maneuvers for a quadrotor UAV using geometric methods on SE(3) », *in*: *arXiv preprint arXiv:1003.2005* (cit. on pp. 24, 30, 86, 88, 89, 91, 125).

— (2010b), « Geometric tracking control of a quadrotor UAV on SE(3) », *in*: *49th IEEE conference on decision and control (CDC)*, IEEE, pp. 5420–5425 (cit. on pp. 24, 30, 36, 86, 88, 89, 91, 125).

— (2013), « Nonlinear robust tracking control of a quadrotor UAV on SE(3) », *in*: *Asian journal of control* 15.*2*, pp. 391–408 (cit. on pp. 24, 30).

Li, Jun and Yuntang Li (2011), « Dynamic analysis and PID control for a quadrotor », *in*: *2011 IEEE International Conference on Mechatronics and Automation*, IEEE, pp. 573–578 (cit. on p. 20).

Madani, Tarek and Abdelaziz Benallegue (2006), « Backstepping control for a quadrotor helicopter », *in*: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, pp. 3255–3260 (cit. on p. 23).

Mahony, Robert, Vijay Kumar, and Peter Corke (2012), « Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor », *in*: *IEEE Robotics and Automation magazine* 19.*3*, pp. 20–32 (cit. on pp. 15, 75).

Marks, Aryeh, James F Whidborne, and Ikuo Yamamoto (2012), « Control allocation for fault tolerant control of a VTOL octorotor », *in*: *Proceedings of 2012 UKACC International Conference on Control*, IEEE, pp. 357–362 (cit. on p. 16).

Marler, R Timothy and Jasbir S Arora (2004), « Survey of multi-objective optimization methods for engineering », *in*: *Structural and multidisciplinary optimization* 26, pp. 369–395 (cit. on pp. 95, 97).

Martins, Luís, Carlos Cardeira, and Paulo Oliveira (2019), « Linear quadratic regulator for trajectory tracking of a quadrotor », *in*: *IFAC-PapersOnLine* 52.*12*, pp. 176–181 (cit. on p. 21).

Mellinger, Daniel and Vijay Kumar (2011), « Minimum snap trajectory generation and control for quadrotors », *in*: *2011 IEEE international conference on robotics and automation*, IEEE, pp. 2520–2525 (cit. on pp. 23, 31, 36, 126).

Mellinger, Daniel, Quentin Lindsey, et al. (2011), « Design, modeling, estimation and control for aerial grasping and manipulation », *in*: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, pp. 2668–2673 (cit. on p. 124).

Michieletto, Giulia, Markus Ryll, and Antonio Franchi (2017), « Control of statically hoverable multi-rotor aerial vehicles and application to rotor-failure robustness for hexarotors », *in*: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 2747–2752 (cit. on p. 18).

Mistler, V, A Benallegue, and NK M'sirdi (2001), « Exact linearization and noninteracting control of a 4 rotors helicopter via dynamic feedback », *in*: *Proceedings 10th IEEE international workshop on robot and human interactive communication. Roman 2001 (Cat. no. 01th8591)*, IEEE, pp. 586–593 (cit. on pp. 15, 77).

Mo, Hongwei and Ghulam Farid (2019), « Nonlinear and adaptive intelligent control techniques for quadrotor uav–a survey », *in*: *Asian Journal of Control* 21.*2*, pp. 989–1008 (cit. on p. 20).

Morbidi, Fabio, Roel Cano, and David Lara (2016), « Minimum-energy path generation for a quadrotor UAV », *in*: *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 1492–1498 (cit. on p. 38).

Murray, Richard M., Muruhan Rathinam, and Willem M. Sluis (1995), « Differential Flatness of Mechanical Control Systems: A Catalog of Prototype Systems », *in*: ASME International Mechanical Engineering Congress and Exposition (cit. on p. 23).

Muscio, Giuseppe et al. (2017), « Coordinated control of aerial robotic manipulators: Theory and experiments », *in*: *IEEE Transactions on Control Systems Technology* 26.*4*, pp. 1406–1413 (cit. on p. 7).

