Algorithms for estimation and control of quadrotors in physical interaction with their environment

Quentin Delamare

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1
COMUE UNIVERSITÉ BRETAGNE LOIRE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Automatique, Productique et Robotique

Par

Quentin DELAMARE

Algorithmes d'estimation et de commande pour des quadrirotors en interaction physique avec l'environnement

Thèse présentée et soutenue à Rennes, le 9/12/19
Unité de recherche : IRISA

Rapporteurs avant soutenance :

Nicolas Marchand	Directeur de Recherche au GIPSA-lab, Université Grenoble-Alpes
Pascal Morin	Professeur à l'ISIR, Université Pierre et Marie Curie

Composition du Jury :

Président :	Isabelle Fantoni	Directrice de Recherche CNRS, LS2N Nantes
Examinateurs :	Isabelle Fantoni	Directrice de Recherche CNRS, LS2N Nantes
	Nicolas Marchand	Directeur de Recherche au GIPSA-lab, Université Grenoble-Alpes
	Fabio Morbidi	Maître de Conférences au MIS, Université de Picardie Jules Verne
	Pascal Morin	Professeur à l'ISIR, Université Pierre et Marie Curie
Dir. de thèse :	Paolo Robuffo Giordano	Directeur de Recherche CNRS, IRISA/Inria Rennes
Co-dir. de thèse :	Antonio Franchi	Chargé de Recherche CNRS, LAAS, Toulouse

Résumé de la thèse

Le domaine de la robotique aérienne est en plein essor grâce aux avancées technologiques des dernières décennies. Le problème de la commande de ces systèmes volants, comme les quadrirotors par exemple, aura constitué un vrai défi du fait de leur sous-actionnement en général, et des phénomènes aérodynamiques complexes impliqués. Il faut noter que les techniques de commande développées sont principalement axées sur la captation de données et la cartographie en environnement ouvert. Au cours des dernières années cependant, le champ de la robotique aérienne pour l'interaction physique s'est beaucoup développé. Ce type de scénario fait intervenir un robot aérien devant appliquer un effort maîtrisé sur un objet ou sur l'environnement, alors qu'il vole. Des avancées significatives ont été réalisées, au travers notamment de projets européens centrés sur cette thématique, comme [2, 3, 8, 4].

Dans ces deux grandes familles d'approches, qui s'intéressent donc aux déplacements libres sans contact *vs.* avec interaction physique, l'environnement est généralement traité comme une contrainte indésirée limitant les mouvements du robot. En d'autres termes, les tâches de déplacements libres/rapides et celles d'interactions physiques sécuritaires sont perçues comme antagonistes avec la présence d'éléments indésirés dans l'environnement. Une conséquence immédiate de ceci est que les mouvements réalisés avec contact sont peu dynamiques. En s'inspirant de l'utilisation des contacts faite en robotique humanoïde, nous proposons dans cette thèse d'exploiter le contact physique avec l'environnement dans le but de réaliser de la locomotion aérienne. Autrement dit, nous proposons de considérer l'environnement comme une source de contacts exploitables à des fins de locomotion, plutôt que comme une contrainte à éviter. Avec cette approche, nous souhaitons exploiter pleinement la dynamique des robots aériens en interaction physique.

Dans la première partie de cette thèse, nous détaillons les travaux réalisés relatifs à ce concept de locomotion aérienne. Cette idée est étudiée et démontrée au travers de simulations et expérimentations d'une nouvelle plate-forme robotique aérienne consistant en un quadrirotor équipé d'un bras robotique à un degré de liberté. Le premier chapitre détaille la dynamique particulière de ce système, qui comporte trois modes de fonctionnement différents. Les deux plus évidents correspondent aux dynamiques

en vol libre et en vol avec contact, tandis que le troisième correspond au comportement particulier lors d'une possible collision au moment d'établir le contact. Un modèle de collision est proposé pour comprendre et intégrer au mieux ce phénomène dans les développements consécutifs. Deux stratégies de commande sont également données afin de permettre à ce robot de suivre une trajectoire en temps-réel.

Dans le deuxième chapitre nous expliquons la démarche entreprise afin de mettre en place un algorithme d'optimisation de trajectoires adapté à ce système. Un cas élémentaire de locomotion est choisi, dans lequel le système doit exécuter une manœuvre partant d'une configuration initiale en contact avec l'environnement (le bras "s'accroche" à un premier point de pivot), et terminant dans une configuration avec un autre point d'attache plus loin, en passant par une phase intermédiaire de vol libre. La structure du planificateur de trajectoires développé lui permet d'intégrer le comportement dynamique complet du robot, y compris une possible collision comme évoqué précédemment, ce qui lui donne la possibilité de tirer au mieux parti de ses particularités. Une fonction de coût spécifique est également développée afin de garantir une précision maximale du robot dans la phase la plus critique. Des trajectoires sont ensuite générées pour différentes conditions, à savoir pour différentes limites d'actionnement et quelques variations de la fonction de coût minimisée. Les trajectoires obtenues sont ensuite analysées au regard de la tâche de locomotion considérée.

Pour terminer, le chapitre 3 expose les détails de conception et de réalisation d'un prototype en vue de tester les simulations précédentes. Un système d'attache magnétique est développé puis réalisé et testé, ce qui permet au robot de s'attacher au point de pivot dans des conditions propices à la locomotion. La phase de conception de ce sous-système est détaillée, notamment la modélisation de celui-ci qui a été réalisée en vue de simuler son comportement magnéto-mécanique et d'optimiser sa performance au vu des conditions. Le comportement thermique de ce sous-système est également modélisé afin d'assurer un fonctionnement nominal non destructif. Des éléments de conception mécanique sont également donnés.

Dans la seconde partie, nous nous intéressons au problème de la précision réalisable par un robot en suivi de trajectoire lorsque le modèle est incertain, et plus précisément lorsque les paramètres du modèle sont entachés d'erreur. Une généralisation de la fonction de coût spécifique introduite auparavant (au chapitre 2) est proposée, améliorant le concept tout en le rendant applicable à une large gamme de robots. En

effet, nous avons étudié le problème de la génération de trajectoires dont la sensibilité aux paramètres du modèle de robot considéré est minimale, dans le cas général. Ce type de trajectoires est généralisé à n'importe quel robot présentant une dynamique avec des incertitudes sur les paramètres, et se révèle particulièrement approprié dans le cas du quadrirotor étudié dans cette thèse, du fait de l'incertitude importante concernant ses paramètres inertiels et d'actionnement. Pour traiter ce problème, nous proposons donc la nouvelle notion de "sensibilité de l'état aux paramètres en boucle fermée" et nous montrons comment cette quantité peut être utilisée dans un contexte d'optimisation de trajectoires afin de générer des trajectoires dont la sensibilité aux paramètres est minimale, garantissant ainsi une forte robustesse.

Le chapitre 4 propose une première approche de ce concept pour produire des trajectoires dites à "sensibilité minimale", avec des analyses statistiques afin de tester l'efficacité de la méthode. Plusieurs cas sont considérés, faisant intervenir un robot mobile différentiel (*unicycle*) et un quadrirotor avec des lois de commande comprenant ou non un intégrateur, et avec une minimisation soit de la sensibilité de l'état final, soit de la sensibilité de l'état sur l'ensemble de la trajectoire (avec un coût intégral). Il en ressort que la méthode semble bien fonctionner en simulation pour les cas considérés, à savoir que les trajectoires générées donnent lieu à une erreur de suivi moindre par comparaison avec des trajectoires non optimisées ayant les mêmes conditions limites lorsque les paramètres du modèle sont mal calibrés.

Finalement, le chapitre 5 propose une généralisation de la méthode afin de produire une théorie plus générale, rigoureuse, et susceptible d'être appliquée plus largement à différents types de robots. Entre autres, la possibilité d'avoir une loi de commande imparfaite, c'est-à-dire, ici, qui n'est pas capable d'annuler l'erreur de suivi en dépit de paramètres bien calibrés et d'absence de perturbations, est intégrée dans la méthode. D'autres métriques issues de la sensibilité sont également calculées. En particulier, une utilisation de la sensibilité des entrées par rapport aux paramètres est proposée pour pallier leur manque de prédictibilité observé dans certains cas. Plusieurs analyses statistiques sont ainsi réalisées sur des simulations, avec des résultats validant là aussi les méthodes proposées en terme de réduction d'erreur. Enfin, une série d'expérimentations est menée sur un robot réel de type robot mobile différentiel (modèle Pioneer 3DX), validant également la méthode en améliorant les performances de suivi de trajectoire obtenues.

Contributions

Les éléments développés dans cette thèse ont fait l'objet de trois publications scientifiques listées ici :

(a) Q. Delamare, P. Robuffo Giordano, and A. Franchi, "Toward aerial physical locomotion : The contact-fly-contact problem," in IEEE Robotics and Automation Letters (RAL), vol. 3, no. 3, pp. 1514–1521, 2018, au sujet de la locomotion aérienne. Un algorithme de génération de trajectoires adapté à cette tâche particulière y est présenté et testé en simulation.

(b) P. Robuffo Giordano, Q. Delamare, and A. Franchi, "Trajectory generation for minimum closed-loop state sensitivity," in 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018, concernant la génération de trajectoires minimisant la sensibilité de l'état aux paramètres. Une analyse statistique y est présentée qui valide l'utilité et l'efficacité de l'algorithme présenté.

(c) N. Staub, M. Mohammadi, D. Bicego, Q. Delamare, H. Yang, D. Prattichizzo, P. Robuffo Giordano, D. Lee, A. Franchi, "The Tele-MAGMaS : an Aerial-Ground Co-manipulator System," in 2018 IEEE Robotics and Automation Magazine (RAM), vol. 25, no. 4, pp 66–75, 2018, qui présente un système de co-manipulation faisant intervenir un hexacoptère spécial et un manipulateur au sol en coopération pour mouvoir un objet.

Plusieurs vidéos ont été réalisées afin d'illustrer les résultats principaux associés à ces travaux, concernant la locomotion aérienne [1], la génération de trajectoires à sensibilité minimale [2], et le système Tele-MAGMaS présenté à la *Hannover Fair* à l'occasion des KUKA Innovation Awards [3] [4].

1. video at https ://proxy.ens-rennes.fr/owncloud/index.php/s/Nrx5Rmm9S93TsmF
2. video at https ://proxy.ens-rennes.fr/owncloud/index.php/s/nxFbMCRzD7wCzQp
3. video at https ://proxy.ens-rennes.fr/owncloud/index.php/s/KZgB2TCAYfoJ6wW
4. video at https ://proxy.ens-rennes.fr/owncloud/index.php/s/6ZBXsf9peCaHX52

REMERCIEMENTS

Je tiens à remercier pour commencer mes directeurs de thèse, Paolo et Antonio, qui ont su faire preuve au cours de ces trois années d'un enthousiasme à toute épreuve et d'une ingéniosité riche et insolite, ce qu'il me fallait. Je remercie également les membres de l'équipe Rainbow pour la bonne ambiance qui y règne. Un grand merci aux ingénieurs pour leur aide précieuse, Fabien, Pol, Thomas, ainsi qu'à Marie pour m'avoir permis d'accéder aux ateliers de l'INSA. Merci également à Tristan et Pascal pour les chouettes moments partagés durant leurs stages. Merci aussi aux collègues de l'ENS, qui m'ont accompagnés et soutenu dans cette aventure tant sur le plan scientifique que sur le plan de l'enseignement. Merci à tous ceux avec qui j'ai partagé de super sessions musicales.

Je remercie également mes proches pour leur soutien et leur patience (il en fallait). Merci à ma famille et Evelyne d'avoir été là plus que je n'ai été là pour eux. Je vous dois beaucoup. Merci aussi aux trois Fontaine de m'avoir abreuvé de moments de bonheur partagés, de randos (promis on l'aura ce vieux caillou), et autres repas à gogo. Merci à Simon pour ces 10 ans de bidouilles en tout genre (ce n'est que le début) et pour ces moments passés tous les 4, et merci aux Mektro12 pour le chemin parcouru ensemble.

Enfin je remercie Anaïs qui, au-delà de son indéfectible soutien, est le beurre salé de mes galettes, et avec qui je suis fier de partager la valse de la vie.

TABLE OF CONTENTS

INTRODUCTION

It is nowadays common to see quadrotor UAVs both in the domain of public entertainment with a large range of commercial products, as well as in professional applications such as remote inspection, cartography, agricultural spraying, or even filming. As well known, this flourishing of the field of aerial robotics has been made possible thanks to the common progress of on-board computational capabilities, together with the miniaturization of the associated electronics and power cells. Indeed, the mass/energy ratio of modern lithium-based batteries coupled with the high efficiency of synchronous motors allow the conversion of enough mechanical power to keep a multirotor aircraft in hover flight, for a non negligible duration. Moreover, the associated control algorithms have been heavily developed together with the computing capacity of on-board electronics.

As a result, current multirotor robots are more than ever autonomous, and present a huge potential for future robotic applications with complex tasks in complex environments. It is clear however that this kind of robotic platforms are challenging to control by nature, because of the complex aerodynamics of the rotating propellers (especially when close to a surface or in adverse wind conditions) and because of their freedom in the 3D environment. The possibilities are thus vast, but still a lot of challenges need be dealt with to make this kind of aerial robots able to handle complex real-world situations. To this extent, the field of aerial robotics have been developed a lot during the past decade, with substantial improvements in the control and localization of multirotor robots.

Context

The development and control of multirotor UAVs can be considered a practically solved problem for a wide range of navigation applications, mainly with surveillance or sensing tasks. Indeed, small aircrafts such as quadrotors are well suited to navigation tasks in tight or complex environments, because of their interesting agility and low price. Their relatively low weight associated with their important actuation capabilities

(in terms of force and torques) allows highly dynamical motions, thus such robots can achieve maneuvers that are adapted to difficult environments, for example in rescue or exploration applications.

Simultaneously, a number of recent works have also studied the possibility of having a contact between such a multirotor aircraft and its environment in the past years, which extends the robot applications a lot. The three main motivations for such studies are the ability of multirotors to

1. grasp some objects in the environment and displace them,
2. apply a force or, more generally, a wrench to some part of the environment such as a switch on a wall, a door handle to turn or some mechanism to screw (as can be found in the DARPA challenge for instance),
3. perform a proper (smooth) landing with special properties, e.g., land against a wall or on a moving platform [25, 46].

However, in these applications of aerial physical interaction, the structure of the environment is always a source of constraints that must be avoided in order to preserve both the integrity of the robot and of the environment. This is even more true for complex environments such as in indoor applications or in cluttered urban or industrial places, where possible collisions are numerous. As a consequence, in this kind of applications, the behavior of aerial robots is mostly bounded by the need to avoid or to master the contacts with the environment.

Overview of the state of the art

Aerial robotics

In this context, numerous works have been conducted to assess the problems of estimation and robust command of quadrotors. The specific dynamics of this kind of aerial robot have been studied and leveraged so that, with the proper framework, one can locate and control them in order to track desired trajectories, see e.g. [22, 36, 44, 37]. The problem of the servoing of these robots has also been studied with a lot of different approaches, e.g., [37, 14, 22, 15, 53], ranging from sliding mode to H-infinity or LQR. These works propose control algorithms that either focus on the robustness, or on the fastness of the control loop.

Note that in the vast majority of these works, the sensing relies on an external localization structure such as indoor motion capture in order to retrieve the location of the aircraft. A number of studies propose methods to improve the sensing component by means of on-board sensors (mainly vision) and relevant associated algorithms [27, 45, 56, 1]. More recently, we have also seen methods that are able to treat cases where only low quality information is available for sensing [52].

As stated before, the main applications of these works deal with navigation and data acquisition, which places the focus on the flight of the quadrotor itself. However, recent research has begun to leverage their operational potential in order to make them physically interact with their environment. With this kind of approach, the quadrotors can be used as aerial manipulators able to interact. The corresponding research field is called Aerial PHysical Interaction (APhI). For instance, [70, 54, 55] have explored the possibility for a multirotor equipped with an on-board 'manipulator' (active or passive) to achieve manipulation tasks. Interestingly, we have also seen a few new concepts in the past years that tend to push the boundaries of multirotor capabilities by means of original features, e.g., [71] which studied an original stabilizing fast perching, or [7] which proposed a multi-part aerial robot with variable configuration for grasping, but also [61] which propose a framework for aerial interaction that is based on a special hexarotor with tilted propellers, thus fully actuated.

In parallel, some works have been focused more on the high level interaction with an operator, e.g., [26, 60], proposing algorithms for conveying information between an operator and the actual contact at best. This area of research presents several challenges since translating the control methods from grounded robots with physical interaction to equivalent aerial situations is not straightforward. This field have also been supported through the past years by large international projects such as [2, 3, 8, 4], which have focused on complex scenarios that tend to be more realistic.

As a consequence, we see that the current state of the art allows one to achieve some aerial robotic task with interaction with the environment. However, in all these works the interaction remains either highly restrictive, or needs to be controlled to follow a mastered mechanical wrench. Therefore, in this thesis we propose to change this perspective by considering the environment as a source of possible contacts that can be leveraged for the sake of locomotion. Indeed, the aerial robot may be able to benefit from the way certain particular contacts affect its dynamics, and thus achieve complex movements that improve its maneuverability.

Figure 1 – Examples of modern applications of aerial robotics. On the left, a commercial product that is designed for waterproof rescue missions. On the middle, a foldable quadrotor able to adapt its shape from [23]. On the right, a cooperative framework sharing the task of moving an object between a grounded manipulator and an aerial robot, from [60].

Trajectory generation

The second topic we are interested in throughout this thesis is the generation of robotic trajectories, in particular for mobile robots. Indeed, the foreseen concept of *aerial* locomotion is closely related to the ability of our algorithms to plan trajectories that achieve the desired maneuvers. This kind of behavior also rely a lot on the ability of the robot to track precisely the planned trajectory via its controller.

Research has been conducted on these subjects over the years, that led to different kind of strategies for trajectory generation. One of the most used techniques for generating feasible trajectories that minimize some cost consists in leveraging the flatness of the robot, whenever this property is available. This concept was introduced in [24] and exploited for trajectory generation in [50]. Basically, featuring this property for a dynamical system means that it is possible to express both its input and output as functions of a certain 'flat output' (and possibly some of its derivatives). In practice, this allows to easily compute the state of the system given a certain trajectory of the flat output, by means of an algebraic relation and thus without needing to integrate the dynamics. This makes the trajectory generation much more efficient in theory. A known pitfall of this method, however, is that the expression for the input (as function of the flat output) is often highly non-linear and computationally heavy, and thus tend to cancel the benefit of the method in cases where the actuation is part of the optimized elements, e.g., in presence of actuation bounds. Moreover, this relation as well as the one for the state require an high accuracy in the model parameters and often include the position of some center of mass and inertia which are difficult parameters to measure.

The most common other methods for generating trajectories are simple direct transcription, *i.e.*, forward integration of the dynamics, but also direct collocation [9]. The method of the (orthogonal) collocation consists of using orthogonal polynomials to represent at the same time the input and output of the dynamics at knot points. Contrary to the flatness method or direct transcription where the dynamics is intrinsically respected by construction, here the relation between the input and output of the robot is enforced as a constraint at each considered knot point (but not between them in general). The precision of this approximation remains however controlled and can be arbitrarily reduced by increasing the number of considered knot points.

A number of works have applied these methods to multirotor UAVs in the past years, e.g., [57, 58, 67, 28, 52], with great success in the obtained performances. In all these studies, the achieved planned trajectories make it possible to optimize an objective (performance of a parameter estimation, observability) and/or respect special constraints (collision avoidance, actuation limits). However, we note that in these works the trajectory tracking remains decoupled from the planning stage. Hence, the achieved performance when tracking the planned trajectory is bound to the strict respect of the conditions that are envisioned for the planning. In particular, any discrepancy that was not modeled at the planning stage, *i.e.*, disturbances, unmodeled phenomena or poorly calibrated model, needs to be compensated by the real-time control loop, which will inevitably affect the tracking performance in an unmastered and potentially high proportion in case of inaccurate modeling.

Another kind of approach is to handle this difficulty by means of strategies that rely on a human-in-the-loop, see e.g. [42, 43, 41]. With such methods, some part of the task realization is given to the responsibility of the operator, which allows good compromises between the flexibility of the human operator and the achievable precision and strength of the robot.

Finally, a more recent class of strategies try to integrate the behavior of the control-loop at the planning stage in order to build ‘control-aware‘ schemes that leverage at most the information of the models, like in [28, 6, 21]. In this last kind of works, the way that the controller will behave during execution of the trajectory is taken into account from the planning stage. This approach makes the complete stack planner–controller more tightly coupled by making the planner ‘aware‘ of the real-time control loop.

In this thesis we will propose a control-aware trajectory planning framework that is adapted to robots with uncertain parameters, such as aerial robots which feature

complex aerodynamics that are difficult to model and compensate precisely.

Thesis contributions

In this thesis we develop strategies for trajectory generation, with the purposes of improving aerial locomotion capabilities, and improving the tracking of trajectories for systems that feature poorly known parameters. In particular, we focus on the two main issues that are raised when considering aerial locomotion, which are

1. the generation and tracking of trajectories for an aerial robot with switching dynamics,
2. the generation of trajectories that are robustly tracked in presence of model parameter uncertainties.

The study of these thematics led to a new trajectory planning algorithm for aerial locomotion, described in

1 Q. Delamare, P. Robuffo Giordano, and A. Franchi, "Toward aerial physical locomotion: The contact-fly-contact problem," in IEEE Robotics and Automation Letters (RAL), vol. 3, no. 3, pp. 1514–1521, 2018.

A video demonstrating simulations of the resulting trajectories for aerial locomotion is available [5]. A prototype have also been realized in the course of the thesis, including a magnetic hooking system and the robotic platform of the MonkeyRotor, which will allow further explorations of the concept of aerial locomotion.

Then, we developed a novel trajectory optimization framework for mobile robots with uncertain models, based on sensitivity metrics, which led to the following contribution:

2 P. Robuffo Giordano, Q. Delamare, and A. Franchi, "Trajectory generation for minimum closed-loop state sensitivity," in 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018,

This work was synthesized in a video showing the improvement in the tracking performance when using a trajectory that is optimized w.r.t. the closed-loop state sensitivity, for a unicycle and a quadrotor [6]. It also led to further developments and validations of this theory as described in the last chapter of thesis, including new metrics of interest based on the sensitivity.

5. video at https://proxy.ens-rennes.fr/owncloud/index.php/s/Nrx5Rmm9S93TsmF
6. video at https://proxy.ens-rennes.fr/owncloud/index.php/s/nxFbMCRzD7wCzQp

Finally, in relation with these two fields, we also participated in the development of the Tele-MAGMaS project presented at the Kuka Innovation Awards 2017 (finalist), which led to the following paper:

3 N. Staub, M. Mohammadi, D. Bicego, Q. Delamare, H. Yang, D. Prattichizzo, P. Robuffo Giordano, D. Lee, A. Franchi, "The Tele-MAGMaS: an Aerial-Ground Co-manipulator System," in 2018 IEEE Robotics and Automation Magazine (RAM), vol. 25, no. 4, pp 66–75, 2018.

A video of the realized simulation framework is available [7], as well as a video of the Hannover Fair demonstration [8].

Thesis structure

This thesis is split in two main parts. The first one, Part I, is dedicated to the study of the MonkeyRotor, a robot concept whose goal is to evaluate the properties and benefits of aerial locomotion. The second part, Part II, contains the theoretical development and validations of a new trajectory generation framework that aims at improving the tracking performances of robots — especially when subject to uncertainties in their model parameters.

Outline of Part I

In this part, we develop the contact-fly-contact problem which is a case-study of aerial locomotion. The MonkeyRotor is introduced as the aerial robot dedicated to the study of this problem.

Chapter 1 provides an analysis of the particular dynamics and control of the MonkeyRotor. The specificities of this robot and how they can be leveraged in the context of aerial locomotion are discussed.

Chapter 2 details the trajectory generation algorithm that was designed for this system. We show the important features of the realized aerial locomotion, with an analysis of the resulting generated trajectories among different planning conditions.

Chapter 3 gives the details about the realization of a prototype of the MonkeyRotor. A novel magnetic hooking system is designed and realized, that makes it possible for

7. video at https://proxy.ens-rennes.fr/owncloud/index.php/s/KZgB2TCAYfoJ6wW
8. video at https://proxy.ens-rennes.fr/owncloud/index.php/s/6ZBXsf9peCaHX52

the system to alternate between states with and without contact as wished.

Outline of Part II

This part is dedicated to the novel algorithm for the generation of ‘minimum sensitive‘ trajectories.

Chapter 4 proposes a method that allows to generate trajectories resulting in a minimization of the tracking error that are due to erroneous calibration of the model parameters. A statistical analysis is conducted which validates the soundness of the concept, based on a Monte Carlo simulation campaign.

Chapter 5 provides a generalization of the theory to robots that have controllers with arbitrary tracking performance (including lag or filtering behaviors). Other sensitivity metrics of interest are also proposed which improve the robustness of the overall task realization. The concept is validated through large scale statistical analysis and real experiment on a unicycle.

PART I

Part I

CHAPTER 1

MONKEYROTOR CONCEPT AND ANALYSIS

1.1 Introduction

In this chapter we will describe the theoretical study of a new robotic concept. As stated in the main introduction, the starting point lies in the observation that in aerial robotics the environment is usually considered as an obstacle to be avoided, or more generally, as a constraint. Conversely, in this Thesis we focus on the possibility of exploiting a contact between the aerial robot and the environment for the sake of enhancing the navigation capabilities.

To do so, we isolate a particular case-study which consists in making a quadrotor equipped with an arm able to locomote under two pivot points. More precisely, the goal of such a robot is to navigate through its environment not only by means of its own ability to fly, but also with phases where a physical contact with the environment happens and is leveraged to the benefit of the overall maneuver. This aerial robot thus has a particular dynamics because of its ability to be in physical contact with its environment. Figure 1.1 illustrates a possible depiction of this system when realizing a maneuver that utilizes a contact with a pivot point.

1.2 Dynamical modeling

The MonkeyRotor consists of a quadrotor UAV equipped with an actuated 1-DOF arm meant to grasp a pivot point (e.g., a branch) in the environment with its end-effector. In this section we illustrate the dynamical model of the MonkeyRotor during the two phases, *i.e.*, hooked and free-flight, by borrowing from the previous works [69, 65] which have considered similar scenarios. In particular, [69] has considered a quadrotor

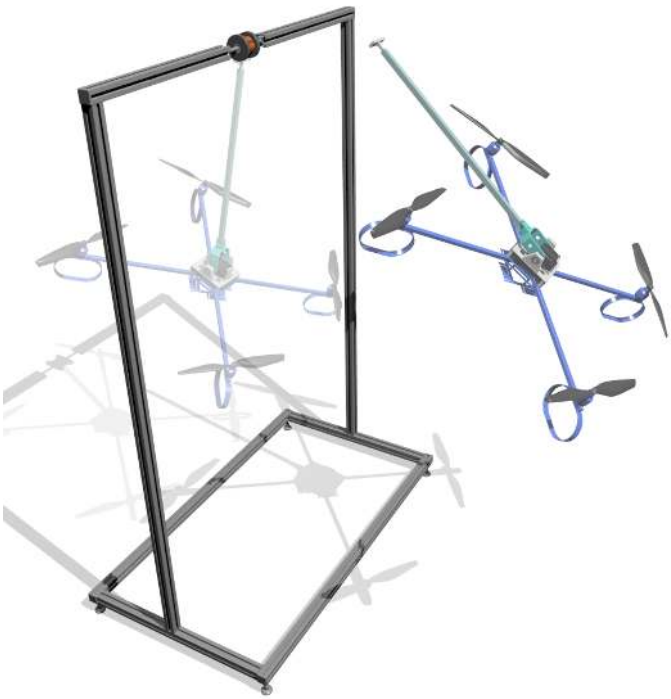

Figure 1.1 – Illustration of the concept of the MonkeyRotor: a quadrotor equipped with an arm which leverages a pivot point in the environment in order to achieve aerial locomotion.

with actuated arm but only in free-flight, while [65] has considered the hooked case but with a passive arm. As already done in many previous works on similar subjects, see, e.g., [29, 64, 49, 69, 65], we restrict the analysis to the vertical plane.

1.2.1 Definitions

With reference to Fig. 1.2, let $\mathcal{F}_W$ be an inertial world frame with axes $\{\mathbf{x}_W, \mathbf{z}_W\}$ and origin $\mathbf{O}_W$, and $\mathcal{F}_B$ a body frame attached to the quadrotor with axes $\{\mathbf{x}_B, \mathbf{z}_B\}$: the axis $\mathbf{z}_B$ represents the body-frame thrust direction, and the origin $\mathbf{O}_B$ is placed at the quadrotor center of mass (CoM). The configuration of the quadrotor can be specified by the position of $\mathbf{O}_B$ in $\mathcal{F}_W$, denoted as $\mathbf{p}_B = [x_B \; z_B]^T \in \mathbb{R}^2$, and the orientation of $\mathcal{F}_B$ w.r.t. $\mathcal{F}_W$ here parametrized by the angle θ_B from $\mathbf{z}_W$ to $\mathbf{z}_B$.

The arm is assumed to have the length L_1 and to have its joint mounted at the

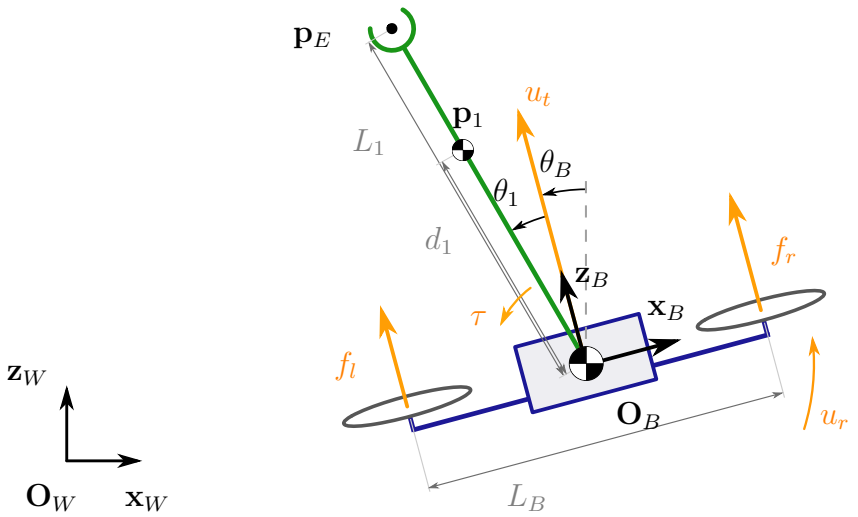

Figure 1.2 – Geometry of the MonkeyRotor, a flying robot with an actuated arm.

quadrotor CoM $\mathbf{p}_B$, around which it can rotate by an angle θ_1 defined as the angle from $\mathbf{z}_B$ to the arm direction. The CoM of the arm, denoted as $\mathbf{p}_1$, is placed at a distance d_1 from $\mathbf{O}_B$. The configuration of the whole MonkeyRotor (quadrotor + arm) is then denoted as $\mathbf{q} = [\mathbf{p}_B^T\ \boldsymbol{\theta}^T]^T \in \mathbb{R}^4$ where we let $\boldsymbol{\theta} = [\theta_B\ \theta_1]^T$.

The quadrotor is equipped with two propellers generating two thrust vectors $f_l\mathbf{z}_B$ and $f_r\mathbf{z}_B$: the forces produced by the propellers result in a total thrust vector $u_t\mathbf{z}_B = (f_r + f_l)\mathbf{z}_B$ and torque $u_r = \frac{L_B}{2}(f_r - f_l)$, with L_B being the distance between the two propellers. The arm is also assumed actuated by a torque τ acting at $\mathbf{O}_B$. These three inputs for the whole MonkeyRotor are then denoted as $\mathbf{u} = [u_t\ u_r\ \tau]^T \in \mathbb{R}^3$. For convenience, we also define the alternative input vector $\mathbf{u}_f = [f_r\ f_l\ \tau]^\top = \mathbf{K}\mathbf{u}$ where

$$\mathbf{K} = \begin{bmatrix} 1/2 & 1/L_B & 0 \\ 1/2 & -1/L_B & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{1.1}$$

Indeed, while the MonkeyRotor dynamics are more naturally expressed in terms of the input vector $\mathbf{u}$, the physical actuation constraints, *i.e.*, min and max joint torque and propeller thrusts, affect the input $\mathbf{u}_f$. This distinction will be important in the next developments. We finally let m_B, J_B, m_1, J_1 be the mass and inertia of the quadrotor and arm, respectively.

We now describe the dynamical model of the MonkeyRotor in the two considered phases of hooked and free-flight.

1.2.2 Hooked phase

Let

$$\mathbf{p}_E = \mathbf{p}_B + L_1 \begin{bmatrix} -\sin(\theta_1 + \theta_B) \\ \cos(\theta_1 + \theta_B) \end{bmatrix} \tag{1.2}$$

represent the position of the arm end-effector in $\mathcal{F}_W$ and $\mathbf{p}_E^* \in \mathbb{R}^2$ the (fixed) position of the hook in $\mathcal{F}_W$. Following [65], the hook constraint $\mathbf{p}_E(\mathbf{q}) = \mathbf{p}_E^*$ restricts the MonkeyRotor motion to a circle centered at $\mathbf{p}_E^*$. In this constrained case the MonkeyRotor configuration is fully determined by the configuration variables $\boldsymbol{\theta}$: by applying standard techniques (Euler-Lagrange procedure), one can then obtain the following (reduced) dynamical model governing the behavior of the states $(\boldsymbol{\theta},\, \dot{\boldsymbol{\theta}})$

$$\mathbf{M}_h(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{g}_h(\boldsymbol{\theta}) = \mathbf{G}_h(\boldsymbol{\theta})\mathbf{u} \tag{1.3}$$

where

$$\mathbf{M}_h(\boldsymbol{\theta}) = \begin{bmatrix} J_B & 0 \\ 0 & J_1 + m_B L_1^2 + m_1(L_1 - d_1)^2 \end{bmatrix}, \tag{1.4}$$

$$\mathbf{G}_h(\boldsymbol{\theta}) = \begin{bmatrix} 0 & 1 & -1 \\ L_1 \sin(\theta_1) & 0 & 1 \end{bmatrix}, \tag{1.5}$$

and $\mathbf{g}_h(\boldsymbol{\theta}) = [0\ (m_B L_1 + m_1(L_1 - d_1))g\sin(\theta_B + \theta_1)]^\top$. Since the matrix $\mathbf{G}_h(\boldsymbol{\theta})$ is always full rank, the hooked MonkeyRotor is overactuated, with two controlled variables $\boldsymbol{\theta}$ for the three control inputs of $\mathbf{u}$. We note that in [65] the joint arm was considered passive, *i.e.*, $\tau = 0$ thus resulting in a fully-actuated system with a singularity for $\theta_1 = 0$ as opposed to the case under consideration which is singularity-free. Sect. 1.3.1 will elaborate more about the possible use of the MonkeyRotor overactuation.

The behavior of the remaining MonkeyRotor states $(\mathbf{p}_B,\, \dot{\mathbf{p}}_B)$ can then be algebraically expressed as a function of $\boldsymbol{\theta}$ and $\dot{\boldsymbol{\theta}}$ by exploiting the hook constraint $\mathbf{p}_E(\mathbf{q}) = \mathbf{p}_E^*$ as

$$\mathbf{p}_B = \mathbf{p}_E^* - L_1 \begin{bmatrix} -\sin(\theta_1 + \theta_B) \\ \cos(\theta_1 + \theta_B) \end{bmatrix} \tag{1.6}$$

and

$$\dot{\mathbf{p}}_B = L_1(\dot{\theta}_1 + \dot{\theta}_B) \begin{bmatrix} \cos(\theta_1 + \theta_B) \\ \sin(\theta_1 + \theta_B) \end{bmatrix}. \tag{1.7}$$

1.2.3 Free-flying phase

The free-flying dynamical model of the MonkeyRotor is a particular case of the system presented in [69]. In particular one has

$$\mathbf{M}_f(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{c}_f(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}_f(\mathbf{q}) = \mathbf{G}_f\mathbf{u}, \tag{1.8}$$

where the expression of the various terms are given in [69].

We note that, as opposed to the hooked scenario, the MonkeyRotor is *underactuated* during free-flight (three inputs $\mathbf{u}$ for four configuration variables $\mathbf{q}$). However, as discussed in [69], it is possible to find a *flat output* or *linearizing output* [24] which allows for full dynamic linearization of the system dynamics. More details about this point are given in Sect. 1.3.2.

1.2.4 Impact Model

One particularity of the concept of aerial locomotion is that the contact between the end-effector of the robot and the pivot point is possibly accompanied by a voluntary shock, which means that the hooking event itself may not be smooth for the sake of the global maneuver. Indeed, let t_h be the time at which the MonkeyRotor switches from a free-flight phase to a hooked phase because the end-effector has reached the pivot location $\mathbf{p}_E^*$ and performed a successful hook. If $\dot{\mathbf{p}}_E(t_h^-) \neq \mathbf{0}$, *i.e.*, the end-effector velocity is non-zero just before hooking, a sudden impact will occur affecting the evolution of the MonkeyRotor state $(\mathbf{q}, \dot{\mathbf{q}})$.

Therefore, the goal of this section is to propose a simple impact model based on impulse theory, see, e.g., [47] able to capture the instantaneous change from $\dot{\mathbf{q}}(t_h^-)$ to $\dot{\mathbf{q}}(t_h^+)$ because of a possible collision between the end-effector and the hook. As customary, we assume continuity of $\mathbf{q}$, *i.e.*, $\mathbf{q}(t_h^-) = \mathbf{q}(t_h^+)$, in presence of an instantaneous impact [47]. The availability of this impact model will then allow us to have a complete model of the MonkeyRotor full dynamics that integrates the effects of a possible collision. In a trajectory planning context, as it will be done later in this Thesis,

this allows the planner to be 'aware' of the possible collision, and thus generate more realistic motion plans that can also take advantage of this . For example, as far as the feasible trajectory space is large enough, the collision between end-effector and pivot can be controlled by the trajectory planner for quickly reducing the system kinetic energy. Conversely, the magnitude of the impact can also be reduced in the same way if necessary.

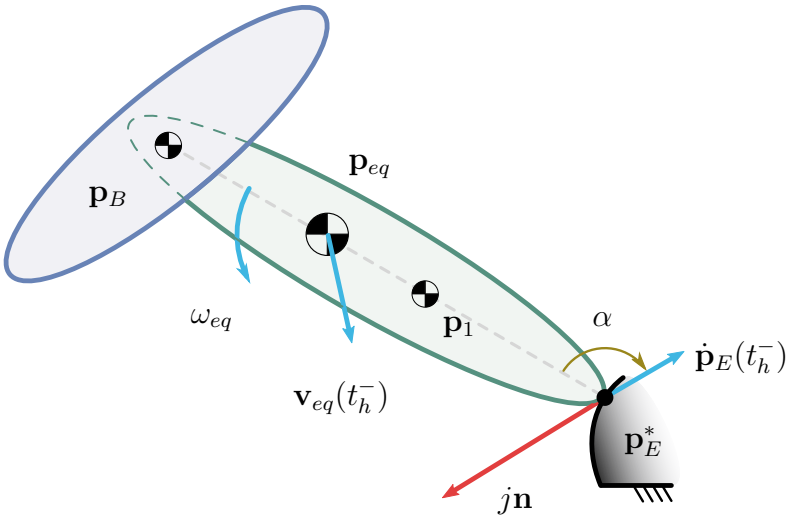

Figure 1.3 – Notations for the collision model. The large and 'instantaneous' reaction force at E is synthesized in the impulse vector $j\mathbf{n}$.

Recalling that $\dot{\mathbf{q}} = [\dot{\mathbf{p}}_B^T \ \dot{\boldsymbol{\theta}}^T]^T$, we first consider the effects on the two angular velocities $\dot{\boldsymbol{\theta}} = [\dot{\theta}_B \ \dot{\theta}_1]^T$. First of all, we remark that the choice of placing the joint base at the quadrotor CoM — a property also known as *protocentricity* [66] — implies that the rotational dynamics of the quadrotor base is completely decoupled from the dynamics of the collision. Indeed, the efforts that are transmitted through the arm joint are only linear forces (no torque), which do not generate any torque on the quadrotor base as they are directly applied to its CoM without offset. Therefore one has $\dot{\theta}_B(t_h^+) = \dot{\theta}_B(t_h^-)$, *i.e.*, the rotational velocity of the quadrotor base is not affected by the impact. Concerning the angular velocity of the arm after the impact, we compute it by assimilating the MonkeyRotor to an equivalent body with the following properties for the sake of impact modeling:

— mass $m_{eq} = m_B + m_1$

— CoM $\mathbf{p}_{eq} = \dfrac{m_B\mathbf{p}_B + m_1\mathbf{p}_1}{m_{eq}}$

— inertia $J_{eq} = J_1 + m_1\|\mathbf{p}_1 - \mathbf{p}_{eq}\|^2 + m_B\|\mathbf{p}_B - \mathbf{p}_{eq}\|^2$,

and with an equivalent linear velocity $\mathbf{v}_{eq} = \dot{\mathbf{p}}_{eq}$ and the total absolute angular velocity $\omega_{eq} = \dot{\theta}_B + \dot{\theta}_1$.

During the short time interval $\delta t = t_h^+ - t_h^-$, a force $\mathbf{F_c}$ is applied by the pivot to the end-effector of the arm because of the collision. One can define the impulse vector $\mathbf{j}$, which is the total momentum exchanged by the end-effector and the pivot during this impact:

$$\begin{aligned}\mathbf{j} = \int_{\delta t} \mathbf{F_c} dt &= j\mathbf{n} \\ &= -j\frac{\dot{\mathbf{p}}_E(t_h^-)}{\|\dot{\mathbf{p}}_E(t_h^-)\|}\end{aligned} \tag{1.9}$$

with $\mathbf{n}$ the unit vector defining the direction of the impulse $\mathbf{j}$, see Fig. 1.3. The duration of the collision is small enough for us to consider that this direction is given by the velocity before the impact $-\mathbf{p}_E(t_h^-)$. Therefore, the direction $\mathbf{n}$ is determined by the MonkeyRotor state at t_h^-.

This quantity can be used for determining the precise effect of the impact. Indeed, the change in the linear and angular velocities $\mathbf{v_{eq}}$ and ω_{eq} before and after the collision can be modeled with

$$\begin{cases} m_{eq}(\mathbf{v}_{eq}(t_h^+) - \mathbf{v}_{eq}(t_h^-)) = \mathbf{j} \\ J_{eq}(\omega_{eq}(t_h^+) - \omega_{eq}(t_h^-)) = (\mathbf{p}_E^* - \mathbf{p}_{eq}) \times (j\mathbf{n}) \\ \qquad\qquad\qquad\qquad = j\|\mathbf{p}_E^* - \mathbf{p}_{eq}\| \sin\alpha \end{cases} \tag{1.10}$$

where $\mathbf{p}_E^*$ is the location of the pivot point where the collision occurs and α is the angle between vectors $\mathbf{p}_E^* - \mathbf{p}_{eq}(t_h^-)$ and $\mathbf{n}$. Thus we get that

$$\begin{cases} \mathbf{v}_{eq}(t_h^+) = \mathbf{v}_{eq}(t_h^-) + \dfrac{\mathbf{j}}{m_{eq}} \\ \omega_{eq}(t_h^+) = \omega_{eq}(t_h^-) + \dfrac{j}{J_{eq}}\|\mathbf{p}_E^* - \mathbf{p}_{eq}\| \sin\alpha \end{cases}. \tag{1.11}$$

Moreover, one has the kinematics relationships

$$\begin{cases} \mathbf{v}_{eq}(t_h^+) = \dot{\mathbf{p}}_1(t_h^+) + \mathbf{S}(\omega(t_h^+)) \cdot (\mathbf{p}_1 - \mathbf{p}_{eq}) \\ \dot{\mathbf{p}}_E(t_h^+) = \dot{\mathbf{p}}_1(t_h^+) + \mathbf{S}(\omega(t_h^+)) \cdot (\mathbf{p}_1 - \mathbf{p}_E) \end{cases} \tag{1.12}$$

where $\mathbf{S}(a) = \begin{bmatrix} 0 & a \\ -a & 0 \end{bmatrix} \in \mathbb{R}^{2\times 2}$.

Then, by combining eq. (1.11) with the kinematics of eq. (1.12), and by using the fact that the end-effector velocity is zero after the impact, *i.e.*, $\dot{\mathbf{p}}_E(t_h^+) = \mathbf{0}$, one can solve for the impulse norm $j = \|\mathbf{j}\|$ as

$$j = \frac{m_{eq}\,\|\dot{\mathbf{p}}_E(t_h^-)\|}{1 + \dfrac{m_{eq}\,\|\mathbf{p}_{eq}(t_h^-) - \mathbf{p}_E^*\|^2}{J_{eq}}\sin\alpha}\,. \tag{1.13}$$

Note that j can be expressed in terms of only known quantities, in particular the MonkeyRotor state $(\mathbf{q}(t_h^-), \dot{\mathbf{q}}(t_h^-))$ just before the collision. Therefore, plugging (1.13) in (1.11) yields the value of $\omega_{eq}(t_h^+) = \dot{\theta}_B(t_h^+) + \dot{\theta}_1(t_h^+)$, which in turn determines $\dot{\theta}_1(t_h^+)$ since, as explained before, $\dot{\theta}_B(t_h^+)$ is known. Having obtained $\dot{\theta}_B(t_h^+)$ and $\dot{\theta}_1(t_h^+)$, the relationship (1.7) finally allows us to determine the remaining $\dot{\mathbf{p}}_B(t_h^+)$ and, thus, the whole vector $\dot{\mathbf{q}}(t_h^+)$ as sought.

We observe that the obtained expression for the impulse norm j is such that 1) if the velocity of the end-effector before the impact is null, *i.e.*, the hooking is done in a perfectly smooth way, then the impact has no effect, and, 2) the impulse is greater when the angle α is closer to zero, *i.e.*, when the arm arrives frontally towards the pivot.

Note that no parameter — like elasticity or any other mechanical property related to the materials — was required in this modeling of the collision, which makes it independent from the mechanical implementation of the end-effector, and from the detailed characteristics of the pivot. Indeed, the possible loss of kinetic energy that occurs with this impact event is purely linked to the direction of the velocity w.r.t. the target pivot.

1.3 Flight control

In this section we propose two control laws that allow the system to track some desired outputs both in the hooked and free-flying phases. The desired output to be tracked and their derivatives may be computed as trajectories in a prior planning stage, as it will be discussed in the next chapter. Then, the tracking policies that are described here compute a real-time input $\mathbf{u}$ for the system dynamics with the goal of bringing its output as close as possible to the desired one, even in the presence of perturbation.

1.3.1 Hooked phase

The goal of the control in the hooked phase is to let the MonkeyRotor configuration $\boldsymbol{\theta}$ track the reference optimal trajectory $\boldsymbol{\theta}^*(t)$ generated by the planning algorithm. This can be accomplished by implementing a static feedback linearization of the MonkeyRotor constrained dynamics (1.3)

$$\mathbf{u} = \mathbf{G}_h^{\dagger}(\boldsymbol{\theta})(\mathbf{M}_h(\boldsymbol{\theta})\boldsymbol{\nu} + \mathbf{g}_h(\boldsymbol{\theta})) + \lambda \mathbf{n}_h \tag{1.14}$$

where the $.^{\dagger}$ operator indicates the usual Moore-Penrose pseudoinverse, $\lambda \in \mathbb{R}$ is a scalar gain and

$$\mathbf{n_h} = \begin{bmatrix} 1 \\ -L_1 \sin(\theta_1) \\ L_1 \sin(\theta_1) \end{bmatrix} \tag{1.15}$$

is a vector spanning the one-dimensional null-space of matrix $\mathbf{G}_h$ (due to the MonkeyRotor overactuation during the hooked phase).

By plugging (1.14) into (1.3), one then obtains the linearized dynamics $\ddot{\boldsymbol{\theta}} = \boldsymbol{\nu}$ which can be stabilized along the reference trajectory $\boldsymbol{\theta}^*(t)$ by choosing

$$\boldsymbol{\nu} = \ddot{\boldsymbol{\theta}}^* + k_d(\dot{\boldsymbol{\theta}}^* - \dot{\boldsymbol{\theta}}) + k_p(\boldsymbol{\theta}^* - \boldsymbol{\theta}) \tag{1.16}$$

where $k_d > 0$ and $k_p > 0$ are suitable gains.

As well-known, setting $\lambda = 0$ in (1.14) yields the minimum-norm solution for vector $\mathbf{u}$. However, the null-space term $\lambda \mathbf{n}_h$ can be exploited for accomplishing a secondary objective besides the tracking of $\boldsymbol{\theta}^*(t)$. In our case, we chose to exploit this term for coping, as much as possible, with the actuation constraints:

$$\underline{\mathbf{u}}_f \leq \mathbf{u}_f \leq \bar{\mathbf{u}}_f. \tag{1.17}$$

This is obtained as follows: by rewriting (1.14)–(1.16) as $\mathbf{u} = \mathbf{u}^* + \lambda \mathbf{n}_h$, we seek the optimal value λ^* solving this linear minimization problem

$$\begin{aligned} \lambda^* = & \arg\min |\lambda| \\ & \text{s.t.} \quad \underline{\mathbf{u}}_f \leq \mathbf{K}\mathbf{u}^* + \lambda \mathbf{K}\mathbf{n}_h \leq \bar{\mathbf{u}}_f. \end{aligned} \tag{1.18}$$

If a solution exists, then setting $\lambda = \lambda^*$ in (1.14) will guarantee fulfilment of the tracking

task and, at the same time, of the actuation constraints with the smallest possible norm for the control input $\mathbf{u}$. In case (1.18) does not admit a solution, no control action can meet the constraints while realizing the tracking task. In this case the input vector $\mathbf{u}_f$ is simply saturated. We note that this case is quite unlikely to occur in practice since the trajectory to be tracked $\boldsymbol{\theta}^*(t)$ is already compliant "by construction" with the actuation constraint. Any additional control authority needed to recover possible perturbations and disturbances during the flight can then be typically accommodated by exploiting the null-space term $\lambda^* \mathbf{n}_h$.