Nguyen, Hoa T et al. (2020), « Control algorithms for UAVs: A comprehensive survey », *in*: *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems* 7.*23*, e5–e5 (cit. on p. 20).

Okyere, Emmanuel et al. (2019), « LQR controller design for quad-rotor helicopters », *in*: *The Journal of Engineering* 2019.*17*, pp. 4003–4007 (cit. on p. 21).

Oleynikova, Helen, Michael Burri, et al. (2016), « Continuous-time trajectory optimization for online uav replanning », *in*: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 5332–5339 (cit. on p. 37).

Oleynikova, Helen, Zachary Taylor, et al. (2017), « Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning », *in*: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 1366–1373 (cit. on p. 37).

Ollero, Anibal and Bruno Siciliano (2019), *Aerial robotic manipulation*, Springer (cit. on p. 7).

Ollero, Anibal, Marco Tognon, et al. (2021), « Past, present, and future of aerial robotic manipulators », *in*: *IEEE Transactions on Robotics* 38.*1*, pp. 626–645 (cit. on p. 7).

Palunko, Ivana and Rafael Fierro (2011), « Adaptive control of a quadrotor with dynamic changes in the center of gravity », *in*: *IFAC Proceedings Volumes* 44.*1*, pp. 2626–2631 (cit. on p. 124).

Passino, Kevin M, Stephen Yurkovich, and Michael Reinfrank (1998), *Fuzzy control*, vol. 42, Addison-wesley Reading, MA (cit. on p. 26).

Patel, Yamika et al. (2017), « A review on design and analysis of the propeller used in UAV », *in*: *Int. J. Adv. Prod. Ind. Eng* 605, pp. 20–23 (cit. on p. 14).

Penin, Bryan, Paolo Robuffo Giordano, and François Chaumette (2018), « Minimum-time trajectory planning under intermittent measurements », *in*: *IEEE Robotics and Automation Letters* 4.*1*, pp. 153–160 (cit. on p. 38).

Penrose, Roger (1955), « A generalized inverse for matrices », *in*: *Mathematical proceedings of the Cambridge philosophical society*, vol. 51, 3, Cambridge University Press, pp. 406–413 (cit. on p. 113).

Ponda, Sameera, Richard Kolacinski, and Emilio Frazzoli (2009), « Trajectory optimization for target localization using small unmanned aerial vehicles », *in*: *AIAA guidance, navigation, and control conference*, p. 6015 (cit. on p. 86).

Pounds, Paul, Robert Mahony, and Peter Corke (2010), « Modelling and control of a large quadrotor robot », *in*: *Control Engineering Practice* 18.*7*, pp. 691–699 (cit. on p. 15).

Pounds, Paul, Robert Mahony, Peter Hynes, et al. (2002), « Design of a four-rotor aerial robot », *in*: *The Australian Conference on Robotics and Automation (ACRA 2002)*, pp. 145–150 (cit. on p. 15).

Preiss, James A et al. (2018), « Simultaneous self-calibration and navigation using trajectory optimization », *in*: *The International Journal of Robotics Research* 37.*13-14*, pp. 1573–1594 (cit. on pp. 31, 33, 85, 86, 94).

Prentice, Samuel and Nicholas Roy (2009), « The belief roadmap: Efficient planning in belief space by factoring the covariance », *in*: *The International Journal of Robotics Research* 28.*11-12*, pp. 1448–1465 (cit. on p. 39).

Raja, Muneeb Masood (2017), « Extended Kalman Filter and LQR controller design for quadrotor UAVs », *in*: (cit. on p. 30).

Rajappa, Sujit et al. (2015), « Modeling, control and design optimization for a fully-actuated hexarotor aerial vehicle with tilted propellers », *in*: *2015 IEEE international conference on robotics and automation (ICRA)*, IEEE, pp. 4006–4013 (cit. on p. 18).

Ratliff, Nathan et al. (2009), « CHOMP: Gradient optimization techniques for efficient motion planning », *in*: *2009 IEEE International Conference on Robotics and Automation*, IEEE, pp. 489–494 (cit. on p. 37).