Note that in the case where we only seek a value for λ that makes the input respect the bounds without considering the cost minimization, one can solve analytically the corresponding problem. Indeed, the following equivalence holds:

$$\begin{cases} \underline{u_1} \leq \lambda \cdot n_{h1} + u_{c1} \leq \overline{u_1} \\ \underline{u_2} \leq \lambda \cdot n_{h2} + u_{c2} \leq \overline{u_2} \\ \underline{u_3} \leq \lambda \cdot n_{h3} + u_{c3} \leq \overline{u_3} \end{cases} \iff \lambda_{min} \leq \lambda \leq \lambda_{max}$$

where

$$\begin{cases} \lambda_{min} = \max(\min(\dfrac{\underline{u_i} - u_{ci}}{n_{hi}}, \dfrac{\overline{u_i} - u_{ci}}{n_{hi}}), \forall i \in [\![1,3]\!] \\ \lambda_{max} = \min(\max(\dfrac{\underline{u_i} - u_{ci}}{n_{hi}}, \dfrac{\overline{u_i} - u_{ci}}{n_{hi}}), \forall i \in [\![1,3]\!] \end{cases}$$

Hence, by calculating the values of λ_{min} and λ_{max}, a range is determined for λ that guarantees that the inputs lie in their bounds. Among this range, we can then choose, e.g., the smaller λ in absolute value, which corresponds to minimizing the growth of the input norm implied by this null-space exploitation. In the case where $\lambda_{min} > \lambda_{max}$, there is no solution and the input must be truncated.

1.3.2 Free-flying phase

As explained in Sect. 1.2.3, during free-flight the MonkeyRotor is underactuated but one can still achieve full dynamical linearization of its dynamics by acting on a suitable flat/linearizing output. In short, this is obtained as follows: let $\theta_{1B} = \theta_1 + \theta_B$, define $\mathbf{y}(\mathbf{q}) = [\mathbf{p}_B^T \ \theta_{1B}]^T \in \mathbb{R}^3$ as the flat/linearizing output and let $\mathbf{y}^*(t)$ be the corresponding reference optimal trajectory generated by a trajectory planner such as the one which will be presented in Sect. 2.2. Let also $\bar{\mathbf{u}} = [\ddot{u}_t \ u_r \ \ddot{\tau}]^T$ be the new (extended) input vector, where two integrators have been placed on both the u_t and τ original inputs.

The new extended state which includes the dynamic extensions of the original inputs is then denoted as $\bar{\mathbf{x}} = [\mathbf{p}_B^T\ \dot{\mathbf{p}}_B^T\ \boldsymbol{\theta}^T\ \dot{\boldsymbol{\theta}}^T\ u_t\ \dot{u}_t\ \tau\ \dot{\tau}]^T \in \mathbb{R}^{12}$. With these settings, one can show (see [69]) that differentiating the flat output $\mathbf{y}$ four times yields

$$\ddddot{\mathbf{y}} = \bar{\mathbf{f}}(\bar{\mathbf{x}}) + \bar{\mathbf{A}}(\bar{\mathbf{x}})\bar{\mathbf{u}} \tag{1.19}$$

where $\bar{\mathbf{A}}(\bar{\mathbf{x}})$ is a square nonsingular matrix as long as $u_t \neq 0$. System (1.19) can then be inverted by choosing $\bar{\mathbf{u}} = \bar{\mathbf{A}}(\bar{\mathbf{x}})^{-1}(\bar{\boldsymbol{\nu}} - \bar{\mathbf{f}}(\bar{\mathbf{x}}))$. Tracking of the optimal trajectory $\mathbf{y}^*(t)$ is then obtained by choosing, as usual,

$$\bar{\boldsymbol{\nu}} = \ddddot{\mathbf{y}}^* + k_1(\dddot{\mathbf{y}}^* - \dddot{\mathbf{y}}) + k_2(\ddot{\mathbf{y}}^* - \ddot{\mathbf{y}}) + k_3(\dot{\mathbf{y}}^* - \dot{\mathbf{y}}) + k_4(\mathbf{y}^* - \mathbf{y}) \tag{1.20}$$

where $k_1,\ k_2,\ k_3,\ k_4 > 0$ are suitable gains.

1.4 Validation of the Control Strategy

In order to test the validity of the proposed dynamics and control laws derived in the previous sections, we conducted simple simulations of the system in the two situations: hooked and free-flying. For the two cases, we design a simple polynomial trajectory for the desired output which allows us to derive the analytical expressions for the time derivatives. In this Thesis we mostly use polynomials for the trajectory representation.

Let γ be a representation function for the trajectory, which transforms a finite vector of coefficients $\mathbf{a}$ and a current time t into an evaluation of the corresponding trajectory at t. For a unidimensional trajectory $y^*(t) \in \mathbb{R}$, this means that the polynomial representation translates into the following expression

$$y^*(t) = \gamma(\mathbf{a}, t) = \sum_{i=0}^{n_a-1} a_{i+1} \left(\frac{t}{t_f}\right)^i \tag{1.21}$$

where the order of the polynomial is $n_a - 1$, and where t_f is the duration of the trajectory (5 s here). Extending to multiple dimensions, *i.e.*, $\mathbf{y}^*(t) \in \mathbb{R}^{n_y}$, is as simple as

duplicating the expression for each coordinate, which can be written

$$\mathbf{y}^*(t) = \gamma(\mathbf{a}, t) = \sum_{i=0}^{N-1} \begin{bmatrix} a_{i+1} \\ a_{i+1+N} \\ \dots \\ a_{i+1+N(n_y-1)} \end{bmatrix} \left(\frac{t}{t_f}\right)^i \tag{1.22}$$

where $N-1$ is the order of the polynomials, such that $n_a = N n_y$.

This definition of the trajectory also allows us to easily construct a vector of polynomial coefficients $\mathbf{a}$ which respects initial and final constraints synthesized in a vector $\mathbf{d}$, by means of the linear relation

$$\mathbf{a} = \mathbf{M}_i \, \mathbf{d} \tag{1.23}$$

where $\mathbf{M}_i$ is a simple matrix that only depends on the duration t_f.

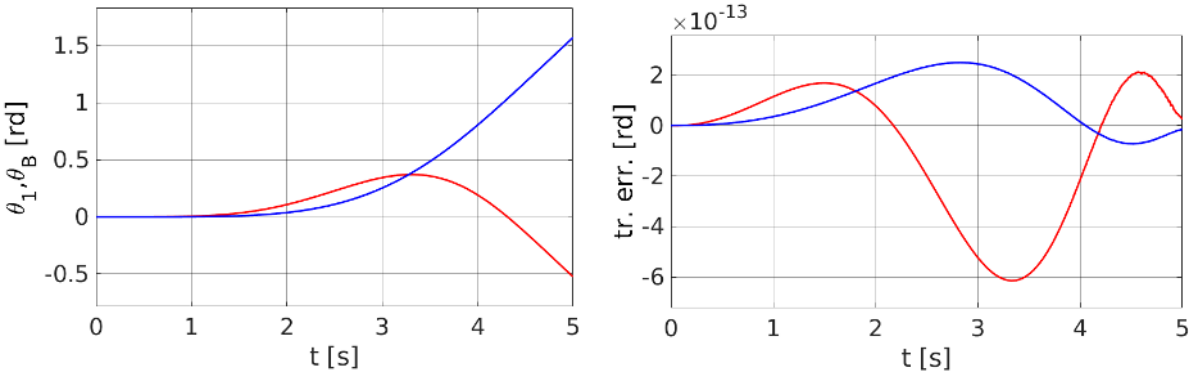

Figure 1.4 – Simulation of the hooked dynamics of the MonkeyRotor. On the left, the realized angles θ_B, θ_1. On the right, the corresponding tracking errors: the controller perfectly tracks the trajectory in these ideal conditions.

For the hooked phase, we test the control law on a simple trajectory that begins with angles $[0, 0]$ and ends at $[-\pi/6, \pi/2]$ rad. The final angular velocities arbitrary are set to $[-\pi/4, \pi/4]$ rad/s, while the initial ones and other derivatives are set to zero. We observe on Fig. 1.4 that the tracking task is realized as expected, with decoupled dynamics for the two angles as wished. The tracking error is of the order of numerical precision of the solver, which means that the controller was able to perfectly track the desired trajectory. This is of course possible because the parameters of the system are perfectly known, and there is no unmodeled perturbation. However, this would not be the case in real conditions, because of these two sources of error.

Then, concerning the free-flying phase, note that for the sake of the implementation we also use the flatness to derive expressions for the initial condition (angle θ_1 in particular). For this test we set the initial position to $[0, 0]$ and the final position to $[3, 2]$ (m). The derivatives and angles are set to zero at the beginning, while a final velocity of $[-1, 1]$ m/s is imposed in order to get a trajectory that excites the dynamics. Figure 1.5 illustrates the results of this simulation. We can see that the tracking is perfectly done: once again the controller was able to cancel the tracking error down to the numerical precision of the solver, which validates the choice of the control law.

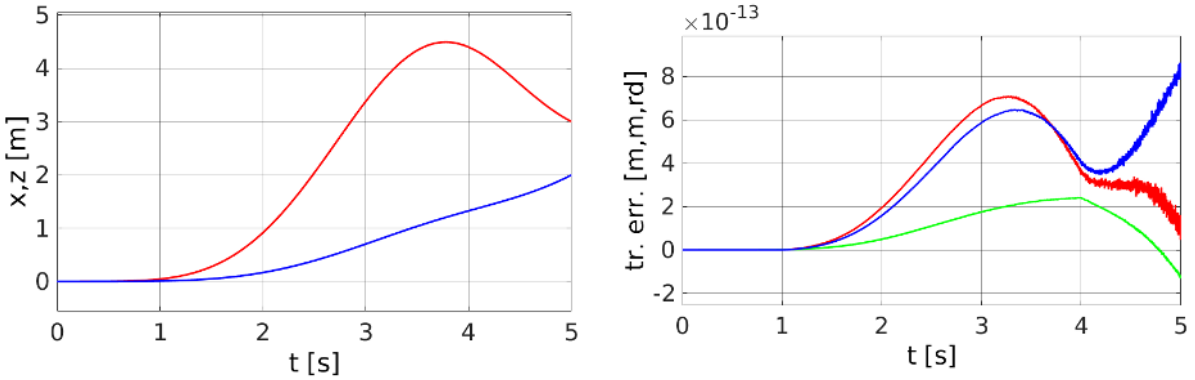

Figure 1.5 – Simulation of the free-flying dynamics of the MonkeyRotor. On the left, the realized position x, z. On the right, the tracking error (in position and angle): the error is of the order of numerical precision, which shows that the controller perfectly tracks the trajectory in these ideal conditions once again.

1.5 Conclusion

In this chapter we have introduced the concept of aerial physical locomotion by considering the MonkeyRotor system — a quadrotor UAV equipped with a 1-DOF arm able to hook at some pivot points and to exploit these contacts for enhancing its maneuvering possibilities. To this end, a suitable dynamical model for both the hooked and free-flying phases has been presented. The specificities of the two corresponding dynamics are mainly related to their degree of actuation: the system is overactuated when in contact, while underactuated when not. As a consequence, the maneuverability of the system varies along with its state, *i.e.*, it is more maneuverable when in contact.

Thus, one can expect that a proper exploitation of the whole dynamics should leverage this particularity: the hooked phase should be subject to ‘informative‘ maneuvers.

We also introduced a collision model for the re-hooking event, which we think is of paramount importance for further exploration of the aerial locomotion concept. This model is based on global energy dissipation, which implies that it does not require any physical parameter. Two control laws for the two hooked and free-flying phases have also been proposed and tested in simulation. In ideal conditions, *i.e.*, parameters perfectly known and no perturbations/unmodeled phenomenon (also no input saturation), the two controllers are able to track a desired trajectory that is submitted to them without any error.

CHAPTER 2

TRAJECTORY PLANNING FOR THE MONKEYROTOR

2.1 Introduction

This chapter is dedicated to the presentation of the trajectory planning algorithm that has been developed and tested in simulation specially for the MonkeyRotor. Still considering the case-study of the contact-fly-contact problem, the sought planner aims at building a trajectory that brings the robot from an initial hooked configuration under a first branch, to a second hooked configuration under another branch. Therefore, this planner is constructed in a way that includes the models of the two dynamics of the system (hooked and free-flying) that were described before, but also the impact that occurs at the rehooking.

2.2 Planning algorithm

As explained above, we focus in this chapter on the objective of bringing the MonkeyRotor from the initial rest configuration under the first branch, to the final rest configuration under the second branch, while minimizing some cost. We formally describe this problem in this section.

To do so, we discuss here a trajectory planning strategy meant to generate feasible trajectories for letting the MonkeyRotor passing from a hooked configuration to another hooked configuration. Figure 2.1 depicts the considered scenario: let $\mathbf{x} = [\mathbf{q}^T \ \dot{\mathbf{q}}^T]^T \in \mathbb{R}^{n_x}$, $n_x = 8$, represent the MonkeyRotor state, and assume two initial and final states $\mathbf{x_0}, \mathbf{x_f}$ are given corresponding to the MonkeyRotor hovering stationary while hooked to the initial and final pivot point. represent the actuation constraints on the MonkeyRotor input $\mathbf{u}_f = \mathbf{K}\mathbf{u}$ (see (1.1)). The goal is to find an optimal (w.r.t. a cost of interest) and

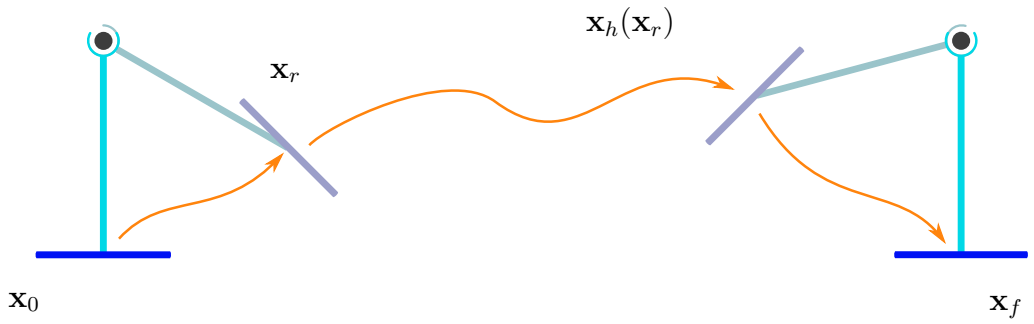

Figure 2.1 – Optimization scheme, where $\mathbf{x}_0$ is the initial state, $\mathbf{x}_r$ the transition state where the system passes from its hooked dynamics to its free flying one, $\mathbf{x}_h$ the reciprocal one and $\mathbf{x}_f$ the final state.

feasible trajectory for the pair $(\mathbf{x}(t), \mathbf{u}(t))$ over a time interval $t \in [t_0, t_f]$ able to bring the MonkeyRotor from $\mathbf{x}(t_0) = \mathbf{x_0}$ to $\mathbf{x}(t_f) = \mathbf{x_f}$ while coping with the actuation constraints 1.17. Depending on the conditions (initial/final states, actuation constraints), one can expect the optimal trajectory to involve an initial 'swinging' (attached to the first pivot point) until the hook is released (state $\mathbf{x}_r$ in Fig. 2.1), followed by a free-flying phase, and subsequently a possible final 'swinging' when re-hooking with the next pivot point (state $\mathbf{x}_h$ in Fig. 2.1). Indeed these swinging maneuvers can be exploited for efficiently building up/losing energy, thus fully exploiting the possibility to actively exchange forces with the environment (as in a locomotion task) in addition to the available thrust/torque inputs.

The complexity of this optimization problem, also due to the change in the MonkeyRotor dynamics when switching from a hooked phase to a free-flying phase, does not allow for an analytical solution (i.e., finding the complete optimal trajectory over $t \in [t_0, t_f]$). Therefore, a numerical optimization method needs to be employed: among the many possible strategies, we now discuss the adopted one which we found amenable to a numerical resolution despite the fact it is possibly slightly suboptimal as we will see.

2.2.1 Optimization procedure

In order to handle the optimization problem, we split it in two loops: the inner loop looks for an optimal trajectory *given* a candidate release state $\mathbf{x}_r$. The outer loop then

tries to optimize the candidate $\mathbf{x}_r$. This method is inspired from the concept of dynamical programming where the global optimum is built from solutions of smaller problems, see [13]. However here, the cost function may differ between the considered subproblems as we will see next, and thus the procedure may not be globally 'optimal' w.r.t. a single objective.

Inner loop

Given a candidate release state $\mathbf{x}_r$ and a cost function $J_1(\mathbf{x})$ (to be specified later on), this first optimization problem

$$\begin{aligned} J_1^*(\mathbf{x}_r) = \min_{\mathbf{u}(t),\, t\in[t_0,\, t_r]} \quad & J_1(\mathbf{x}) \\ \text{subject to} \quad & \dot{\mathbf{x}} = \mathbf{f_h}(\mathbf{x}) + \mathbf{G_h}(\mathbf{x})\mathbf{u} \\ & \mathbf{x}(t_0) = \mathbf{x}_0 \\ & \mathbf{x}(t_r) = \mathbf{x}_r \\ & \underline{\mathbf{u}}_f \leq \mathbf{K}\mathbf{u} \leq \bar{\mathbf{u}}_f \end{aligned}$$

returns the optimal trajectory w.r.t. the cost $J_1(\mathbf{x})$ for joining $\mathbf{x}(t_0) = \mathbf{x}_0$ with $\mathbf{x}(t_r) = \mathbf{x}_r$ at some release time $t_r > t_0$ to be determined by the optimization algorithm. Here, $\dot{\mathbf{x}} = \mathbf{f_h}(\mathbf{x}) + \mathbf{G_h}(\mathbf{x})\mathbf{u}$ is a shorthand for the MonkeyRotor constrained dynamics (1.3)–(1.6–1.7). Note also that the optimal cost $J_1^*(\mathbf{x}_r)$ is a function of the release state $\mathbf{x}_r$.

Subsequently, this second optimization problem

$$\begin{aligned} J_2^*(\mathbf{x}_r) = \min_{\mathbf{u}(t),\, t\in[t_r,\, t_h]} \quad & J_2(\mathbf{x}) \\ \text{subject to} \quad & \dot{\mathbf{x}} = \mathbf{f_f}(\mathbf{x}) + \mathbf{G_f}(\mathbf{x})\mathbf{u} \\ & \mathbf{x}(t_r) = \mathbf{x}_r \\ & \mathbf{p}_E(t_h) = \mathbf{p}_E^* \\ & \|\dot{\mathbf{p}}_E(t_h)\| \leq \mathbf{v}_{max} \\ & \underline{\mathbf{u}}_f \leq \mathbf{K}\mathbf{u} \leq \bar{\mathbf{u}}_f \end{aligned}$$

finds an optimal trajectory for bringing the (now free-flying) MonkeyRotor from $\mathbf{x}(t_r) = \mathbf{x}_r$ to a hooked state with the second pivot point represented by the hook constraint $\mathbf{p}_E(t_h) = \mathbf{p}_E^*$, where $t_h > t_r$ (the hooking time) is to be determined by the optimization. Here, similarly to before, the notation $\dot{\mathbf{x}} = \mathbf{f_f}(\mathbf{x}) + \mathbf{G_f}(\mathbf{x})\mathbf{u}$ is a shorthand for the

free-flying MonkeyRotor dynamics (1.8).

Note that the expected constraint $\dot{\mathbf{p}}_E(t_h) = \mathbf{0}$ (null end-effector velocity when hooking) is here replaced by the milder $\|\dot{\mathbf{p}}_E(t_h)\| \leq v_{max}$, with $v_{max} > 0$ being a small positive threshold. Indeed, we empirically found that accepting a nonzero but small $\|\dot{\mathbf{p}}_E(t_h)\|$ facilitates the optimization procedure since the optimal trajectory is allowed to 'exploit' a hard but controlled impact with the pivot for quickly reducing the system energy without spending control effort, in a way, again, reminiscent of how humans/animals exploit contact when moving. We note that the effects of a possible nonzero $\|\dot{\mathbf{p}}_E(t_h)\|$ are taken into account by the impact modeling discussed in Sect. 1.2.4). Finally, note that the optimal cost $J_2^*(\mathbf{x}_r)$ and the whole optimal state evolution $\mathbf{x}^*(t)$, $t \in [t_r,\, t_h]$, are again a function of the release state $\mathbf{x}_r$. We will then denote with $\mathbf{x}_h(t_h;\ \mathbf{x}_r)$ the final hook state reached at t_h as a function of the release state $\mathbf{x}_r$.

Finally, this third optimization problem

$$
\begin{aligned}
J_3^*(\mathbf{x}_r) = \min_{\mathbf{u}(t),\, t\in[t_h,\, t_f]} \quad & J_3(\mathbf{x}) \\
\text{subject to} \quad & \dot{\mathbf{x}} = \mathbf{f_h}(\mathbf{x}) + \mathbf{G_h}(\mathbf{x})\mathbf{u} \\
& \mathbf{x}(t_h) = \mathbf{\Gamma}(\mathbf{x}_h(t_h^-;\ \mathbf{x}_r)) \\
& \mathbf{x}(t_f) = \mathbf{x}_f \\
& \underline{\mathbf{u}}_f \leq \mathbf{K}\mathbf{u} \leq \bar{\mathbf{u}}_f
\end{aligned}
$$

finds an optimal trajectory for bringing the MonkeyRotor which is now hooked from $\mathbf{x}(t_h)$ to the final state $\mathbf{x}(t_f) = \mathbf{x}_f$, where $t_f > t_h$ is to be determined by the optimization algorithm. Here $\mathbf{\Gamma}(\mathbf{x}_h(t_h^-;\ \mathbf{x}_r))$ is a shorthand for the reset action performed by the collision model of Sect. 1.2.4 because of the possibly nonzero hooking velocity $\dot{\mathbf{p}}_E(t_h)$. Finally, the optimal cost $J_3^*(\mathbf{x}_r)$ is, again, a function of the release state $\mathbf{x}_r$.

These three optimization problems are solved by exploiting the direct transcription method, in particular the Matlab implementation of the Drake libraries [63], on second order spline trajectories for $\mathbf{x}$ and $\mathbf{u}$. Other possible approaches could include the use of the flatness property for the MonkeyRotor, in order to avoid numerical integration of the system dynamics [50, 40], or a direct collocation method [9]. We found out that the flatness approach is not very well suited to this case because the expressions of the input constraints are too complex, especially for the free-flying dynamics. Likewise, the direct collocation method, though it seems computationally interesting by construction, did not give a significant upturn in the solving speed, hence the choice of the direct

transcription method.

Outer loop

The outer loop, as opposed to the inner one, attempts to determine the optimal release state $\mathbf{x}_r^*$ by solving the following minimization problem

$$\mathbf{x}_r^* = \arg\min_{\mathbf{x}_r}(J_1^*(\mathbf{x}_r) + J_2^*(\mathbf{x}_r) + J_3^*(\mathbf{x}_r)).$$

In this case, we opted for a simple grid search (*i.e.*, brute-force) algorithm for finding the optimal $\mathbf{x}_r^*$. Indeed $\mathbf{x}_r$ can be parameterized by the pair $(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ (four variables) since it must be compatible with the hook constraints (1.6–1.7), thus considerably reducing the search space.

2.2.2 Cost function

Reasonable choices for the cost functions $J_1(\mathbf{x})$, $J_2(\mathbf{x})$ and $J_3(\mathbf{x})$ could be the execution time or control effort or energy for generating minimum-time or minimum-effort/energy trajectories from $\mathbf{x}_0$ to $\mathbf{x}_f$. Based on the observation that the precision of the trajectory tracking is highly dependent on the quality of the parameter estimation, here we however choose to also consider optimality of the *state sensitivity* w.r.t. variations in the system parameters, e.g., mass, inertia, CoM location, propeller characteristics, and so on. Indeed, one can expect some unavoidable level of uncertainty in the various parameters of the MonkeyRotor dynamical and actuation model. Starting from the idea of [5], we thus aim at generating an optimal trajectory that ensures minimal effect of the parametric uncertainty onto the tracking performance, *i.e.*, generate an optimal state trajectory $\mathbf{x}^*(t)$ which ends up to be most insensitive to parametric variations by construction. Therefore, as a first step to this approach, we implement a minimization of some norm of the open-loop state sensitivity. The tracking of such optimized trajectory will be facilitated when a parameter is poorly known as explained in [5].

Note that this open-loop state sensitivity does not capture the effect of the controller on the dynamics when tracking a trajectory with parameters that are not perfectly calibrated. A detailed study of this problem and a more complete framework is developed in Part II, which is able to take into account the control laws at the planning stage.

We here recap, for the reader convenience, the essential machinery for computing the sought open-loop state sensitivity. Let then $\mathbf{p} \in \mathbb{R}^p$ be a vector of parameters of interest which, in our case, is taken as $\mathbf{p} = [J_B \; J_1 \; m_B \; m_1 \; L_B \; d_1] \in \mathbb{R}^{n_p}$, $n_p = 6$, and define

$$\mathbf{\Pi}(t) = \frac{\partial \mathbf{x}(t)}{\partial \mathbf{p}} \in \mathbb{R}^{n_x \times n_p} \tag{2.1}$$

as the state sensitivity matrix w.r.t. the parameters $\mathbf{p}$. Although $\mathbf{\Pi}$ does not admit, in general, a closed-form expression, one can find an expression for its dynamics as

$$\dot{\mathbf{\Pi}}(t) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\mathbf{\Pi}(t) + \frac{\partial \mathbf{f}}{\partial \mathbf{p}}, \qquad \mathbf{\Pi}(t_0) = \mathbf{0}, \tag{2.2}$$

where $\mathbf{f}$ in our context stands for the hooked or free-flying dynamics depending on the particular phase. It is then possible to numerically integrate (2.2) over the interval $[t_0, \, t_f]$ for obtaining the behavior of $\mathbf{\Pi}(t)$.

By exploiting the availability of $\mathbf{\Pi}$, we then choose, among other possibilities, to minimize a weighted sum of the total execution time $t_f - t_0$ *and* of a norm of the state sensitivity at the hook time $\mathbf{\Pi}(t_h)$, with the aim of generating near minimum-time trajectories that are also most insensitive to uncertainties in the MonkeyRotor parameters when approaching the second hook. This is formally obtained by letting

$$J_1 = t_r - t_0, \quad J_2 = t_h - t_r + \gamma \|\mathbf{\Pi}(t_h)\|_{\mathbf{w}}, \quad J_3 = t_f - t_h. \tag{2.3}$$

The gain $\gamma > 0$ is meant to tune the relative weight between the two optimization objectives which we combine here, and the matrix norm is defined as

$$\|\mathbf{\Pi}\|_{\mathbf{w}}^2 = \sum_{i,j} w_{ij} \Pi_{ij}^2 \tag{2.4}$$

for a set of non-negative weights $\mathbf{w} = [\dots w_{ij} \dots]$ whose purpose is to select (and give relative importance to) the desired entries in matrix $\mathbf{\Pi}$.

2.3 Results

In this section we present a number of simulation results meant to validate the proposed modeling, planning and control strategies for the MonkeyRotor. The first subsection is dedicated to the results of the trajectory planning algorithm, and the second

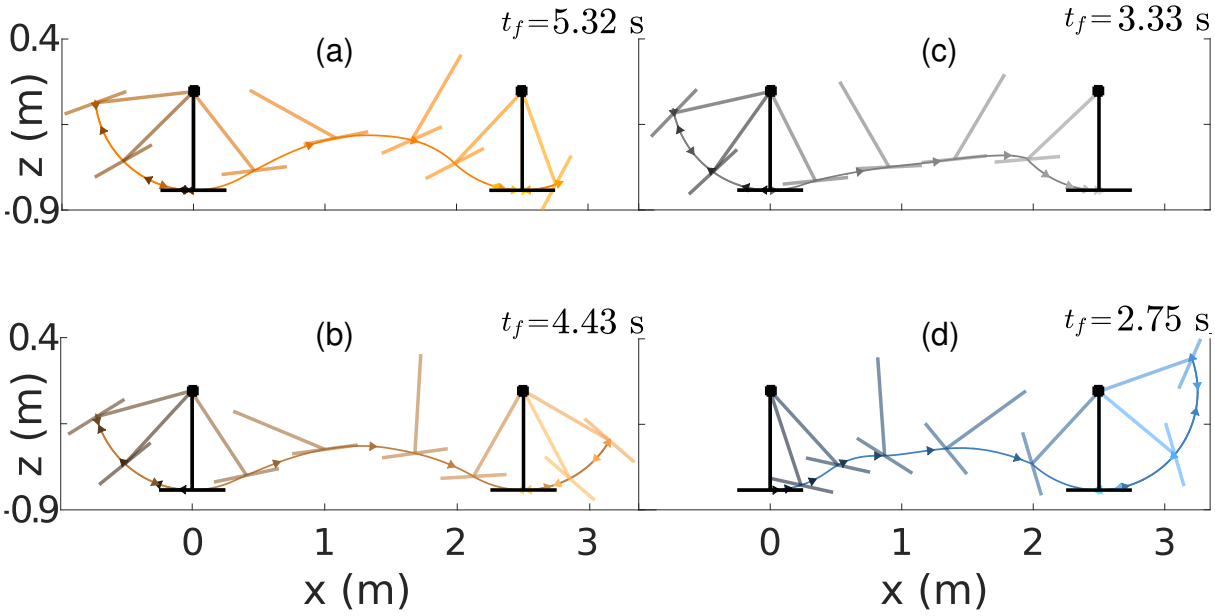

Figure 2.2 – Time-optimal trajectories of the MonkeyRotor CoM $\mathbf{p}_B(t)$ for the cases of a total thrust/weight ratio of (a) 60%, (b) 70%, (c) 90% and (d) 150%. Note how, depending on the case, a swinging maneuver is produced for either building up energy before flight and/or for quickly losing energy after flight.

one to the control tracking performance when also considering parameter uncertainty. A video was made that illustrates some of the tested cases [1].

2.3.1 Trajectory planning

We implemented the trajectory planning framework described in Sect. 2.2 with the values reported in Table 2.1. We first report the results of only minimizing w.r.t. the execution time by setting $\gamma = 0$ in (2.3)). We then consider the concurrent minimization of the state sensitivity norm by setting $\gamma = 1$.

Minimization w.r.t. execution time

In order to better appreciate the effects of the actuation constraints on the trajectory generation, we considered a total thrust limited to 60%, 70%, 90% and 150% of the total weight while keeping the same constraints on the other inputs for testing the

1. video at https://proxy.ens-rennes.fr/owncloud/index.php/s/Nrx5Rmm9S93TsmF

Hook-hook distance (m)	d_1 (m)	L_1 (m)	L_B (m)	m_B (kg)	m_1 (kg)	J_B (kg m^2)	J_1 (kg m^2)
2.5	0.3	0.75	0.5	1.3	0.2	0.33	0.027

Table 2.1 – Values of the parameters used for the MonkeyRotor model.

States and parameters	$(x,\, z)$ vs. $(m_B,\, m_1)$	$\mathbf{p}_E$ vs. $\mathbf{p}$	$\mathbf{q}$ vs. $\mathbf{p}$	$\mathbf{q}$ vs. d_1
Only time-optimal	0.1902	0.1615	0.0376	0.2128
Time- and sensitivity-optimal	0.0827	0.1083	0.0163	0.1157

Table 2.2 – Comparison of the norm of the state difference at t_h between a time-optimal trajectory, and a time- and sensitivity-optimal trajectory. As expected, when perturbing the parameters, the perturbed state deviates less from the nominal state when executing a time- and sensitivity-optimal trajectory.

MonkeyRotor behavior in different regimes from low to high control effort modes, and ultimately assessing how the environment interaction can be exploited for performing the desired maneuver. The resulting trajectories are reported in Fig. 2.2. In all cases, the MonkeyRotor starts at rest while hooked at the left pivot, and ends at rest hooked at the right pivot. One can note how, in the 'low' control effort modes (cases (a)–(c) with thrust less than weight), an initial swing allows for building up the energy needed for reaching the second hook. In particular, in cases (a) and (b) the low thrust/weight ratio causes the trajectory to look approximately ballistic during the free-flying part. On the other hand, when more thrust is available (cases (c) and even more (d)), the free-flying phase is much more "direct": however, the breaking phase at the second hook is nevertheless performed by exploiting hook constraint, either thanks to the allowed collision with the second hook in case (c), or by performing a final swing in case (d) where the thrust exceeds the total weight.

We believe that the representative cases reported in Fig. 2.2 constitute a good validation of the MonkeyRotor concept, in particular of its switching dynamics which is cleverly leveraged by the trajectory optimization algorithm.

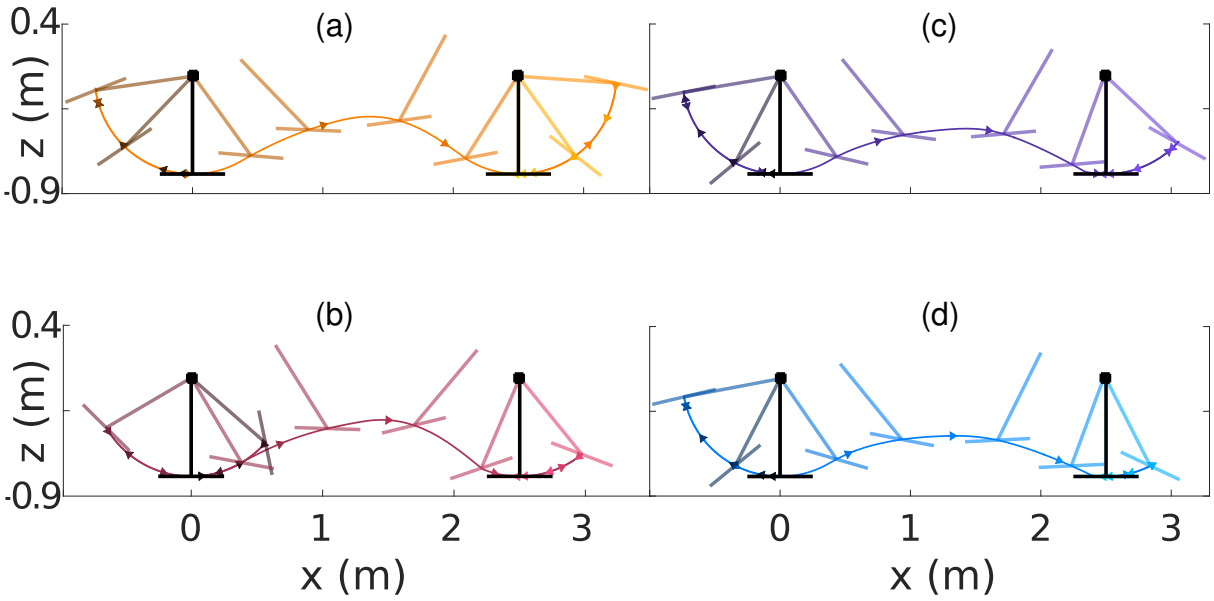

Figure 2.3 – MonkeyRotor trajectories when minimizing for time *and* state sensitivity at t_h. The considered sensitivities are (a) x and z w.r.t. m_B and m_1, (b) $\mathbf{p}_E$ w.r.t. $\mathbf{p}$, (c) q w.r.t. $\mathbf{p}$, (d) q w.r.t. d_1. These sensitivity minimized trajectories are less direct and therefore slower as compared to the time-optimal cases of Fig. 2.2, but they are also less sensitive to variations in the considered parameters.

Minimization w.r.t. execution time and state sensitivity

Focusing on the state sensitivity minimization, several trajectories have been generated while considering the sensitivity of different sets of states and parameters by suitably activating/deactivating the coefficients Π_{ij} via the weighting matrix $\mathbf{w}$ in (2.4). The thrust limit was always fixed at 70% of the total weight, as in the case of Fig. 2.2(b). The resulting trajectories (and combinations of states/parameters) are reported in the four case studies of Fig. 2.3. We can notice that the trajectories, although close in shape, present some variations in their characteristics. In particular, the shape of the flying phase is closer to a ballistic parabola in the cases (a) and (b) while a bit flattened in the other cases. Furthermore, the re-hooking state $\mathbf{x}_h$ systematically comes later (i.e., θ_{1B} is closer to 0) than in the corresponding time-optimal trajectory of Fig. 2.2.

In order to verify the effectiveness of having also optimized w.r.t. the state sensitivity, we performed the following test: we simulated the evolution of the MonkeyRotor states when applying the optimal (open-loop) input $\mathbf{u}^*(t)$ in the nominal non-perturbed case

and in the perturbed case (by increasing each considered parameter by 10%). We then evaluated the difference in the selected states at t_h when executing a trajectory only optimized w.r.t. time and when executing a trajectory *also* optimized w.r.t. the state sensitivity at t_h. Table 2.2 reports the results: one can note how the norm of the difference between nominal and perturbed states at t_h is always lower in the case of a trajectory also optimized w.r.t. the state sensitivity, thus indicating that open-loop execution of this trajectory results intrinsically more robust w.r.t. parametric variations, as expected.

Behavior of the optimization

Fig. 2.4 highlights the role of the outer loop of the optimization. It shows how the duration of a trajectory varies when changing the release state $\mathbf{x}_r$ characterized by the four coordinates θ_B, θ_{1B}, $\dot{\theta}_B$ and $\dot{\theta}_{1B}$.

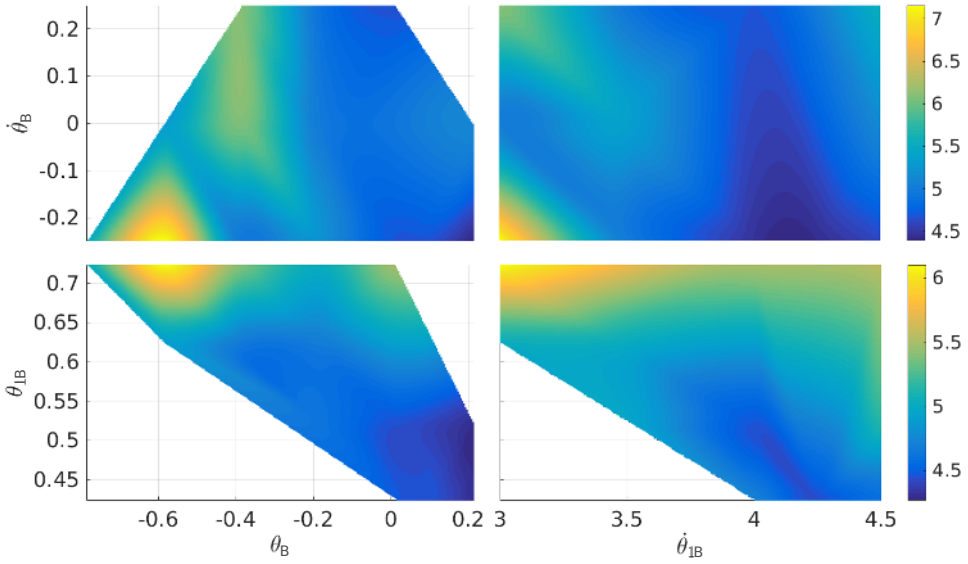

Figure 2.4 – Colormaps of the minimum trajectory duration in seconds, reachable with two fixed states at the release instant.

Interestingly, we observe that the resulting costs seem to feature some smoothness in the explored space. In particular, we see that the release velocity $\dot{\theta}_{1B}$ features a maximum zone around a certain value ($4.2\ m/s$ here). Likewise, we see from the plots that the release angle needs to remain small while not cancelled. This is explained by the fact that the free-flying phase benefits from a nonzero initial vertical velocity which ballistically pre-compensates the effect of gravity.

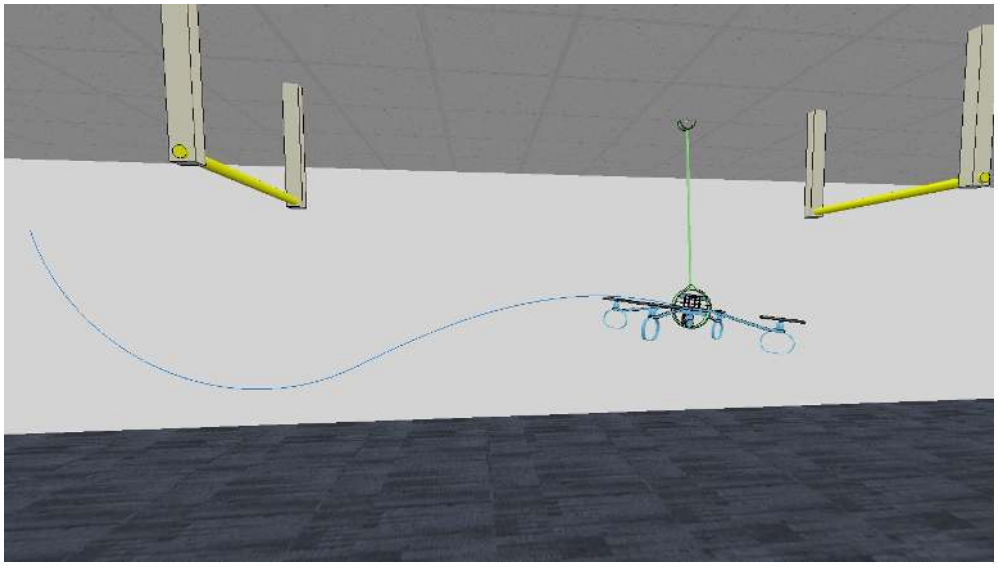

Figure 2.5 – V-REP visualization of the MonkeyRotor.

2.3.2 Trajectory tracking

We now illustrate the performance of the control laws described in Sect. 1.3 in tracking the reference near-optimal trajectories generated by the planning algorithm. To this end, we implemented the MonkeyRotor switching dynamics and the control laws in Simulink, and employed, for the sake of visualization, the V-REP [2] simulation environment as shown in Fig. 2.5. This software, V-REP, provides a specific client-server interface allowing to control it by means of the provided "Remote API" library. We integrated the corresponding Remote API functions within C S-Functions in Simulink, which makes possible to set the pose of any virtual object or joint in the virtual environment. The resulting 3-D visualization is illustrated on Fig. 2.5.

As a representative case study, Fig. 2.6 reports the tracking performance for a time-optimal trajectory obtained for a thrust limit of 70% of the weight, and with random perturbations to the parameters of $\pm 5\%$ of their nominal values. It is worth noting how, despite the parametric variations, the control inputs always remain within their bounds (represented by dashed horizontal lines), and how the norm of the end-effector velocity $\|\dot{\mathbf{p}}_E\|$ falls below the threshold $\mathbf{v}_{max}$ at t_h as planned. The performance in tracking the reference optimal state (dashed lines) is also quite satisfactory.

As an additional validation, we also ran a statistical analysis of the overall tracking

2. http://www.coppeliarobotics.com/

error (averaged over the whole trajectory) and end-effector re-hooking error at t_h on the trajectory of Fig. 2.3-(b) (whose state sensitivity is optimized against all the considered parameters $\mathbf{p}$) when varying some parameters of interest from 90% to 110% of their nominal value. Figure 2.7 reports the results of this analysis: in Figs. 2.7(a–c) we consider the variation of m_B, L_1 and m_1, while Fig. 2.7(d) considers the presence of an external disturbance, a wind gust of varying amplitude with duration $0.2\ s$ and applied during the free-flying case along the negative $\mathbf{x}_W$ axis.

One can note how the performance remains quite satisfactory despite the parameter variations and/or external disturbance especially in terms of the re-hooking error, thus showing that the proposed combination of the state sensitivity minimization (planning stage) and the closed-loop tracking controller control is able to yield a successful MonkeyRotor maneuver also in more realistic conditions. We note that the re-hooking error remains almost constant except when changing the length of the arm, which is expected since the parameter L_1 does not affect the free-flying dynamics and thus is not taken into account in the sensitivity minimization.

2.4 Conclusion

In this chapter we have proposed and tested an optimization framework for generating optimal motion plans for the MonkeyRotor under constrained actuation. The proposed algorithm breaks the problem into three subproblems and assesses then in two stages: an inner loop and an outer loop. The whole concept has been successfully validated in a number of simulations, including the behavior of the trajectory tracking by means of the two control laws derived in the previous chapter.

Several conditions have been considered for the trajectory generation, including a range of thrust limitations and multiple cost functions to be minimized, which allows us to widely study the behavior of the planner and the resulting trajectories for aerial locomotion.

Possible improvements of the proposed framework could include the possibility of executing more complex maneuvers, e.g., jumping to multiple branches in sequence, as well as the use of online replanning strategies for continuously refining the initial optimal trajectory during motion. An application of the trajectory generation framework presented in Part II is also foreseen, which should improve even more the tracking performance by leveraging the knowledge of the control law (as opposed to the open-

loop state sensitivity approach used in this chapter).

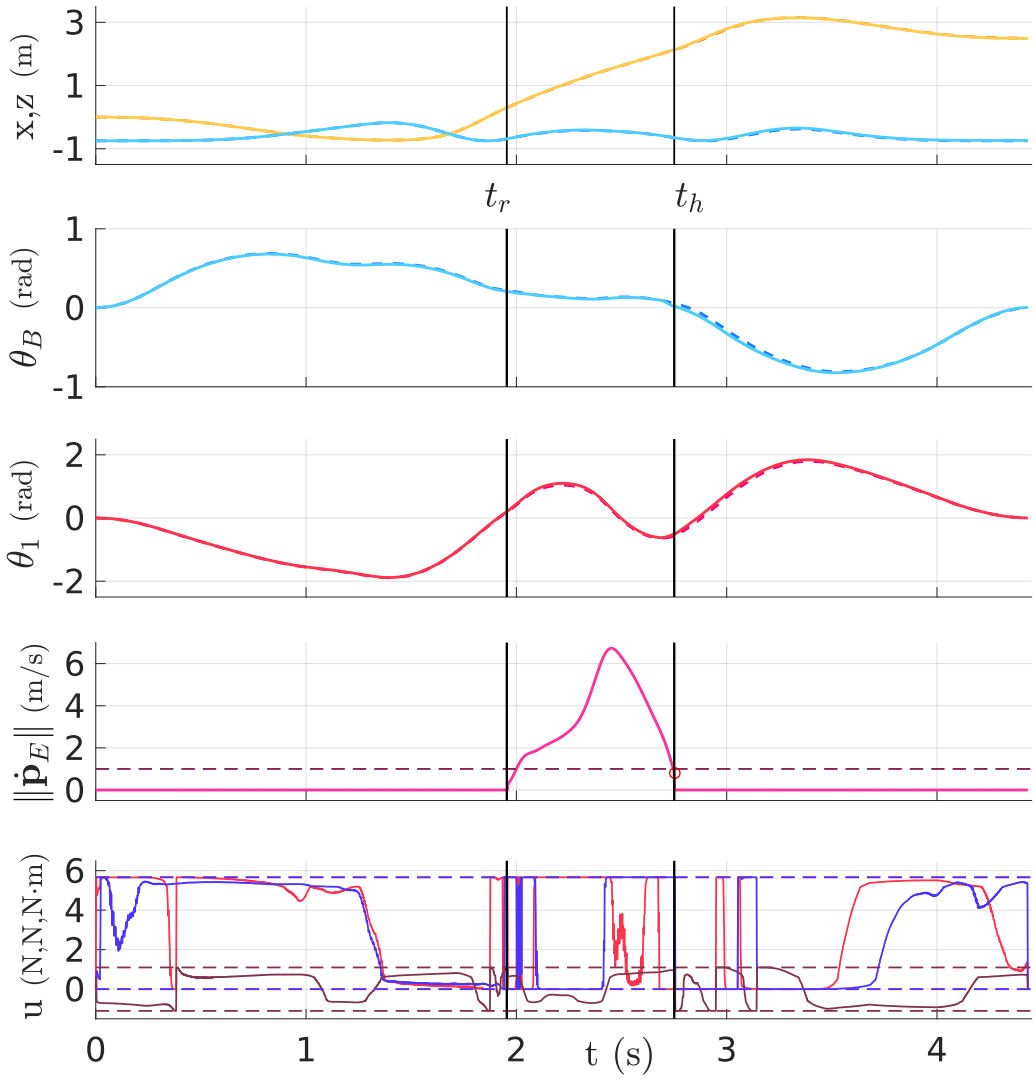

Figure 2.6 – Behavior of the MonkeyRotor states, inputs and end-effector norm velocity while tracking an optimal trajectory.