Richter, Charles, Adam Bry, and Nicholas Roy (2016), « Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments », *in*: *Robotics research*, Springer, pp. 649–666 (cit. on p. 36).

Robuffo Giordano, Paolo, Quentin Delamare, and Antonio Franchi (2018), « Trajectory generation for minimum closed-loop state sensitivity », *in*: *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 286–293 (cit. on pp. 40, 61–64, 67, 69, 79, 85, 86, 88, 91, 109, 119).

Rucker, Caleb (2018), « Integrating rotations using nonunit quaternions », *in*: *IEEE Robotics and Automation Letters* 3.*4*, pp. 2979–2986 (cit. on p. 53).

Ryll, Markus, Heinrich H Bülthoff, and Paolo Robuffo Giordano (2014), « A novel overactuated quadrotor unmanned aerial vehicle: Modeling, control, and experimental validation », *in*: *IEEE Transactions on Control Systems Technology* 23.*2*, pp. 540–556 (cit. on p. 18).

Ryll, Markus, Giuseppe Muscio, et al. (2017), « 6D physical interaction with a fully actuated aerial robot », *in*: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 5190–5195 (cit. on p. 18).

Salih, Atheer L et al. (2010), « Modelling and PID controller design for a quadrotor unmanned air vehicle », *in*: *2010 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, vol. 1, IEEE, pp. 1–5 (cit. on p. 20).

Santos, Matilde, Victoria Lopez, and Franciso Morata (2010), « Intelligent fuzzy controller of a quadrotor », *in*: *2010 IEEE international conference on intelligent systems and knowledge engineering*, IEEE, pp. 141–146 (cit. on p. 27).

Schperberg, Alexander et al. (2021), « SABER: Data-Driven Motion Planner for Autonomously Navigating Heterogeneous Robots », *in*: *IEEE Robotics and Automation Letters* 6.*4*, pp. 8086–8093.

Sciavicco, Lorenzo and Bruno Siciliano (2001), *Modelling and control of robot manipulators*, Springer Science & Business Media (cit. on p. 112).

Şenkul, Fatih and Erdinç Altuğ (2014), « Adaptive control of a tilt-roll rotor quadrotor UAV », *in*: *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, pp. 1132–1137 (cit. on p. 18).

Sepulchre, Rodolphe, Mrdjan Jankovic, and Petar V Kokotovic (2012), *Constructive nonlinear control*, Springer Science & Business Media (cit. on p. 23).

Shi, Bohui et al. (2020), « UAV Trajectory Generation Based on Integration of RRT and Minimum Snap Algorithms », *in*: *2020 Chinese Automation Congress (CAC)*, IEEE, pp. 4227–4232.

Singer, Neil C and Warren P Seering (1990), « Preshaping command inputs to reduce system vibration », *in*: (cit. on p. 40).

Sola, Joan (2017), « Quaternion kinematics for the error-state Kalman filter », *in*: *arXiv preprint arXiv:1711.02508* (cit. on p. 50).

Steuer, Ralph E and Eng-Ung Choo (1983), « An interactive weighted Tchebycheff procedure for multiple objective programming », *in*: *Mathematical programming* 26.*3*, pp. 326–344 (cit. on pp. 95, 97).

Suarez, Alejandro et al. (2022), « Aerial device delivery for power line inspection and maintenance », *in*: *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, pp. 30–38 (cit. on p. 6).

Sun, Sihao, Angel Romero, et al. (2022), « A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight », *in*: *IEEE Transactions on Robotics* (cit. on p. 25).

Sun, Sihao, Xuerui Wang, et al. (2020), « Incremental nonlinear fault-tolerant control of a quadrotor with complete loss of two opposing rotors », *in*: *IEEE Transactions on Robotics* 37.*1*, pp. 116–130 (cit. on p. 14).

Thrun, Sebastian (2002), « Probabilistic robotics », *in*: *Communications of the ACM* 45.*3*, pp. 52–57 (cit. on p. 39).