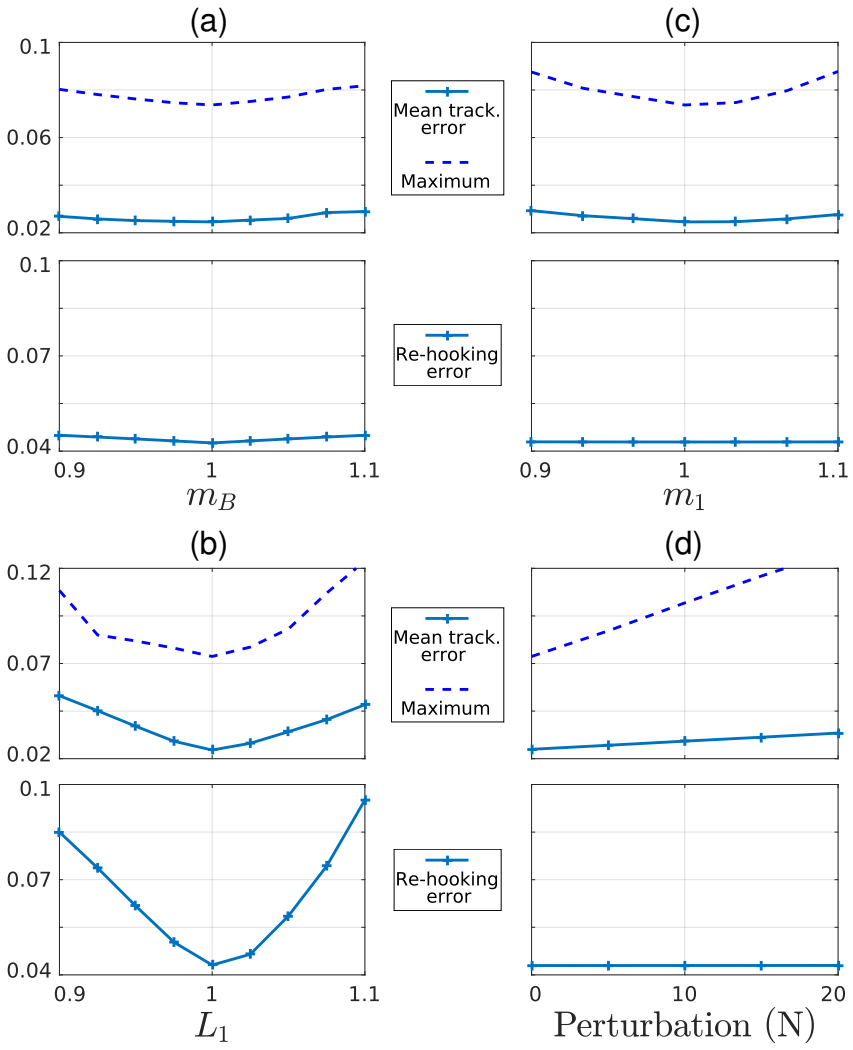

Figure 2.7 – Performance of the proposed planning/control framework under parametric variations (Figs. (a–c)) and external disturbances (Fig. (d)): in each figure the top plot reports the mean (solid line) and max (dashed line) values of the tracking error norm during the trajectory execution, and the bottom plot reports the value of the re-hooking error $\|\mathbf{p}_E(t_h) - \mathbf{p}_E^*\|$. The parameters m_B, m_1 and L_1 are varied from 90% to 110% of their nominal value, while the external perturbation (wind gust) has amplitude ranging from 0 N to 20 N.

CHAPTER 3

CONCEPTION OF THE MONKEYROTOR PROTOTYPE

3.1 Introduction

This chapter is dedicated to the description of the MonkeyRotor prototype that has been designed and realized in order to test the previously derived theoretical elements in real conditions. The content is divided in two main parts that each covers one of the main design themes that have been involved in the process, namely the design of the hooking system, and the implementation of a real robot with details on its mechanical structure and the software architecture.

3.2 Design of the hooking system

3.2.1 General concept

In the following, we call the part of the environment to which the MonkeyRotor hooks itself a *branch*.

The first thing to notice about the joint between the MonkeyRotor and the branch is that it is a multifunctional subsystem. In fact, it should implement the following requirements:

1. the rotating joint with the environment when the system is hooked;
2. the controllable hooking, *i.e.*, arm of the MonkeyRotor attached to the branch or not.

Note that the aim for the scope of this chapter is to build a prototype in order to test and showcase the MonnkeyRotor special dynamics, which implies that we allow

ourselves to design a special branch system that includes parts of the actuation mechanism. However, of course, it is clear that in a real application the branch would be supposed to be some passive part of the environment, which is not necessarily under control and thus may not provide such ideal rotating joint and hooking system as here. This is considered to be out of the testing scope we have here and thus, we limit ourselves to the simpler case where the branch is completely under control at the design stage.

Several classes of solutions have been imagined to implement these two functionalities. One first intuitive idea consists of a mechanical gripper with two or more fingers able to grasp the branch. Starting from there, two possibilities arise: either the rotating joint is part of the arm — below the gripper —, or it is part of the branch. In the case it is part of the arm, it means that during the flying phase it has to be somehow oriented towards the branch for the hooking event to happen properly. This could be achieved either with a passive spring system, or with the assistance of a dedicated actuator. In both cases the dynamics is affected, and the onboard weight is increased.

Another solution consists of a pneumatic system. This system would rely on a suction cup able to grip a planar surface by means of a commanded depression of the air contained in the cup. Several electrical pumping systems exist that are able to fit this need. The main interest of this class of solutions lies in the small complexity of the mechanical parts. The same two options as for the mechanical gripper remain concerning the placement of the rotating joint, *i.e.*, either we place it above of below the end-effector. As discussed before, the case where the joint is under the attachment system requires the use of a heading actuation — passive or active. On the other hand, a pneumatic gripper would require some flat surface in order to work properly, which means that an actuated heading of the target flat surface must be set up.

A third idea is to leverage magnetism to tackle this attachment problem. As illustration, the use of magnetism in such a system would be the same as in an electrically commanded door: a coil is powered with a controlled electrical current which results in a magnetic field both in the coil and in the separated part to be locked, *i.e.*, the door, and for us the end-effector at the extremity of the arm. The resulting magnetic field lines circulate in a magnetic circuit especially designed for this purpose, which leads to an adhesion force that keeps the end-effector stuck to the coil. Such a solution may seem unreasonable at first glance because of the required metallic parts for the magnetic circuit and the coil, as well as the electrical power to be fed into the coil which

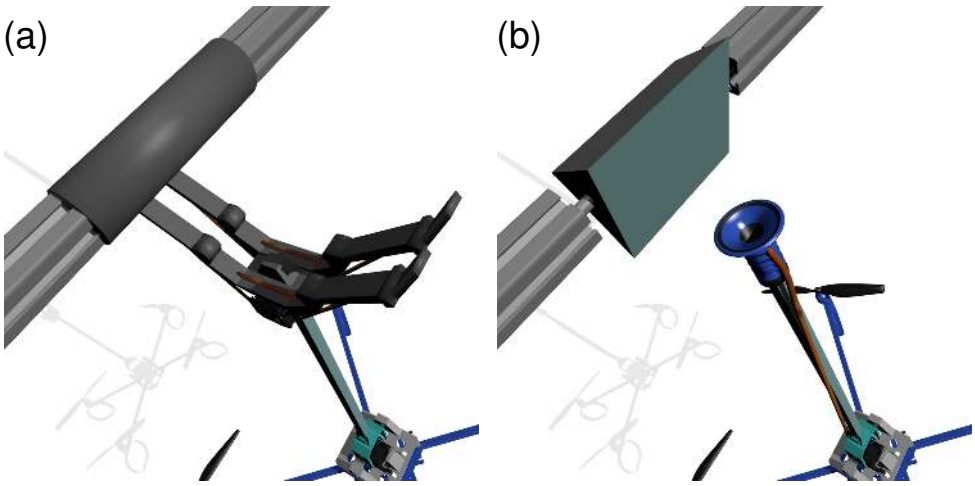

Figure 3.1 – Two of the four proposed solutions for the hooking system. On (a), an onboard mechanical gripping system that surrounds the branch, and on (b), an active suction cup that grips a pivoting planar surface available on the branch.

intuitively seems high. Nevertheless, putting the coil system on the environment side considerably simplifies the design.

Finally, the possibility of a passive hook can be considered. Indeed, choosing some particular rounded shape for the end-effector could allow the system to implement the sought behavior. However, this would require a special control policy in order to achieve a release maneuver, which may be quite restrictive for the dynamics, because of the very specific movements it would require in order to ensure the complete release. This solution features the advantage of simplicity, at the expense of restrictions in the exploitation of the system particular dynamics.

3.2.2 Choice of the solution

These four design ideas are illustrated in figures 3.1 and 3.2, which render some possible implementations. Table 3.1 synthesizes the pros and cons that were previously evoked for each solution.

Though it may sound heavy and inappropriate at first glance, we found out that the magnetic solution actually features significant advantages:

1. it keeps the mechanics simple;

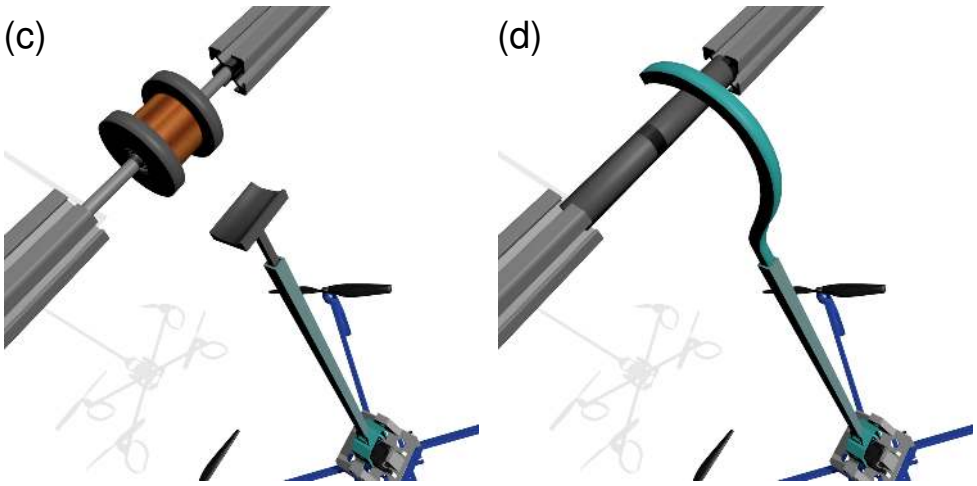

Figure 3.2 – Two of the four proposed solutions for the hooking system. On (c), an electromagnetic system keeping the end-effector stuck to a coil that is part of the branch, and on (d), a passive hooking system that features the pivoting ability by a direct mechanical contact between the end-effector and the branch.

2. the end-effector only consist of a small passive ferromagnetic piece which is light;
3. it is simple to control;
4. some margin is allowed for the re-hooking phase in case of positioning error.

For these reasons, we finally chose to implement the hooking system with a **magnetic** solution. Moreover, the rotating joint functionality is undertaken by the branch instead of the end-effector, which allows a clean, efficient and robust joint design without onboard-related limitations.

The specific design of such a magnetic hooking system will now be described in detail.

3.2.3 Magnetic coil design

Geometry

First of all, we chose to implement the rotating joint on the branch side in accordance with the advantages discussed before.

The concept of the magnetic hooking system basically relies on the ability of magnetism to imply a force on a part of a magnetic circuit towards another, which we seek

	Mechanical complexity	**Onboard load**	**Control complexity**	**Hooking margin**
Gripper	medium	high	medium	medium
Pneumatic	high	high	medium	low
Magnetic	simple	medium	simple	medium
Passive	simple	low	high	low

Table 3.1 – Comparison of the pros and cons of each solution. The mechanical gripper and the pneumatic system both feature a non-neglectible onboard load and a certain control complexity. The magnetic and passive solutions on the other hand are simple and light.

to be strong enough to maintain the MonkeyRotor hooked to the branch by design. To achieve such a magnetic force, the geometry of the magnetic circuit has to meet simple criteria:

1. when entering in hooked state, *i.e.*, when the end-effector is close to the right position and the coil is powered, the field lines should travel the shortest possible distance in the air;
2. when in hooked state, the field lines should not go through air but only stay inside the magnetic circuit.

We found out that a good geometry to answer these requirements is the one depicted on Fig. 3.3. It consists of the coil being oriented coaxially to the rotating joint, with the magnetic circuit being closed by the end-effector disposed in parallel. This choice for the geometry makes the hooking isotropic, *i.e.*, the end-effector can arrive from any direction in the plane without the need for steering the magnetic hooking system towards it, thanks to its cylindrical shape.

Note that this geometry of the coil armature features an axial symmetry, which is adapted to the design of the rotating joint. One drawback of this solution, though, is that the coil rotates w.r.t. to the branch and at the same time needs to be powered through electrical wires. The resulting rotating limitation is overcome by letting a sufficient amount of wire between the moving coil and its grounded power supply. As a consequence, we assume that the total rotation of the branch joint is still limited to a certain range of minimum and maximum angles — which can be purposely set far enough for the prototype operating mode. In case the system would need to achieve multiple turns, a contact system similar to the collector in continuous current electrical

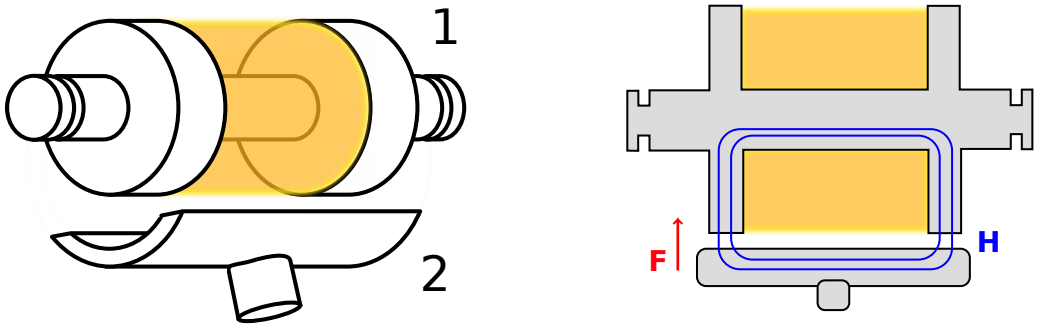

Figure 3.3 – Concept of the magnetic hooking system. The hooking system 1 is linked to the ground via a rotating joint, while the passive ferromagnetic piece 2 is the end-effector at the extremity of the arm of the MonkeyRotor. When subject to an electrical current, the coil induces a magnetic field H in the magnetic circuit — blue lines are the corresponding field lines — that produces an attractive force F onto part 2.

motors is possible.

Magnetic force model when hooked

When feeding the coil with electrical current, a magnetic field occurs in the magnetic circuit which induces an adhesion force onto the end-effector, and thus makes it stuck to the armature of the coil. Note that this is done with a 0% efficiency, because the electrical power injected in the coil is completely dissipated into heat without mechanical output power. Indeed, when the end-effector is hooked there is no relative movement between the arm of the MonkeyRotor and the hooking system, forbidding any mechanical power to be produced. In other words, the system only converts electrical current into the adhesion force, and does not produce output power. As a consequence, the coil will warm up because the electrical power it consumes is converted into Joule losses.

The detailed design of the magnetic circuit geometry needs to take this behavior into account to ensure that 1) the force is maximized given certain geometrical limits, and 2) the coil remains cold enough in continuous operation. Hence, the problem of designing this magnetic hooking system can be expressed as an optimization problem under constraints. Let g be a vector of the geometrical parameters, F_{adh} be the magnitude of the adhesion force that the hooking system is able to provide, T_{coil} the temperature of the coil, and T_{max} the maximum allowed temperature for the coil. With these notations,

the problem can be stated as

$$\begin{aligned} \max_{\mathbf{g}} \quad & F_{adh} \\ \text{s.t.} \quad & T_{coil} < T_{max} \end{aligned} . \tag{3.1}$$

Of course, we need to build a model for the adhesion force in order to assess this design optimization problem. To do so, the classical magnetic models and associated methods can be employed, such as the ones described in [32]. Let r_i be the inner radius and r the outer radius of the coil, l the length of the coil, a the border cylinders width, e the thickness of the end effector, and α the angle it makes with the rotating joint axis. These geometric parameters are depicted on Fig.3.4, which shows how they are arranged together with the global geometry.

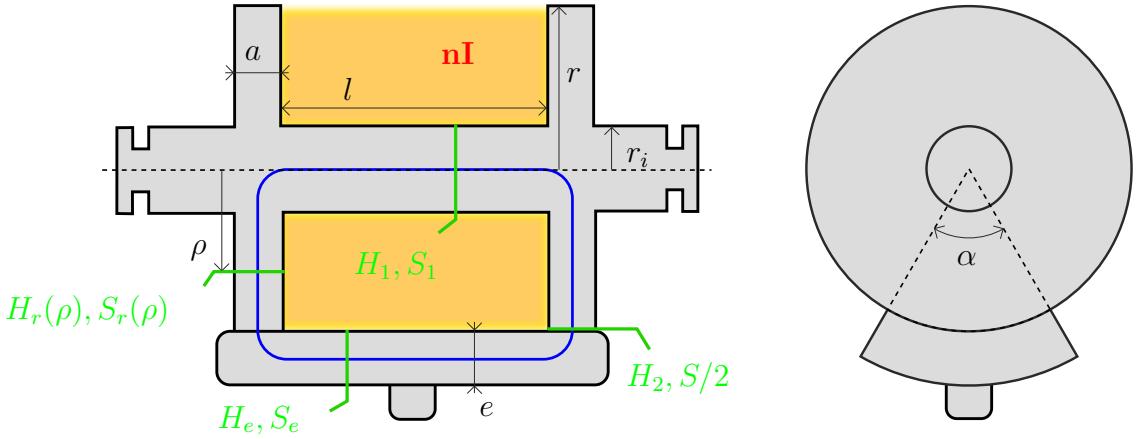

Figure 3.4 – Geometry of the coil with its armature. There are n turns of wire in the coil, each one passed through by a current I.

Let also n be the number of turns of the coil wire, I the electrical current feeding it, S the section of the magnetic circuit at the contact, and H the magnetic field inside the magnetic circuit. This magnetic field inside the magnetic circuit is derived by applying the Ampere law to some closed field line that loops in the whole circuit, as depicted by the blue line in Fig. 3.4,

$$\oint H dl = nI \tag{3.2}$$

where the field is considered aligned with the normal of the crossed section. Let L_c be the total length of the magnetic circuit, *i.e.*, of an average field line. Let also μ_r be the

relative permeability of the magnetic circuit material and μ_0 the one of the vacuum, such that the absolute permeability of the magnetic material is $\mu = \mu_0\mu_r$. In the case where the section of the magnetic circuit is constant, the above relation can be integrated as a whole in order to derive the corresponding induction. In fact, the application of the law results in the magnetic induction approximation

$$B = \mu H = \frac{\mu_0\mu_r nI}{L_c}. \tag{3.3}$$

Indeed, this expression does not hold for the detailed geometry that we consider, because the magnetic field may not be the same in the different parts of the magnetic circuit. However, this relation allows to compute a first approximation of the adhesion force that keeps the end-effector stuck against the armature of the hooking system. In fact, by linking the work of this magnetic force along a small virtual displacement δ with the corresponding magnetic energy density, we can derive an expression for the force. Let S be the area of the contact surface between the end-effector and the armature of the hooking system, on the two cylindrical plates.

$$F\delta = \frac{B^2}{2mu_0}\delta S \tag{3.4}$$

Injecting the known expression of the magnetic induction B derived in eq. (3.3), this rewrites

$$F = \frac{B^2 S}{2\mu_0} = \frac{S}{2\mu_0}\left(\frac{\mu_r nI}{L_c}\right)^2. \tag{3.5}$$

This expression for the magnetic adhesion phenomenon allows to propose two general design rules, *i.e.*,

1. the length L_c of the magnetic circuit must be the *shortest* possible in order to maximize the adhesion force;
2. the contact area S between the end-effector and the hooking system must be the largest possible.

Nevertheless, the steel-based ferromagnetic materials available to produce the hooking system actually saturate. A consequence of such a saturation is that the magnetic induction B is limited at some point by the material itself instead of the I electrical current sent into the coil. In order to chose the geometry correctly, we need to model this saturation at the different stages of the magnetic circuit. In practice the magnetic circuit

has a complex form and its section changes along with the field lines, which involves a notable refinement of the magnetic model.

Given the parametric geometry, we can model in a better way this phenomenon in order to quantify the magnetic induction that is responsible for the adhesion force. Such a model will also provide a control of the induction in the different parts of the circuit in order to avoid the saturation. Let us cut the magnetic circuit into six parts:

1. the middle axis of the coil, of section S_1, of magnetic field H_1, and of length $l_1 = l + a$;
2. the two plates of the armature, of section $S_r(\rho)$, and of magnetic field $H_r(\rho)$ for a certain radius $0 \leq \rho \leq r$;
3. the end-effector, of section S_e, of magnetic field H_e, and of length $l_e = l + a$;
4. the two portions that link the center of the end-effector with the contact surface, of section $S/2$, of magnetic field H_2, and of length $e/2$.

The contact between the end-effector and the armature of the hooking system is split into two areas, each characterized by a magnetic field H_2 and a section $S/2$. For the section $S_r(\rho)$ inside the plates, we consider a linear interpolation between the 2 limit areas S_1 and $S/2$. This is justified by the fact that the field lines always take the shortest path, which is linear here to fill the extremum sections. Note that some detailed phenomenon like local saturation or field leakage may make this assumption false, which would definitely not change the result a lot.

In this more detailed magnetic circuit, the application of the Ampere law becomes

$$\begin{aligned} nI &= \oint H dl \\ &= H_1 l_1 + 2\int_0^r H_r(\rho) d\rho + H_2 e + H_e l_e \end{aligned} \tag{3.6}$$

Let now ϕ be the magnetic flux in the magnetic circuit, which is such that $\phi = \mu H_i S_i$ at any place of section S_i and of magnetic field H_i in the magnetic circuit. One property of this magnetic flux is that it remains constant along the whole circuit. This allows us to rewrite eq. (3.6) by factorizing the constant magnetic flux ϕ

$$nI = \frac{\phi}{\mu}\left(\frac{l+a}{S_1} + 2\int_0^r \frac{d\rho}{S\frac{\rho}{2r} + S_1(1-\frac{\rho}{r})} + \frac{2e}{S} + \frac{l+a}{S_e}\right) \tag{3.7}$$

where the integral term can be computed explicitly

$$2\int_0^r \frac{d\rho}{S\frac{\rho}{2r} + S_1(1-\frac{\rho}{r})} = \frac{4r\ln(\frac{S}{2S_1})}{S-2S_1}. \tag{3.8}$$

Let Λ be the characteristic length of the magnetic circuit, which is such that

$$\begin{aligned}\Lambda^{-1} =& \frac{l+a}{S_1} + \frac{4r\ln(\frac{S}{2S_1})}{S-2S_1} + \frac{2e}{S} + \frac{l+a}{S_e} \\ =& \frac{l+a}{\pi r_i^2} + \frac{2r\ln(\frac{r\alpha a}{\pi r_i^2})}{r\alpha a - \pi r_i^2} + \frac{e}{r\alpha a} + \frac{l+a}{(r+\frac{e}{2})\alpha e}\end{aligned} . \tag{3.9}$$

Consequently, the magnetic flux can be computed with

$$\phi = \mu n I \Lambda \tag{3.10}$$

From this expression of the magnetic flux, one can go back to each part of the magnetic circuit in order to compute the corresponding magnetic inductions and check that they do not reach the saturation. Indeed, we have

$$\begin{cases} B_1 = \dfrac{\phi}{S_1} \\ B_2 = \dfrac{2\phi}{S} \\ B_e = \dfrac{\phi}{S_e} \end{cases} . \tag{3.11}$$

Finally, the force of adhesion can be computed based on the same idea than previously. To be more precise here, the curvature of the surface have to be taken into account. This is done by applying the same process than before on an infinitesimal portion of the surface. It leads to the following expression for the adhesion force,

$$\begin{aligned}F =& \int_{-\frac{\alpha}{2}}^{\frac{\alpha}{2}} \frac{B_2^2}{2\mu_0} dS(\theta) \\ =& \int_{-\frac{\alpha}{2}}^{\frac{\alpha}{2}} \frac{B_2^2}{2\mu_0} 2ra\cos(\theta)d\theta. \\ =& \frac{B_2^2 S}{2\mu_0}\,\mathrm{sinc}\left(\frac{\alpha}{2}\right)\end{aligned} \tag{3.12}$$

Based on the trajectories that where generated in the previous chapter, we estimated that the maximum centrifugal force that the MonkeyRotor should endure should be about 60 N, and thus we aim at an adhesion force of 120 N with a security coefficient of 2.

Note that the magnetic circuit physically saturates at an induction of 1 T, which means that the force is limited by this maximal induction.

Thermal model

As discussed before, there is no mechanical power generated by the system while it is hooked, and thus all the electrical power is converted into heat via Joule effect. This heat makes the temperature of the coil increase until an equilibrium is reached between the source of heat and its dissipation to the environment. Such a temperature increase typically has a slow exponential transient, which we are not really interested in here.

Taking apart this transient, the maximal temperature increase $\Delta T = T_{coil} - T_{env}$ w.r.t. the environment reached at the equilibrium can be modeled with the basic law

$$\Delta T = P R_{th}$$

where P is the heat, and R_{th} is the total thermal resistance of the coil w.r.t. the environment. Furthermore, we know that the heat is the electrical power, such that $P = RI^2$.

In our case the thermal resistance is the consequence of three phenomena, which are the *conduction* through the whole magnetic system, and the *convection* and *radiation* at the exchange surfaces.

These phenomena can be modeled with 3 corresponding thermal resistances R^d, R^v and R^r for each subpart of the system, as depicted on Fig. 3.5. Let R^d_c, R^v_c and R^r_c be the conductive, convective and radiative thermal resistances of the coil. Let also R^d_e, R^v_e and R^r_e be the conductive, convective and radiative thermal resistances of the cylindrical surface of one side plate. Likewise, let R^d_s, R^v_s and R^r_s be the conductive, convective and radiative thermal resistances of the lateral surface of one side plate.

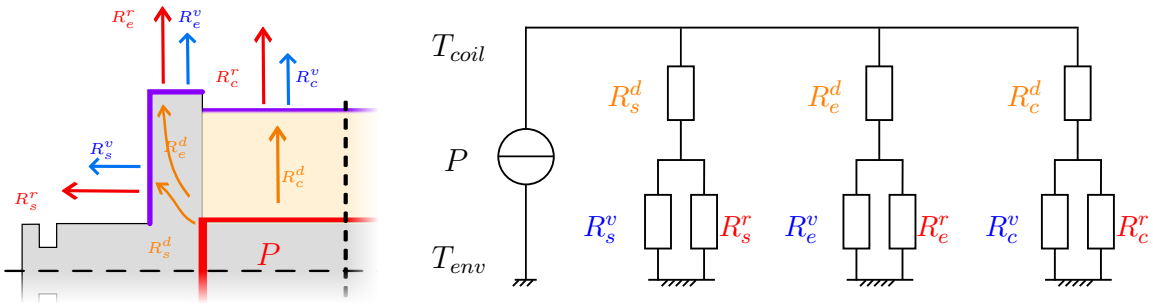

Figure 3.5 – Heat transfers and analog electrical schematic. The orange arrows represent conductive transfers, the blue ones correspond to convection and the red ones represent radiation. The source of heat is considered to be the inner cylinder of the coil highlighted in red. The exchange surfaces that are considered for the model are highlighted in magenta.

We thus get the following expressions:

$$\begin{cases} R_c^d = \dfrac{\ln(r/r_i)}{2\pi\lambda_c l} \\ R_e^d = \dfrac{\ln(r/r_i)}{2\pi\lambda_a a} \\ R_s^d = \dfrac{a}{\lambda_a \pi(r^2 - r_i^2)} \end{cases} \tag{3.13}$$

for the conductive terms, where λ_c and λ_a are the thermal conductivities of the coil and hooking system armature ($Wm^{-1}K^{-1}$). Similarly, we get

$$\begin{cases} R_c^v = \dfrac{1}{2h\pi r l k_\alpha} \\ R_e^v = \dfrac{1}{2h\pi r a k_\alpha} \\ R_s^v = \dfrac{1}{h\pi(r^2 - r_i^2)} \end{cases} \tag{3.14}$$

for the convective terms, where $k_\alpha = 0.7(1 - \frac{\alpha}{2\pi})$ is a coefficient that models the surface reduction that is due to the presence of the end-effector onto the magnetic hooking system, and where h is the convective coefficient of the material ($Wm^{-2}K^{-1}$). Finally,

the radiative terms write

$$\begin{cases} R_c^r = \dfrac{\Delta T}{2\epsilon_c \sigma \pi r l k_\alpha ((T_{env} + \Delta T)^4 - T_{env}^4)} \\ R_e^r = \dfrac{\Delta T}{2\epsilon_a \sigma \pi r a k_\alpha ((T_{env} + \Delta T)^4 - T_{env}^4)} \\ R_s^r = \dfrac{\Delta T}{\epsilon_a \sigma \pi (r^2 - r_i^2)((T_{env} + \Delta T)^4 - T_{env}^4)} \end{cases} \tag{3.15}$$

where $\sigma = 5.67\ Wm^{-2}K^{-4}$ is the Stefan–Boltzmann constant and ϵ_a and ϵ_c are the emissivities of the hooking system armature and coil (considered constant with the temperature here).

Defining the synthetic exchange resistances

$$\begin{cases} R_c^{ex} = \dfrac{1}{\frac{1}{R_c^v} + \frac{1}{R_c^r}} \\ R_e^{ex} = \dfrac{1}{\frac{1}{R_e^v} + \frac{1}{R_e^r}} \\ R_s^{ex} = \dfrac{1}{\frac{1}{R_s^v} + \frac{1}{R_s^r}} \end{cases} \tag{3.16}$$

allows to rewrite the total thermal resistances of the three subparts, and finally the total resistance of the hooking system:

$$\begin{cases} R_c = R_c^d + R_c^{ex} \\ R_e = R_e^d + R_e^{ex} \\ R_s = R_s^d + R_s^{ex} \\ R_{th} = \dfrac{1}{\frac{1}{R_c} + \frac{2}{R_e} + \frac{2}{R_s}} \end{cases} . \tag{3.17}$$

Note that this final expression for the total thermal resistance depends on the temperature of the coil, but at the same time is required to estimate this T_{coil}. Instead of trying to solve analytically the expression of the resistance, which would be complicated perhaps impossible, we chose to implement an iterative method in order to compute its value. The idea of this iterative process is to consider an initial guess for the temperature, use it to estimate the thermal resistance, then refine the temperature estimate and so on. We found out that this method converges with few steps: in practice just

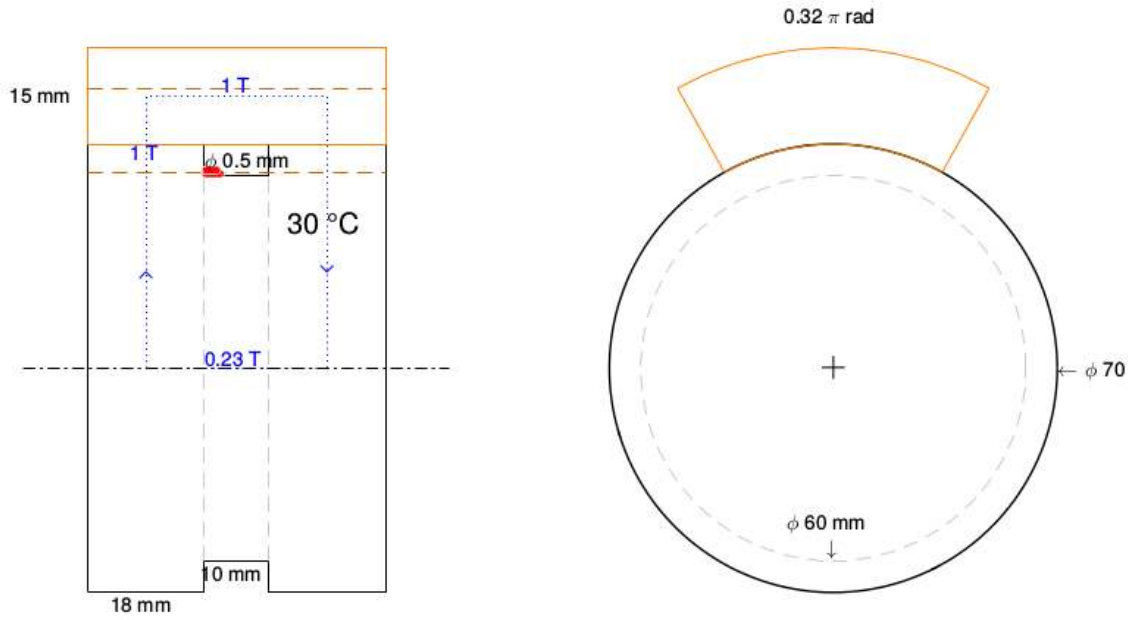

Figure 3.6 – Optimized magnetic system geometry. On the left, a front view where the center dash line is the revolution axis of the hooking system, and on the right, a side view orthogonally to this axis.

three steps led to a precision inferior to 0.1% for the temperature estimate, which we consider sufficient.

Design optimization

Having derived the magnetic and thermal models, one can now implement a solver for the optimization problem (3.1). Using the solver fmincon in Matlab along with an implementation of equations (3.11–3.12,3.13–3.17), we could achieve this optimization and find an optimal geometry.

l [mm]	a [mm]	e [mm]	r [mm]	r_i [mm]	α [rad]	nI [A]	wire length [m]	resistance [Ω]
10	18	15	35	30	0.32π	63.6	33.1	3.04

Table 3.2 – Optimized parameters for the magnetic hooking system.

Table 3.2 synthesizes the obtained optimized geometry parameters. As one can notice, the result may not seem very intuitive at first glance. Indeed, one could have

expected a large space for the coil as well as a long coil so that the magnetic 'strength' would be maximized. However, we see from the equations that the combination of the magnetic saturation with the behavior of the magnetic field lines makes this optimized geometry legitimate. Indeed, the thin coil is sufficient to create a large magnetic induction, which is amplified by the reduction of the section from the coil to the contact area.

Attraction before hooking

When the MonkeyRotor approaches the branch, a magnetic force applies which attracts the end-effector to the coil. Note that the analytical force computation is impossible because the exact path where the magnetic field will pass is not known. However some approximations can be done to build an estimation of this force profile shape.

In this situation, let x be the distance from the end-effector to the coil. When the x is low enough w.r.t. the magnetic circuit length L_c, thus guaranteeing that the induction B is uniform enough, the magnetic force has the expression $F \approx \frac{\mu_0 S}{2}(\frac{nI}{L_c/\mu_r+2x})^2$, which is of the form $\frac{1}{x^2}$ whenever $x >> \frac{L_c}{2\mu_r}$. This condition realized for low distances because the relative permeability μ_r is very high (around 1000 in practice).

Then when x gets higher, the magnetic field leaks around the coil without passing through the end-effector anymore, making the force almost null. This happens when $x \approx l/2$.

In conclusion the force profile is near to an inverse square function smoothly saturated at the beginning, and with a highly decreasing slope from $l/2$. This confirms that the magnetic hooking system is able to provide some margin at the re-hooking, however we note that the inverse square form of the force together with its zeroing at $l/2$ are quite restrictive, and thus the margin remains low in practice.

Active release

Given the hysteresis behavior of the magnetization in the magnetic circuit, releasing the electrical power in the coil will not completely cancel the force of adhesion.

For this reason a soft ferromagnetic material is required, *i.e.*, which has a thin hysteresis cycle width. We thus chose a standard steel for its mechanical and magnetic properties. Note that this material choice, though it helps, is not sufficient to allow a

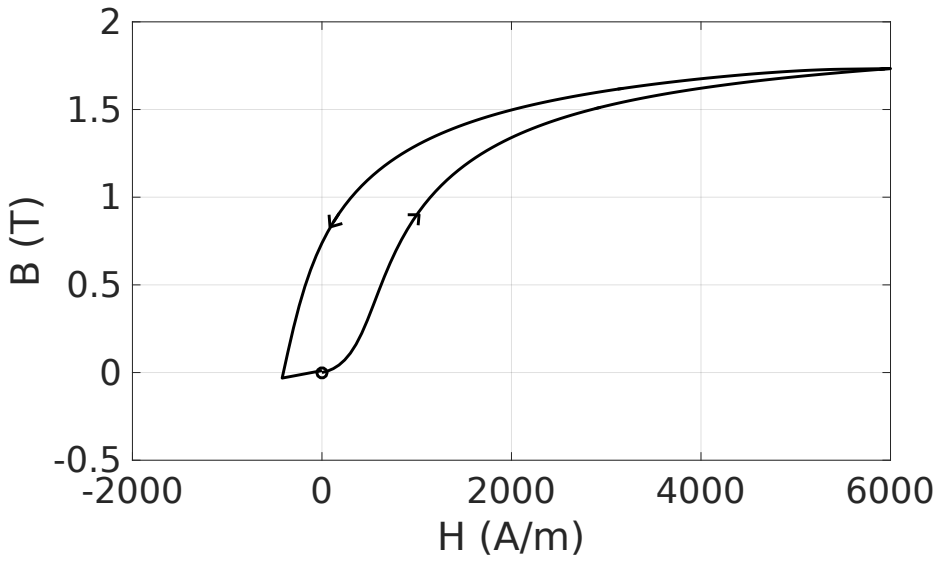

Figure 3.7 – Simulation of the hysteresis behavior of a ferromagnetic material, with a negative impulse at the release to compensate the residual induction.

complete release when stopping to feed the coil with electrical current. In other words, there will always be a remaining force of adhesion because of the magnetic hysteresis.

As a demonstration of this phenomenon and of a possible workaround, we conducted a numerical simulation of this hysteresis behavior by integrating the Jiles–Atherton equations that are able to model it, see [31]. For this simulation we considered a variable magnetic field H that grows from 0 to some high value that saturates the material, and then decreases back to 0. Doing so would let the final induction B to a non-zero value (even if it started at 0) because of the hysteresis. From there two workarounds exist that make it possible to cancel this residual induction:

1. impose a sinusoidal magnetic field H of decreasing amplitude, which results in successive cycles of decreasing area around 0,
2. or find the proper negative value for the magnetic field H that is such that when going from the prior high H value to this negative value and then back to 0, the induction is perfectly cancelled.

Of course this second strategy is faster, but on the other hand requires a perfect knowledge of the right negative value for H. Figure 3.7 shows the result of this simulation with such a strategy, in a nearly perfect case: after a complete cycle the induction is (almost) cancelled. In practice we measured a remaining adhesion force up to a third of

the maximum adhesion force when simply releasing the electrical current without this kind of compensation.

Hence, the chosen solution consists in smaller negative current in the coil in order to cancel out the field. Obviously, this method would require a perfect knowledge of the hysteresis cycle parameters to be able to exactly bring the field to zero, which would be difficult in practice due to the rounded shape of the hooking system and difficulty to measure the necessary parameters. Instead, we chose to implement a tunable negative current impulse electronics, that is intended to be adjusted experimentally by trial and error on the real hooking system. We thus designed an electrical power command, based on a simple L293D H-bridge, that is able to revert the current in the coil, see Fig. 3.8.

Once tuned, we confirm that the real system is able to release the end-effector as wished.

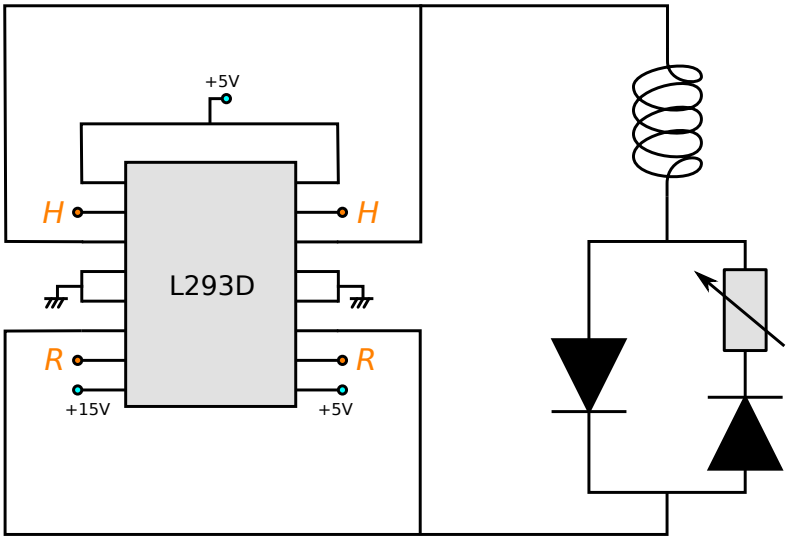

Figure 3.8 – Electrical schematic for the commanded power supply of the magnetic hooking system.

3.2.4 Rotating joint design

There are a few mechanical rules for bearing arrangement which one needs to apply in such a situation:

1. a ring which conveys a charge always in the same (local) direction has to be arranged with clearance,
2. conversely, a ring which sees a rotating charge has to be arranged tight,
3. the bearings can be axially pre-charged or not, towards or against each other,
4. the effective position where the efforts apply is virtually displaced by this pre-charge.

In our case the direction of the effort changes little w.r.t. the direction of the arm of the MonkeyRotor, because it is mainly due to the centrifugal force. As a consequence, the bearing ring *attached to the coil* must be arranged *with clearance*. The *other ring* must be arranged *tight*.

Concerning the axial pre-charge, it is generally used to compensate for axial forces that may occur on the joint, and/or to act on the rigidity of the joint by extending/reducing the length between the points of application of the efforts. In our case, the axial force should remain low as the system evolves in the vertical plane. However, the rigidity is highly desirable because the distance from the joint to the center of mass will imply high radial torques which we don't want to destabilize nor deteriorate the system.

Therefore, the bearings must be arranged in the 'O' configuration, which means that the bearings are stopped

1. *outside* for the rings that are *attached* to the coil, *i.e.*, axially the farthest, and
2. *inside* for the *fixed* rings, *i.e.*, axially the nearest.

Two possible designs ensue, depending on whether the external rings of the bearings are attached to the coil or to the fixed structure. For practical and productibility reasons, we choose the second option, *i.e.*, the external rings of the bearings are attached *to the fixed structure* of the branch. As a consequence, the external rings must be stopped from inside while the internal rings must be stopped from outside. Fig.3.9 illustrates these arrangement choices.

3.3 Implementation

3.3.1 Hardware manufacturing

After fixing small productibility details, the armature of the coil was manufactured by steel turning. Note that two small holes were additionally made in one of the border

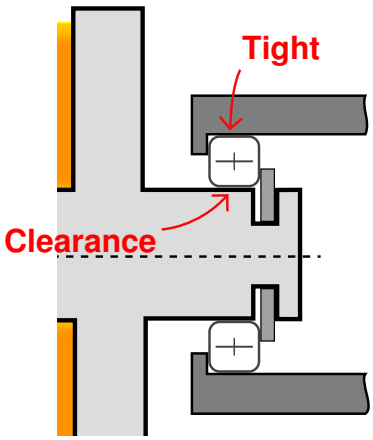

Figure 3.9 – Bearing arrangement for one side of the hooking system. We see from this schematic that the bearing must be assembled to the ground support first, and the hooking system must be mounted with (low) clearance afterwards.

plates in order to pass the electrical wire for the coil. The resulting magnetic hooking system and its electronic command can be seen in Fig. 3.10.

The hooking command is handled by the electrical power circuit described previously, connected to a simple Arduino Uno in order to interface it to the rest of the software via USB, see Fig. 3.10.

A validation of the magnetic hooking system ability to hold the MonkeyRotor was done by means of a dynamometer, see Fig. 3.11. We measured that the coil is able to hold at least 12 kg as wished, without coming loose. The stall occurs at around 14 kg.

Concerning the shape of the MonkeyRotor itself, we need to design and realize a special mechanical structure that is able to host the actuated arm as well as the common electronics for the command of the 4 motors. Given the particularity that the CoM of the quadrotor base and the arm actuator are coaxial, we chose to design a specific body and 3D-print it. The shape of this body has to meet some practical requirements:

1. provide space and attachment facilities for the actuator,
2. feature a maximal strength while being light,
3. provide space for the onboard electronics and battery.

In order to ensure these requirements, we conducted a numerical shape optimization in FreeCAD, see Fig. 3.12. This tool allows to determine a global shape that maxi-

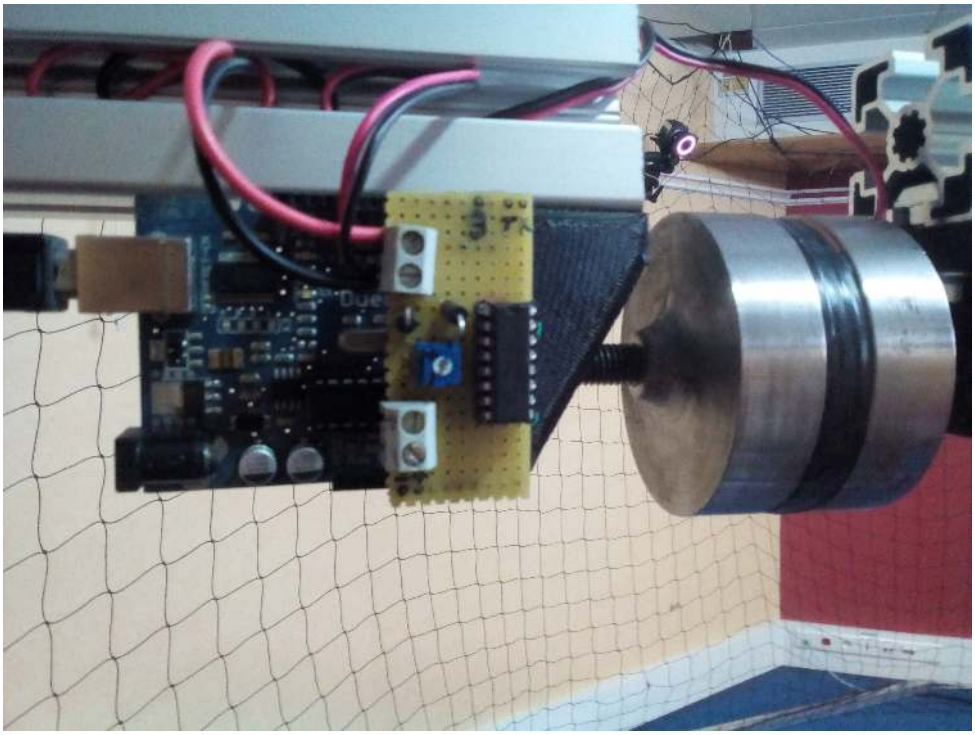

Figure 3.10 – Implementation of the real hooking system.

mizes the strength of the object while minimizing its weight given limit efforts, by means of finite-element analysis and optimization. The resulting shape was then reworked in order to be mechanically sound and to provide the required interfaces with the other components.

3.3.2 Software structure

The middleware genom is then used to interconnect the software components that provide the necessary fonctionnalities. This software allows us to have a modular architecture, based on the interaction of multiple processes associated to the different functions required for the system to be operational. Fig. 3.15 illustrates the global software architecture, which rely on the interaction of multiple components (individual processes) through the genom structure. The communication is undertaken by ROS [1], and the whole machinery is supervised in MATLAB Simulink *via* the genomix client–server feature.

1. https://www.ros.org/

Figure 3.11 – Measure of the magnetic adhesion force with a dynamometer. The desired maximal load of 12 kg can be held as wished.

3.4 Conclusion

In this chapter we have presented the design process and the resulting prototype that we have made. The design of the subparts of the whole system (hooking system and MonkeyRotor body) have been explained in detail.

Although we could not yet achieve the whole trajectory tracking that has been validated in simulation in the previous chapter, several steps have been realized towards this final goal. In particular, we have designed and realized a magnetic hooking system that leverages magnetism to create a sufficient force of adhesion. The hooking system associated with its special electronics is able to fulfil the aim of hooking or releasing the end-effector with enough strength and despite magnetic hysteresis that we

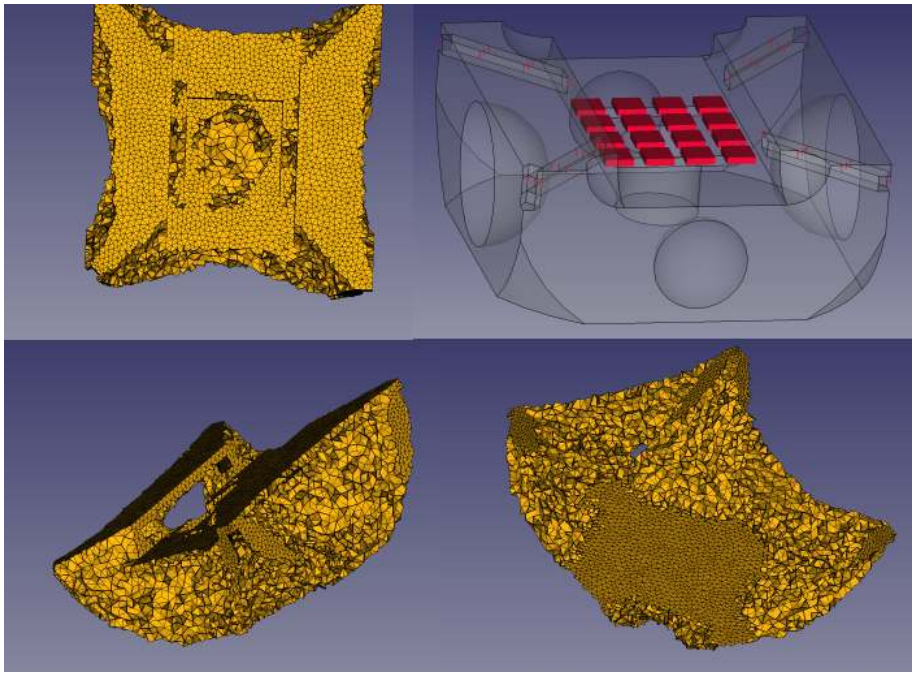

Figure 3.12 – Optimization of the shape of the quadrotor body.

compensated.