Tognon, Marco, Hermes A Tello Chávez, et al. (2019), « A truly-redundant aerial manipulator system with application to push-and-slide inspection in industrial plants », *in*: *IEEE Robotics and Automation Letters* 4.*2*, pp. 1846–1851 (cit. on p. 18).

Tognon, Marco and Antonio Franchi (2020), *Theory and applications for control of aerial robots in physical interaction through tethers*, vol. 140, Springer Nature (cit. on p. 7).

Tsourdos, Antonios, Brian White, and Madhavan Shanmugavel (2010), *Cooperative path planning of unmanned aerial vehicles*, John Wiley & Sons.

Van Den Berg, Jur, Pieter Abbeel, and Ken Goldberg (2011), « LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information », *in*: *The International Journal of Robotics Research* 30.*7*, pp. 895–913 (cit. on p. 86).

Van Nieuwstadt, Michiel J and Richard M Murray (1998), « Real-time trajectory generation for differentially flat systems », *in*: *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* 8.*11*, pp. 995–1020 (cit. on p. 23).

Vinnicombe, Glenn (2001), *Uncertainty and Feedback: H [infinity] Loop-shaping and the [nu]-gap Metric*, World Scientific (cit. on p. 21).

Visioli, Antonio (2006), *Practical PID control*, Springer Science & Business Media (cit. on p. 20).

Walters, Robert W and Luc Huyse (2002), « Uncertainty analysis for fluid mechanics with applications », *in*: (cit. on p. 7).

Weiss, Stephan et al. (2012), « Versatile distributed pose estimation and sensor self-calibration for an autonomous MAV », *in*: *2012 IEEE International Conference on Robotics and Automation*, IEEE, pp. 31–38 (cit. on p. 94).

Weiss, Stephan M (2012), « Vision based navigation for micro helicopters », PhD thesis, ETH Zurich (cit. on p. 31).

Welch, Greg, Gary Bishop, et al. (1995), « An introduction to the Kalman filter », *in*: (cit. on p. 29).

Wendel, Jan et al. (2006), « An integrated GPS/MEMS-IMU navigation system for an autonomous helicopter », *in*: *Aerospace science and technology* 10.*6*, pp. 527–533 (cit. on p. 29).

Wierzbicki, Andrzej P (1995), « Multiple criteria games—Theory and applications », *in*: *Journal of Systems Engineering and Electronics* 6.*2*, pp. 65–81 (cit. on pp. 95, 97).

Wilson, Andrew D, Jarvis A Schultz, and Todd D Murphey (2014), « Trajectory optimization for well-conditioned parameter estimation », *in*: *IEEE Transactions on Automation Science and Engineering* 12.*1*, pp. 28–36 (cit. on p. 86).

Xi, Chenyang and Xinfu Liu (2020), « Unmanned Aerial Vehicle Trajectory Planning via Staged Reinforcement Learning », *in*: *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, pp. 246–255.

Xiong, Jing-Jing and En-Hui Zheng (2015), « Optimal kalman filter for state estimation of a quadrotor UAV », *in*: *Optik* 126.*21*, pp. 2862–2868 (cit. on p. 30).

Yang, Hongjiu et al. (2017), « Active disturbance rejection attitude control for a dual closed-loop quadrotor under gust wind », *in*: *IEEE Transactions on control systems technology* 26.*4*, pp. 1400–1405 (cit. on p. 21).

Yang, Liang et al. (2014), « A literature review of UAV 3D path planning », *in*: *Proceeding of the 11th World Congress on Intelligent Control and Automation*, IEEE, pp. 2376–2381 (cit. on p. 34).

Yedamale, Padmaraja (2003), « Brushless DC (BLDC) motor fundamentals », *in*: *Microchip Technology Inc* 20.*1*, pp. 3–15 (cit. on p. 14).

Young, K David, Vadim I Utkin, and Umit Ozguner (1999), « A control engineer's guide to sliding mode control », *in*: *IEEE transactions on control systems technology* 7.*3*, pp. 328–342 (cit. on p. 25).