The ongoing and future work on this topic consists in exploiting the realized prototype to achieve the tracking of the trajectories that were planned in the previous chapter.

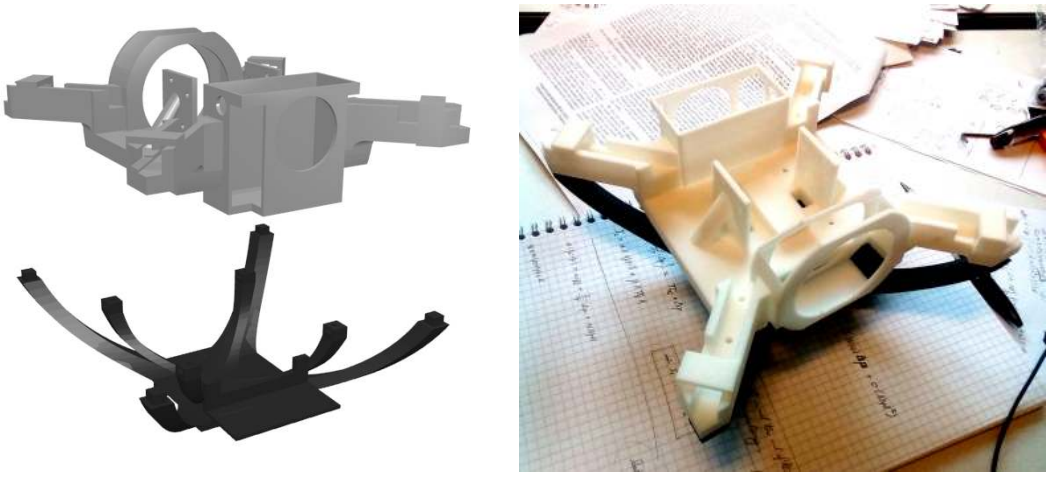

Figure 3.13 – Design of the MonkeyRotor body. On the left, numerical 3D model before printing. On the right, printed parts of the body.

Figure 3.14 – Complete MonkeyRotor prototype with the structure for the branch.

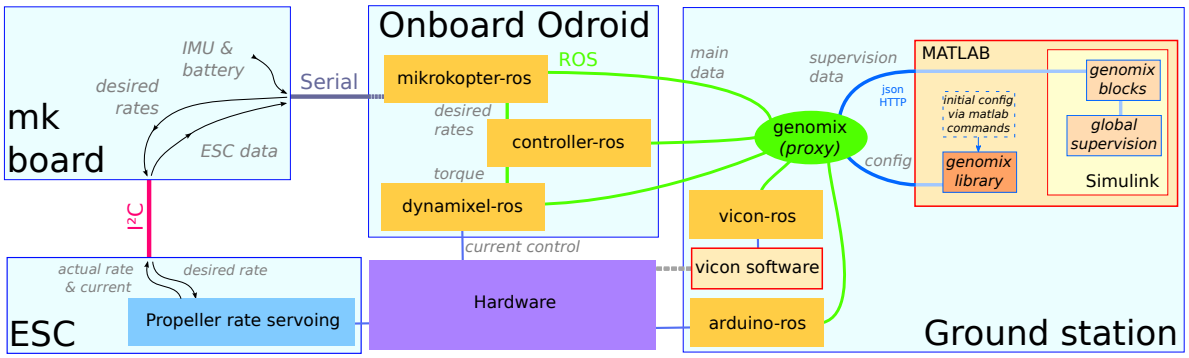

Figure 3.15 – Software architecture based on genom. The orange blocks are genom components which are independent processes communicating through ROS thanks to the genom structure and libraries.

PART II

Part II

CHAPTER 4

TRAJECTORY GENERATION FOR MINIMUM STATE SENSITIVITY

4.1 Introduction

This chapter will introduce the concept of state sensitivity for a dynamical system, and its possible uses in the context of trajectory generation for robotics applications.

At first an intuitive approach will be proposed to give some insights to the reader. A generalization of the method will then derived in a rigorous formalization in the following sections.

The concept arose during the development of our previous study on the Monkey-Rotor, which requires — as most of the modern robotic applications — high precision in the realized task despite the many uncertainties in the system model and actuation. As a starting point, we considered a global control scheme where a trajectory is first planed for the future, and then tracked by the robot by means of a controller. This method has been used for many years, either with offline planning or online replanning as in Model Predictive Control (MPC) [20, 11]. It was proved to be a good way to combine both local robustness thanks to the real-time stabilization provided by the controller, and the ability to include complex features such as collision avoidance and cost minimization in the global robot behavior.

Unfortunately, in some demanding circumstances where high precision is required and/or complex dynamics are excited, an online replanning scheme could be insufficient. This kind of situation is typically encountered with aerial robotics and especially when dealing with physical interaction. First, the performance is highly dependent on the precision of the parameters used in the planning phase and in the controller. Indeed, in many cases the main difficulty lies in the model parameter estimation, or other uncertainties in the actuation, which may represent a complex step of the task im-

plementation, often handled with many fine-tunings. Second, there is no control over the effect of running the closed-loop system with erroneous parameters on a certain planned trajectory, because of the decoupling between the planning (which is done before) and the tracking.

Indeed, trajectory planning gives the opportunity to determine *a priori* a feedforward term that the controller will be able to exploit at the execution stage, which is a very important step in robotics for maximizing performance or guaranteeing other requirements (e.g., minimum energy, time, actuation bound or collision avoidance, etc.). Besides, the robustness is in general a goal that is undertaken by the controller at the execution time, through the derivation of a proper feedback term. For instance, the control community have worked over the years on adaptive and robust strategies, such as H-infinity or passivity-based, that allow a good level of robustness against uncertainties and disturbances, see, e.g., [14, 12]. However, these approaches are mostly "local" and, besides robustness, can hardly tackle other requirements such as feasibility, (e.g., limited actuation, obstacle avoidance), performance and global optimality.

In our case the goal is to study how to generate a feedforward term that directly integrates information about parametric uncertainty, and thus improves the effectiveness of the feedback policy. Said otherwise, we seek a strategy for generating a feedforward that is not decoupled anymore (compared to state of the art strategies) from the feedback, by integrating as much information as possible concerning the closed-loop behavior of the robot at the early planning stage. A few works have been conducted that try to tackle this issue with related approaches [30, 39], however they suffer from a lack of generality because of special dynamics considered and limited robustness because of their "open-loop" nature (coming from the decoupling of the planning and control layers).

A possible approach to try to improve the robustness at the planning stage is, hence, to leverage the knowledge of how the model depends on any uncertain parameter, and try to minimize this dependence in the first place. A few work have treated this problem in the past, e.g., [17, 16, 34] and more recently [5] which proposes a method that we aim at extending in this thesis. In fact, as we will see, it is possible to quantify this dependence through a suitable *state sensitivity* matrix, whose norm can then be exploited as metric in a trajectory generation step. We will now describe more in detail the idea of the state sensitivity and how it can be leveraged in the context of trajectory

generation.

The first section reproduces the idea of the open-loop state sensitivity of [5], which was applied to the MonkeyRotor in Part I. Then the closed-loop state sensitivity is derived, which improves the concept, and tested in simulations.

4.2 Open-loop state sensitivity

As explained just before, we aim in this development at reducing the errors achieved by a robotic system when it tracks a planed trajectory. Two main sources for the tracking errors can be identified, namely 1) the unmodeled perturbations arising at the execution stage, e.g., unforeseen obstacles, complex aerodynamics turbulences, defaults in the mass repartition etc., and 2) the phenomena that are modeled but badly identified because of approximations in the values of the model parameters. For example, the inertia or the aerodynamics coefficients are often difficult to estimate precisely, which affects the closed-loop dynamics of aerial robots in a non-negligible fashion.

The first kind of causes is not treatable at the planning stage by definition. The related induced errors are meant to be handled by the real-time controller (or reactive planning strategy) whose role is to keep the system stable by compensating for them as much as possible. The second kind of causes — discrepancies between the real parameters and the estimated ones used in the controller — are usually considered as unknown perturbations of the dynamics as well. However they are intrinsically different because the model of the system used in the controller gives structure to the way the parameters intervene in the dynamics. This makes it possible to leverage the knowledge of the model to reduce the second kind of induced tracking errors.

Indeed, the analytical model of the dynamics is meant to be only evaluated on the estimated parameters, as the real parameters are unknown. Nevertheless, it can also be exploited to measure the effect of an evaluation on a different set of parameters than the estimated ones. In particular, an infinitesimal variation of the parameters around the estimated ones should have an infinitesimal effect on the state dynamics which can be computed through the corresponding jacobian. We propose then to try to compute this quantity.

We begin by defining a generic non-linear dynamic system described by the differ-

ential relation

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

where $\mathbf{q} \in \mathbb{R}^{n_q}$ is the state of the system, $\mathbf{u} \in \mathbb{R}^{n_u}$ the input fed into it, and $\mathbf{p} \in \mathbb{R}^{n_p}$ a vector of model parameters, including e.g., masses, lengths, and actuation properties. Note that many real-world systems can be described with this kind of model, including non-robotic ones.

Then, recall from Chapter 2 that the *state sensitivity* is defined as the quantity $\mathbf{\Pi}(t) \in \mathbb{R}^{n_q \times n_p}$, such that

$$\mathbf{\Pi}(t) = \frac{\partial \mathbf{q}}{\partial \mathbf{p}}(t). \tag{4.2}$$

This matrix represents how much sensitive the state is to the parameters, with each element $\Pi_{i,j}$ being the sensitivity of one state q_i w.r.t. one of the parameters p_j. As it is solely related to the dynamical model of the system, the analytic knowledge of this model should be sufficient to compute this quantity at first glance. Assuming that one can compute this quantity, it then becomes possible to search a trajectory that, in the null-space of the task constraints, ensures the sensitivity to be as small as possible.

As a consequence, the errors caused by the discrepancy between the (known) estimated parameters and the (unknown) real parameters should be reduced. Though it may not always be possible to completely annihilate these errors, one can reduce them to the minimum possible by means of trajectory optimization with a cost obtained from the state sensitivity matrix.

4.2.1 Simple integrator case

Consider the most elementary possible dynamic system which consists in a simple integrator. Let q be its scalar state and u its scalar input. The dynamics of such a system is given by

$$\begin{cases} q(t) = q_0, & t = 0 \\ \dfrac{dq}{dt}(t) = u(t)p, & t \geq 0, \end{cases} \tag{4.3}$$

where p is a linear parameter. This model could represent for example the linear dynamics of a one-dimensional object of speed q and mass p^{-1}, subject to a force u.

Hence the state at each time can be expressed analytically with

$$q(t) = p \int_0^t u(\tau) d\tau + q_0. \tag{4.4}$$

In that case the sensitivity of the state w.r.t. the parameter is the scalar quantity

$$\begin{aligned} \Pi(t) &= \frac{\partial q}{\partial p}(t) \\ &= \int_0^t u(\tau) d\tau \\ &= \frac{q(t)}{p}. \end{aligned} \tag{4.5}$$

The direct interpretation of this result is that the state of the system after some time t is more sensitive to the parameter when the state itself is greater. Coming back to the one-dimensional object subject to a force u, the result is quite intuitive : the speed q of the object depends more on its mass if it has changed after being pushed more in one direction than in the other.

As a consequence, pushing the object in a way that makes its speed close to the initial one after a duration t_f would ensure that the final speed $q(t_f)$ is less dependent on the mass. Moreover, leading the final speed exactly to the same value as the initial speed is the best and unique way of completely breaking the dependence, even if the speed has changed arbitrary meanwhile.

Fig. 4.1 illustrates this interpretation with a simulation of this integrator in the relevant conditions. As predicted, we can see that the final state varies with the parameter when the input is asymmetric, whereas it does not in the symmetric case. For such a toy system, one could then envisage to plan “robust” trajectories such that the input is as symmetric as possible for reducing the effects of model inaccuracies in the first place.

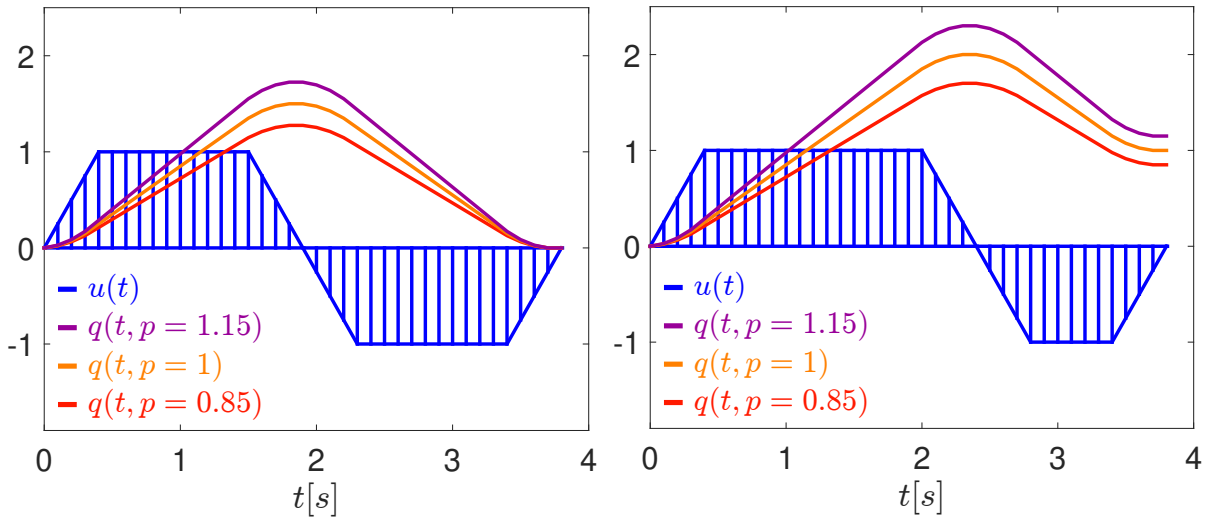

Figure 4.1 – Simulation of a simple integrator with three different parameters for the same input. The input is symmetric on the left, leading to a common final state of zero, and asymmetric on the right, leading to different final states depending on the parameter value.

4.2.2 More complex dynamics

The concept remains the same when the dynamics becomes more complex. Let us add some damping p_2 to the model, such that

$$\begin{cases} q(t) = 0, & t = 0 \\ \dfrac{dq}{dt}(t) = u(t)p_1 - q(t)p_2, & t \geq 0, \end{cases} \tag{4.6}$$

where the mass inverse parameter p was renamed to p_1, and the initial state is set to zero for convenience.

This differential equation can be classically solved in the Laplace domain. Indeed, taking the Laplace transform of eq. (4.6), we get

$$Q(s) = U(s)\frac{\frac{p_1}{p_2}}{\frac{s}{p_2} + 1} \tag{4.7}$$

where $Q(s)$ and $U(s)$ are the Laplace transform of $q(t)$ and $u(t)$, functions of the Laplace variable s. Defining the operator $\mathcal{LPF}_{\omega_0}\{f(t)\}$ which gives a first order unitary low-pass

filter of bandwidth ω_0 of a function $f(t)$, eq. (4.7) rewrites in the time domain

$$q(t) = \frac{p_1}{p_2}\, \mathcal{LPF}_{p_2}\{u(t)\}. \tag{4.8}$$

From there the sensitivity of the state w.r.t. the parameter p_1 is given by the relation

$$\begin{aligned} \Pi_1(t) &= \frac{\partial q}{\partial p_1}(t) \\ &= p_2^{-1}\, \mathcal{LPF}_{p_2}\{u(t)\} \\ &= \frac{q(t)}{p_1} \end{aligned} \tag{4.9}$$

which is similar to the previous case where no damping was modeled. Thus, the same interpretation than before still holds.

In contrast, the expression of the sensitivity related to the damping parameter p_2 is a bit more complex. To derive it, one can leverage the Laplace domain once more. As the derivative is a linear operation, we can compute the sensitivity directly in the Laplace domain,

$$\frac{\partial Q(s)}{\partial p_2} = -U(s)\frac{\frac{p_1}{p_2^2}}{(\frac{s}{p_2}+1)^2} \tag{4.10}$$

and take the inverse Laplace transform to get back to the time domain. We obtain that

$$\begin{aligned} \Pi_2(t) &= \frac{\partial q}{\partial p_2}(t) \\ &= -\frac{p_1}{p_2^2}\, \mathcal{LPF}^2_{p_2}\{u(t)\} \\ &= -\frac{\mathcal{LPF}_{p_2}\{q(t)\}}{p_2}. \end{aligned} \tag{4.11}$$

As a consequence, the state sensitivity w.r.t. the damping parameter p_2 grows with the *low-pass filtered* state. This means that similarly to the behavior of Π_1, the sensitivity of the state w.r.t. p_2 can be reduced at some final time t_f by ensuring that the state $q(t_f)$ gets close to the initial state q_0, but only on a large enough time base, i.e., with a characteristic time $>> p_2^{-1}$. As shown by Fig. 4.2, the state may thus be affected by p_2 depending on the rate it varies at. Moreover, it may happen that the state is very shaky — with a characteristic frequency $>> p_2$ — near t_f, without the sensitivity being increased a lot, because of the low-pass filtering effect.

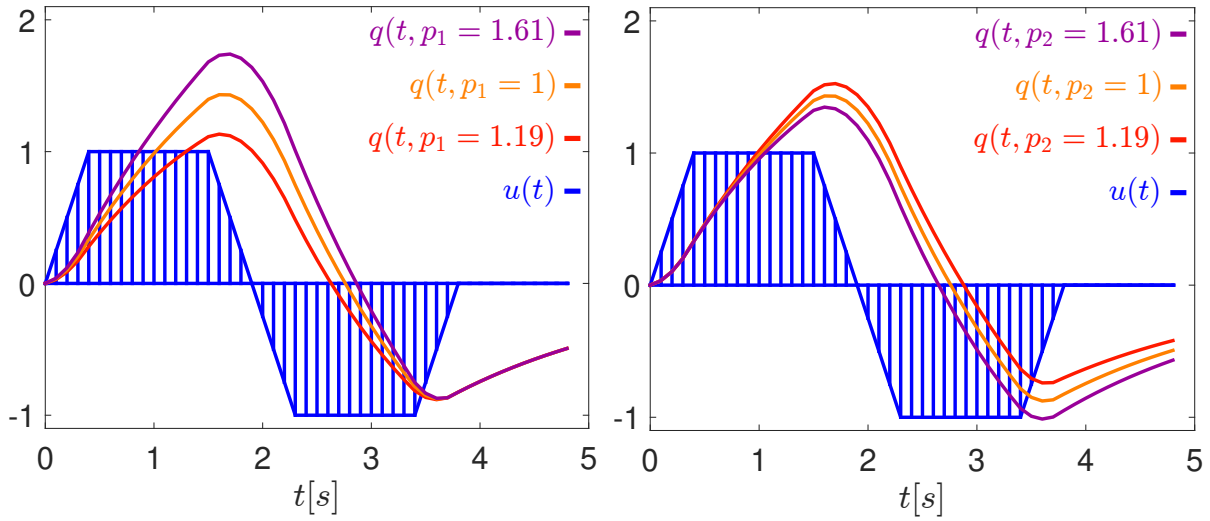

Figure 4.2 – Simulation of a damped integrator with three different values of p_1 on the left and three different values of p_2 on the right, for the same input. On the left, the final state is the same for any parameter value, because the input is symmetric. On the right, the gaps between the three states converge in a low-pass filtered way at the end, as set out in the formal derivation. It can also be noted that the negative sign of the sensitivity is reflected in the order of the curves which is reversed in that case.

4.2.3 Computation of the sensitivity in the general case

As we have seen when the complexity of the dynamics increases — even when remaining linear — it becomes more and more difficult to compute and interpret the sensitivity of the state w.r.t. the parameters. In fact, it is in most cases even impossible to get a closed form expression for $\mathbf{\Pi}(t)$ for a non-linear system. For this reason, it is desirable to seek some general method to compute the value of $\mathbf{\Pi}$ for a generic robot dynamics.

This can be obtained as follows: considering the input as an unknown independent variable *a priori*, we can take the derivative of eq. (4.1) w.r.t. the parameters $\mathbf{p}$ which leads to

$$\begin{cases} \mathbf{\Pi}(0) = \mathbf{0} \\ \dot{\mathbf{\Pi}}(t) = \dfrac{\partial \mathbf{f}}{\partial \mathbf{q}}(\mathbf{q}(t), \mathbf{u}(t), \mathbf{p}) \cdot \mathbf{\Pi}(t) + \dfrac{\partial \mathbf{f}}{\partial \mathbf{p}}(\mathbf{q}(t), \mathbf{u}(t), \mathbf{p}) \end{cases} \tag{4.12}$$

where the initial condition for the sensitivity is zero because the initial state is assumed not to depend on the parameters [1].

1. In certain particular cases the initial state may actually depend on the parameters, e.g., the height of the center of mass of a humanoid robot depends on numerous parameters including the leg lengths.

Though we still do not have a closed form for $\mathbf{\Pi}(t)$, eq. (4.12) provides us with a way to compute it. Indeed, integrating this equation over time gives the sought quantity. Besides, this can be done numerically whenever the dynamics are too complex to solve the differential equation analytically.

Indeed, at this point we have a differential equation for $\mathbf{\Pi}(t)$ that only implies the jacobians $\frac{\partial \mathbf{f}}{\partial \mathbf{q}}$ and $\frac{\partial \mathbf{f}}{\partial \mathbf{p}}$, which are matrices that only depend on the state $\mathbf{q}$, the input $\mathbf{u}$ and the parameters $\mathbf{p}$. Thus, for a certain run of the system with known state and input, one can evaluate these jacobians at any time t for some parameters. In practice, we do not know the set of real parameters, and thus we evaluate these matrices at the best guess we have for the parameters.

This best guess of the parameters vector is also the one used by the controller, hence we call it the *control* parameters vector $\mathbf{p}_c$. Note that the exact definition of the 'real' parameters $\mathbf{p}$ is subject to discussion, as the set of parameters might seem more related to the chosen model than to the real system itself. In the following we consider an ideal paradigm where the real system does have a unique set of parameters associated to, that represents it at best for a given model that we decide to employ. This notion of best representation obviously raises a question of best-fitting criterion choice, we simply consider here that the real parameters are the ones minimizing the difference between the observed real behavior and the modeled one for the sake of simplicity [2].

Based on the resolution of the previous equation (analytically or numerically), one can estimate the value of the state sensitivity matrix. Note that the values of the input over time are required as well as the state and parameters, in order to do so. As said, although the parameters $\mathbf{p}$ appearing in this equation are ones of the real system, they are unknown. However, the known *control* parameters, which are the estimation of the real parameters, denoted by $\mathbf{p}_c$ in the following developments, can be used to evaluate it in practice. The discrepancy between those is supposed to be small for two reasons, 1) because $\mathbf{p}_c$ is a good approximation of $\mathbf{p}$ by definition, and 2) because the trajectory over which eq. (4.12) will be integrated will be constructed with the goal of minimizing the sensitivity, meaning that by construction the effect of the difference between $\mathbf{p}_c$ and $\mathbf{p}$ will be reduced to the minimum possible.

2. This is part of the future work that we plan to achieve concerning the sensitivity notion and applicability.

Given that one can compute a good estimate of the state sensitivity matrix, generating a 'minimum-sensitive' trajectory translates into minimizing a particular cost. For example, a trajectory that is computed such that some norm of the final state sensitivity $||\mathbf{\Pi}(t_f)||$ is minimized should ensure that the effect of the discrepancy between estimated parameters $\mathbf{p}_c$ and real parameters $\mathbf{p}$ on the precision of the reached state $\mathbf{q}(t_f)$ is reduced at best.

This idea has been proposed first in [5] where it has been used to generate feedforward trajectories for a simple vehicule which are, as sought, minimally insensitive to parametric model uncertainty. Moreover, as discussed before, we have also used it in Chap. 2, Sec. 2.2.2 for planning the aerial locomotion trajectories of the MonkeyRotor.

4.3 Closed-loop sensitivity

As we have seen, it is possible to compute the value of the matrix $\mathbf{\Pi}(t)$ in the general case given the trajectories of the state and input vectors over time $(\mathbf{q}(\mathbb{T}), \mathbf{u}(\mathbb{T}))$, where $\mathbb{T} \subset \mathbb{R}^+$ is the considered time interval. We will now put this result in perspective and propose an extension to it.

4.3.1 Motivation

As the careful reader may have noticed, considering the input $\mathbf{u}(t)$ as an independent variable from the state $\mathbf{q}(t)$ is actually a strong hypothesis, since it is equivalent to neglecting the controller of the robot. Indeed, the input is usually a function of the state $\mathbf{q}(t)$ and a target state to be reached $\mathbf{q}^*(t)$, which enables the system to correct perturbations and stabilize itself. In other words, the quantity computed in eq. (4.12) evaluates how sensitive to the parameters the state is, given a *predetermined* input trajectory $\mathbf{u}(\mathbb{T})$ that would not be adapted if the state deviates from the target. For this reason, we chose to call it the *open-loop* state sensitivity.