Zhang, Weixuan, Mark W Mueller, and Raffaello D'Andrea (2016), « A controllable flying vehicle with a single moving part », *in*: *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 3275–3281 (cit. on p. 14).

Zheng, Peter et al. (2020), « TiltDrone: A fully-actuated tilting quadrotor platform », *in*: *IEEE Robotics and Automation Letters* 5.*4*, pp. 6845–6852 (cit. on p. 18).

Zhou, Fengyu, Baoye Song, and Guohui Tian (2011), « Bézier curve based smooth path planning for mobile robot », *in*: *Journal of Information &Computational Science* 8.*12*, pp. 2441–2450 (cit. on p. 69).

Zhou, Yaoming, Haoran Zhao, and Yaolong Liu (2020), « An evaluative review of the VTOL technologies for unmanned and manned aerial vehicles », *in*: *Computer Communications* 149, pp. 356–369 (cit. on p. 13).

Zulu, Andrew and Samuel John (2016), « A review of control algorithms for autonomous quadrotors », *in*: *arXiv preprint arXiv:1602.02622* (cit. on p. 20).

COLLEGE DOCTORAL BRETAGNE

MATHS, TELECOMS
INFORMATIQUE, SIGNAL
SYSTEMES, ELECTRONIQUE

Université de Rennes

Titre : Algorithmes de planification de trajectoires robustes pour des tâches robotiques en présence d'incertitudes paramétriques

Mot clés : robotique aérienne, incertitudes paramétriques, planification de trajectoires

Résumé : L'un des défis majeurs des systèmes automatisés réside dans la nécessité de fonctionner dans des conditions réelles, donc incertaines. L'incertitude paramétrique est un problème courant, et se présente lors de l'exécution de tâches robotiques diverses. Dans cette thèse, nous explorons les possibilités apportées par la planification de trajectoires, dont le suivi est intrinsèquement robuste aux incertitudes. Dans la première contribution, nous étendons les algorithmes de planification de trajectoires à sensibilité minimale en introduisant la sensibilité de l'entrée, qui, une fois réduite, conduit à une prédiction accrue de l'entrée du système. Ce problème, bien que traité pour de la robotique aérienne, est généralisé pour tout système. Le problème de sensibilité reposant sur l'hypothèse de connaissance parfaite de l'état, dans la seconde contribution nous combinons les algorithmes *control-aware* et *observability-aware* grâce à la méthode de Chebyshev, afin de générer des trajectoires robustes aux incertitudes, et assurant une meilleure estimation des variables/paramètres du système. Enfin, dans la dernière contribution, nous développons une théorie qui exploite les données d'incertitude paramétrique, afin de construire les *tubes de déviations du pire cas* autour des variables *nominales* du système. Cette nouvelle expression du problème permet d'augmenter la fiabilité des systèmes, car nous assurons la faisabilité, même pour les plus fortes déviations paramétriques.

Title: Robust trajectory planning algorithms for robotic tasks with parametric uncertainties

Keywords: aerial robotics, parametric uncertainties, trajectory planning

Abstract: One of the major challenges of automated systems is the need to operate under real-world, thus uncertain conditions. Parametric uncertainty is a common problem, and arises when performing various robotic tasks. In this thesis, we explore the possibilities provided by the planning of trajectories, whose tracking is inherently robust to uncertainties. In the first contribution, we extend minimum sensitivity trajectory planning algorithms by introducing the input sensitivity, which, when reduced, leads to an increased prediction of the system input. This problem, although treated for aerial robotics, is generalised for any system. As the sensitivity problem relies on the assumption of perfect knowledge of the state, in the second contribution, we combine the *control-aware* and *observability-aware* algorithms thanks to the Chebyshev method, in order to generate trajectories robust to uncertainties, that also ensure a better estimation of the system's variables/parameters. Finally, in the last contribution, we develop a theory that exploits the parametric uncertainty data, and construct the *worst case deviation tubes* around the *nominal* system's variables. This new expression of the problem increases the reliability of systems, as we ensure feasibility even for the largest parametric deviations.