Note that in a replanning context such as MPC, the input is actually adapted over time since the initial state at each planning iteration reflects the real behavior of the system. Still, this manner of closing the loop is outside the trajectory generation scope and it lies in the external algorithm exploiting the successive planed trajectories at each timestep. Therefore, we keep this denomination of *open-loop* state sensitivity even for

replanning, as this quantity is used with the goal of generating one single trajectory at a time.

Based on this consideration, the direct extension we seek is to try to integrate the controller behavior in the computation of the state sensitivity in order to derive a *closed-loop* sensitivity. By doing this, we aim at making the generation of trajectories based on *closed-loop* sensitivity minimization aware of the particular adopted control strategy, such that the way it affects the dynamics is taken into account at the planning stage. We call this method *control-aware* trajectory generation. Note that this is different from what was done in [5], as the effect of the control policy is taken into account in the trajectory generation, prior to execution.

4.3.2 Derivation

To begin with, we define the desired output trajectory $\mathbf{y}^*(t)$ to be tracked by the system. As our goal is to compute a trajectory of this quantity over the time interval $\mathbb{T}$, let $\mathbf{a} \in \mathbb{R}^{n_a}$ be a finite vector of coefficients parametrizing the trajectory such that, as in Chapter 1,

$$\mathbf{y}^*(t) = \gamma(\mathbf{a}, t) \tag{4.13}$$

with γ the chosen representation function, which can be polynomials for example, as used before in Part I. Note that the vector of coefficients $\mathbf{a}$ being of finite dimension implies that the possible represented trajectories are only a subset of all possible trajectories $(\mathbb{R}^{n_y})^{\mathbb{T}}$, e.g., the continuously derivable ones. However, in our case the trajectory of the state is already enforced to be regular because of the differential equation (4.1) it follows. Moreover, polynomials are known to be able to fit with arbitrary precision any continuous function of the time. This allows us to use polynomials or an equivalent representation function without loss of generality.

Now let the input $\mathbf{u}(t)$ be any differentiable function of the state and of the desired target output $\mathbf{y}^*(t)$. The goal of such control function is, obviously, to achieve $\mathbf{y}(t) \to \mathbf{y}^*(t)$, *i.e.*, to make the output of the system converge towards the desired one. In order to remain general, let it also be a function of an internal state $\boldsymbol{\xi}(t)$ of the controller. This internal state vector may represent, e.g., an integral action, estimation of parameters

for adaptive control, and so on. We define the control laws as

$$\begin{cases} \boldsymbol{\xi}(0) = \boldsymbol{\xi}_0 \\ \dot{\boldsymbol{\xi}}(t) = \mathbf{g}(\mathbf{q}(t), \boldsymbol{\xi}(t), \mathbf{a}, t, \mathbf{p}_c) \\ \mathbf{u}(t) = \mathbf{h}(\mathbf{q}(t), \boldsymbol{\xi}(t), \mathbf{a}, t, \mathbf{p}_c) \end{cases} \tag{4.14}$$

where the whole desired trajectory $\mathbf{y}^*(\mathbb{T})$ is given to the controller through the vector $\mathbf{a}$. Indeed, in some cases the sole current target state $\mathbf{y}^*(t)$ may be sufficient to the controller, e.g., if it implements a simple proportional feedback law, or if the target is constant (in a regulation setup). But generally several derivatives are also required depending on the complexity of the dynamics and of the chosen control law. With this formulation the possibly necessary derivatives of order o can computed inside the controller functions $\mathbf{g}$ and $\mathbf{h}$ by means of the representation function γ,

$$\frac{d^o \mathbf{y}^*}{dt^o}(t) = \frac{d^o \gamma(\mathbf{a}, t)}{dt^o}. \tag{4.15}$$

Now if we consider (4.1) and take the derivative of it w.r.t. the parameters, we obtain a new equation which is different from eq. (4.12) because it integrates the full closed-loop dynamics including the controller. To do so we define the quantity

$$\boldsymbol{\Theta}(t) = \frac{\partial \mathbf{u}(t)}{\partial \mathbf{p}} \in \mathbb{R}^{n_u \times n_p} \tag{4.16}$$

which we will be calling the *input sensitivity* from now on.

The sensitivity differential equation becomes

$$\begin{cases} \boldsymbol{\Pi}(0) = \mathbf{0} \\ \dot{\boldsymbol{\Pi}}(t) = \dfrac{\partial \mathbf{f}}{\partial \mathbf{q}} \cdot \boldsymbol{\Pi}(t) + \dfrac{\partial \mathbf{f}}{\partial \mathbf{u}} \cdot \boldsymbol{\Theta}(t) + \dfrac{\partial \mathbf{f}}{\partial \mathbf{p}} \end{cases} \tag{4.17}$$

where the evaluation arguments $(\mathbf{q}(t), \mathbf{u}(t), \mathbf{p})$ of the jacobian matrices have been omitted for the sake of clarity.

This equation has only one more term than before, however integrating it is not obvious because $\boldsymbol{\Theta}(t)$ is not known *a priori*. It is in fact possible to compute it through

eq. (4.14) which we can rewrite

$$\boldsymbol{\Theta}(t) = \frac{\partial \mathbf{h}}{\partial \mathbf{q}} \cdot \boldsymbol{\Pi}(t) + \frac{\partial \mathbf{h}}{\partial \boldsymbol{\xi}} \cdot \frac{\partial \boldsymbol{\xi}}{\partial \mathbf{p}}(t) \tag{4.18}$$

after derivation w.r.t. $\mathbf{p}$, where the evaluation arguments $(\mathbf{q}(t), \boldsymbol{\xi}(t), \mathbf{a}, t, \mathbf{p}_c)$ of the two jacobian matrices have been omitted here again.

Unfortunately there is still one unknown term here, the *internal state sensitivity* which we denote as

$$\boldsymbol{\Pi}_{\boldsymbol{\xi}}(t) = \frac{\partial \boldsymbol{\xi}(t)}{\partial \mathbf{p}}. \tag{4.19}$$

This issue is resolved by applying the same procedure as before to the first controller equation (4.14), namely taking the derivative w.r.t. the parameters directly of the differential equation. This leads to a new differential relation on the internal state sensitivity

$$\begin{cases} \boldsymbol{\Pi}_{\boldsymbol{\xi}}(0) = \mathbf{0} \\ \dot{\boldsymbol{\Pi}}_{\boldsymbol{\xi}}(t) = \dfrac{\partial \mathbf{g}}{\partial \mathbf{q}} \cdot \boldsymbol{\Pi}(t) + \dfrac{\partial \mathbf{g}}{\partial \boldsymbol{\xi}} \cdot \boldsymbol{\Pi}_{\boldsymbol{\xi}}(t) \end{cases} \tag{4.20}$$

which can be forward integrated to compute the effective value of this matrix. Here again the initial value of $\boldsymbol{\Pi}_{\boldsymbol{\xi}}$ is considered to be null.

To synthesize, it is possible to compute the closed-loop state sensitivity by integrating a set of differential equations. 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 vector arguments $\mathbf{y}$. With this shorthand, the set of differential equations becomes

$$\begin{cases} \boldsymbol{\Pi}(0) = \mathbf{0} \\ \boldsymbol{\Pi}_{\boldsymbol{\xi}}(0) = \mathbf{0} \\ \dot{\boldsymbol{\Pi}}(t) = \mathbf{f}_{,\mathbf{q}} \cdot \boldsymbol{\Pi}(t) + \mathbf{f}_{,\mathbf{u}} \cdot \boldsymbol{\Theta}(t) + \mathbf{f}_{,\mathbf{p}} \\ \dot{\boldsymbol{\Pi}}_{\boldsymbol{\xi}}(t) = \mathbf{g}_{,\mathbf{q}} \cdot \boldsymbol{\Pi}(t) + \mathbf{g}_{,\boldsymbol{\xi}} \cdot \boldsymbol{\Pi}_{\boldsymbol{\xi}}(t) \\ \boldsymbol{\Theta}(t) = \mathbf{h}_{,\mathbf{q}} \cdot \boldsymbol{\Pi}(t) + \mathbf{h}_{,\boldsymbol{\xi}} \cdot \boldsymbol{\Pi}_{\boldsymbol{\xi}}(t) \end{cases} . \tag{4.21}$$

Note that, of course, the true parameters of the system $\mathbf{p}$ are not known. Hence, this set of differential equations cannot be integrated for evaluating the sensitivity of the true system w.r.t. its parameters. However, as evoked before, one can evaluate the values of the necessary jacobians at the control parameters $\mathbf{p}_c$ instead. The consequence is

that the resulting computed sensitivity corresponds to the one of a virtual system that would have the parameters $\mathbf{p}_c$. Although this may not seem to be the sought value, the difference is actually supposed to have negligible impact on the results, because by nature the control parameters are a good approximation of the true parameters even if tainted with small errors, *i.e.*, $\mathbf{p}_c \approx \mathbf{p}$. Hence, the difference between $\mathbf{\Pi}(\mathbf{p})$ and $\mathbf{\Pi}(\mathbf{p}_c)$ is of second order and can be neglected [3].

4.4 Application to robotic trajectory generation

In this section we will describe how one can exploit the previous closed-loop state sensitivity computations for generating 'minimum-sensitive' trajectories for two representative robots. The method will be applied to a unicycle robot, which is considered to be a good case study in order to test the applicability of the theory. Indeed, it is simple and still presents interesting properties such as the differential flatness, and a (planar) quadrotor which is also a flat system while having a more complex and challenging dynamics and control. We first derive the dynamics and controller of the two robots.

4.4.1 Unicycle dynamics and control

Let $\mathbf{q} = [x\ y\ \theta]^\top \in \mathbb{R}^3$ be the unicycle state in a world frame $\mathcal{F}_W = \{\boldsymbol{O}_W;\ \mathbf{x}_W,\ \mathbf{y}_W\}$, with $(x,\ y)$ being the planar position in $\mathcal{F}_W$ and θ the unicycle heading, see Fig. 4.3. Let also $(v,\ \omega)$ be the unicycle linear and angular velocities and $(\omega_R,\ \omega_L)$ the right and left wheel spinning velocities. As it is well-known, these velocities are related by

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{2b} & -\frac{r}{2b} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix} = \mathbf{S} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix} \tag{4.22}$$

where r is the wheel radius and b the distance between the wheels, and where the introduced matrix $\mathbf{S}$ is constant for the robot. The differential drive configuration is retained in our case, which means that the actual inputs of the unicycle are the wheel velocities $(\omega_R,\ \omega_L)$. We thus set $\mathbf{u} = [\omega_R\ \omega_L]^T$ as the unicycle control inputs. Because of this choice, any uncertainty in the calibration parameters r and b directly affects the

3. Further justifications will be given in the next chapter.

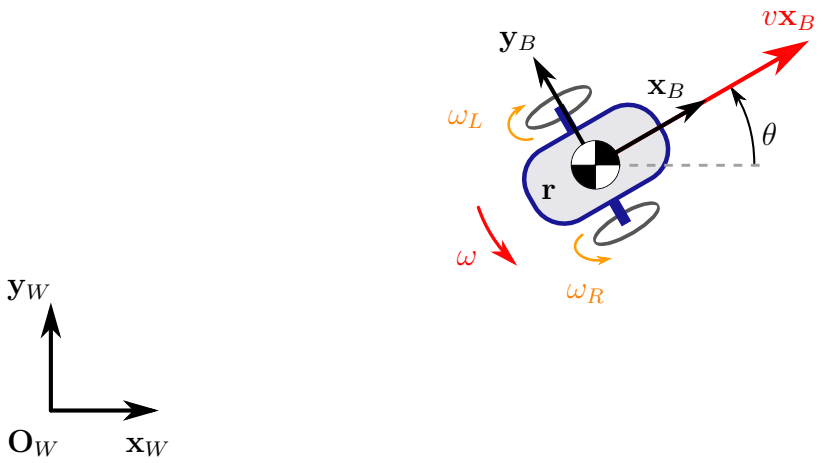

Figure 4.3 – Illustration of the main quantities characterizing the unicycle model

system dynamics. Therefore, we take $\mathbf{p} = [r \ b]^T$ as the vector of system parameters w.r.t. which the closed-loop state sensitivity will be evaluated.

With these settings, the unicycle dynamics has the expression

$$\dot{\mathbf{q}} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \mathbf{S}(\mathbf{p})\mathbf{u} = \mathbf{f}(\mathbf{q}, \mathbf{u}, \mathbf{p}). \tag{4.23}$$

The chosen control task is that of letting the output $\mathbf{r} = [x \ y]^T \in \mathbb{R}^2$, which is the unicycle planar position, tracking a reference trajectory $\mathbf{r}_d(t) \in \mathbb{R}^2$. We solve this tracking control task by implementing a dynamic feedback linearization (DFL) controller with an integral action, with the aim of guaranteeing the best possible tracking performance in the nominal case where $\mathbf{p}_c = \mathbf{p}$, see, e.g., [51].

This control strategy can be summarized as follows: let $\boldsymbol{\xi} = [\xi_v \ \xi_x \ \xi_y]^T \in \mathbb{R}^3$ be the controller states, where ξ_v represents the dynamic extension of the unicycle linear velocity v, and (ξ_x, ξ_y) are the two states of the integral action. Let also the matrix $\mathbf{A}(\mathbf{q}, \boldsymbol{\xi}) \in \mathbb{R}^{2\times 2}$ be defined as

$$\mathbf{A}(\mathbf{q}, \boldsymbol{\xi}) = \begin{bmatrix} \cos(\theta) & -\xi_v \sin(\theta) \\ \sin(\theta) & \xi_v \cos(\theta) \end{bmatrix}. \tag{4.24}$$

Differentiating twice the unicycle position $\mathbf{r}(\mathbf{q})$ w.r.t. time yields

$$\ddot{\mathbf{r}} = \begin{bmatrix} \cos(\theta) & -\xi_v \sin(\theta) \\ \sin(\theta) & \xi_v \cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{v} \\ \omega \end{bmatrix} = \mathbf{A}(\mathbf{q},\, \boldsymbol{\xi}) \begin{bmatrix} \dot{v} \\ \omega \end{bmatrix} \tag{4.25}$$

Now define the following vectors

$$\begin{cases} \dot{\mathbf{r}}_\xi = [\cos(\theta)\xi_v \;\; \sin(\theta)\xi_v]^T \\ \boldsymbol{\xi}_{xy} = [\xi_x \;\; \xi_y]^T \\ \eta = \ddot{\mathbf{r}}_d + k_v(\dot{\mathbf{r}}_d - \dot{\mathbf{r}}_\xi) + k_p(\mathbf{r}_d - \mathbf{r}) + k_i \boldsymbol{\xi}_{xy} \end{cases} \tag{4.26}$$

where $k_v > 0$, $k_p > 0$ and $k_i > 0$ are suitable control gains. As detailed in [51], the dynamics of the control states $\boldsymbol{\xi}$ can then be written as

$$\dot{\boldsymbol{\xi}} = \begin{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{A}^{-1}\eta \\ \mathbf{r}_d - \mathbf{r} \end{bmatrix} = \mathbf{g}(\boldsymbol{\xi},\, \mathbf{q},\, \mathbf{r}_d(t)) \tag{4.27}$$

and the unicycle control inputs as

$$\mathbf{u} = \mathbf{S}_c^{-1} \begin{bmatrix} \xi_v \\ \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{A}^{-1}\eta \end{bmatrix} = \mathbf{h}(\boldsymbol{\xi},\, \mathbf{q},\, \mathbf{r}_d(t),\, \mathbf{p}_c). \tag{4.28}$$

Note that in eq. (4.28) the calibration matrix $\mathbf{S}_c$ is supposed to be correctly evaluated on the nominal control parameters $\mathbf{p}_c$.

4.4.2 Planar quadrotor dynamics and control

Let $\mathcal{F}_W = \{\boldsymbol{O}_W;\, \mathbf{x}_W,\, \mathbf{z}_W\}$ be a world frame and $\mathcal{F}_B = \{\boldsymbol{O}_B;\, \mathbf{x}_B,\, \mathbf{z}_B\}$ the body frame attached to the quadrotor center of mass, with $\mathbf{z}_B$ aligned with the thrust direction, see Fig. 4.4. In the planar quadrotor case, the state consists of the quadrotor position $\mathbf{r} = (x,\, z)$ and linear velocity $\mathbf{v} = (v_x,\, v_z)$ in $\mathcal{F}_W$, and of the quadrotor body orientation θ and angular velocity ω with, thus, $\mathbf{q} = [\mathbf{r}^T \;\; \mathbf{v}^T \;\; \theta \;\; \omega]^T \in \mathbb{R}^6$. Similarly to the previous unicycle case, we can distinguish the 'effective' inputs $(f,\, \tau)$ which are the total thrust and torque, and the actual system inputs $(w_R,\, w_L)$ which are the right and left propeller

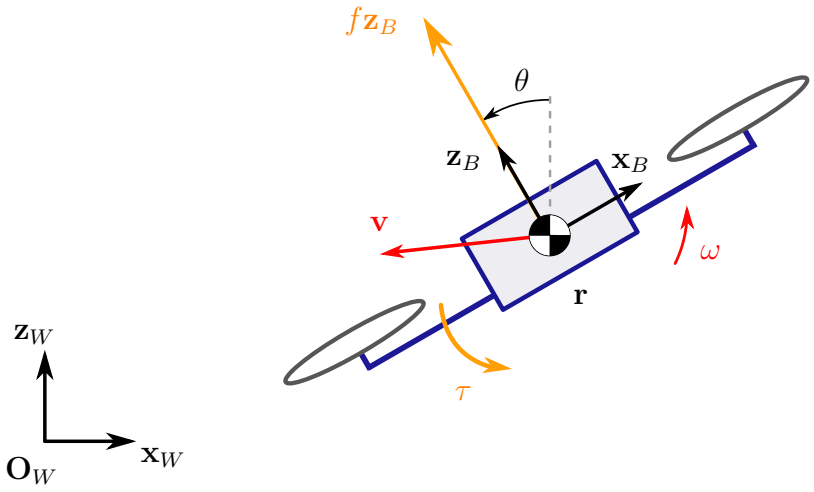

Figure 4.4 – Illustration of the main quantities characterizing the quadrotor model.

speeds. These four values are related by

$$\begin{bmatrix} f \\ \tau \end{bmatrix} = \begin{bmatrix} k_f & k_f \\ k_\tau & -k_\tau \end{bmatrix} \begin{bmatrix} w_R \\ w_L \end{bmatrix} = \mathbf{T} \begin{bmatrix} w_R \\ w_L \end{bmatrix}, \tag{4.29}$$

where k_f and k_τ are, in first approximation, calibration parameters depending on the propeller characteristics, see, e.g., [38]. Throughout the following developments, we will then take $\mathbf{u} = [w_R \; w_L]^T$ as the quadrotor control inputs.

The quadrotor dynamical model considered here is

$$\begin{cases} \dot{\mathbf{r}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= \begin{bmatrix} 0 \\ -g \end{bmatrix} + \dfrac{f}{m} \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix} - \mathbf{B}(\theta)\mathbf{v} \\ \dot{\theta} &= \omega \\ \dot{\omega} &= \dfrac{\tau}{I} \end{cases} \tag{4.30}$$

with m and I being the quadrotor mass and inertia, and g the gravity acceleration magnitude. Note that the control inputs $\mathbf{u} = [w_R \; w_L]^T$ enter in (4.30) *via* (4.29). This then induces a dependency on the propeller characteristics $(k_f, \, k_\tau)$. Furthermore, the term $\mathbf{B}(\theta)\mathbf{v}$ is meant to model a body-frame air drag with possible different magnitudes

along the horizontal and vertical quadrotor axes $\mathbf{x}_B$ and $\mathbf{z}_B$. Letting $\mathbf{R}(\theta) \in SO(2)$ be the 2D rotation matrix from $\mathcal{F}_W$ to $\mathcal{F}_B$, matrix $\mathbf{B}(\theta)$ is defined as

$$\mathbf{B}(\theta) = \mathbf{R}(\theta) \begin{bmatrix} b_x & 0 \\ 0 & b_z \end{bmatrix} \mathbf{R}^T(\theta) \tag{4.31}$$

where $b_x \geq 0$ and $b_z \geq 0$ are the body-frame drag coefficients along $\mathbf{x}_B$ and $\mathbf{z}_B$, respectively. Finally, the system parameter vector considered for the closed-loop sensitivity optimization is taken as

$$\mathbf{p} = \begin{bmatrix} \dfrac{k_f}{m} & \dfrac{k_\tau}{I} & b_x & b_z \end{bmatrix}^T \in \mathbb{R}^4.$$

Indeed, as clear from (4.29–4.30), the quadrotor dynamics is only affected by the ratios k_f/m and k_τ/I and not by the individual values of these parameters.

The control task is, as before, the one of tracking of a reference trajectory $\mathbf{r}_d(t)$ for the quadrotor position $\mathbf{r}$. Analogously to the unicycle case, we implement a DFL controller with an integral term for obtaining the best possible tracking performance of $\mathbf{r}_d(t)$ in the nominal case where $\mathbf{p}_c = \mathbf{p}$, see, e.g., [48]. In the quadrotor case, the controller states are $\boldsymbol{\xi} = [\xi_f \ \xi_{df} \ \xi_x \ \xi_z] \in \mathbb{R}^4$, where ξ_f and ξ_{df} represent the two dynamic extensions of the thrust input f, and (ξ_x, ξ_z) are again the internal states of the integral action.

We construct the controller by differentiating four times the quadrotor position $\mathbf{r}$ w.r.t. time,

$$\ddddot{\mathbf{r}} = \mathbf{A}(\mathbf{q}, \boldsymbol{\xi}, \mathbf{p}_c) \begin{bmatrix} \ddot{f} \\ \tau \end{bmatrix} + \mathbf{b}(\mathbf{q}, \boldsymbol{\xi}, \mathbf{p}_c) \tag{4.32}$$

where the detailed expressions of $\mathbf{A}(\mathbf{q}, \boldsymbol{\xi}, \mathbf{p}_c) \in \mathbb{R}^{2\times 2}$ and $\mathbf{b}(\mathbf{q}, \boldsymbol{\xi}, \mathbf{p}_c) \in \mathbb{R}^2$ can be found in [48]. Let now

$$\begin{cases} \ddot{\mathbf{r}}_\xi &= \begin{bmatrix} 0 \\ -g_c \end{bmatrix} + \dfrac{\xi_f}{m_c} \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix} - \mathbf{B}(\theta)\mathbf{v} \\ \dddot{\mathbf{r}}_\xi &= \dfrac{\xi_{df}}{m_c} \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix} - \dfrac{\xi_f}{m_c} \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \omega - \dot{\mathbf{B}}\mathbf{v} - \mathbf{B}\ddot{\mathbf{r}}_\xi \\ \boldsymbol{\xi}_{xz} &= [\xi_x \ \xi_z]^T \\ \eta &= \ddddot{\mathbf{r}}_d + k_j(\dddot{\mathbf{r}}_d - \dddot{\mathbf{r}}_\xi) + k_a(\ddot{\mathbf{r}}_d - \ddot{\mathbf{r}}_\xi) + k_v(\dot{\mathbf{r}}_d - \mathbf{v}) + \\ &\quad + k_p(\mathbf{r}_d - \mathbf{r}) + k_i\boldsymbol{\xi}_{xz} \end{cases} \tag{4.33}$$

where $k_j > 0$, $k_a > 0$, $k_v > 0$, $k_d > 0$, $k_p > 0$ and $k_i > 0$ are suitable control gains. The dynamics of the control states can then be written as

$$\begin{bmatrix} \dot{\xi}_f \\ \dot{\xi}_{df} \\ \dot{\boldsymbol{\xi}}_{xz} \end{bmatrix} = \begin{bmatrix} \xi_{df} \\ \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{A}^{-1}(\eta - \mathbf{b}) \\ \mathbf{r}_d - \mathbf{r} \end{bmatrix} = \mathbf{g}(\boldsymbol{\xi},\, \mathbf{q},\, \mathbf{r}_d(t),\, \mathbf{p}_c) \tag{4.34}$$

and the quadrotor control inputs are given by

$$\mathbf{u} = \mathbf{T}_c^{-1} \begin{bmatrix} \xi_f \\ \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{A}^{-1}(\eta - \mathbf{b}) \end{bmatrix} = \mathbf{h}(\boldsymbol{\xi},\, \mathbf{q},\, \mathbf{r}_d(t),\, \mathbf{p}_c). \tag{4.35}$$

4.4.3 Trajectory generation

Now that the system dynamics and control strategies have been defined, we have closed form expressions for the required vector functions $\mathbf{f}$, $\mathbf{g}$ and $\mathbf{h}$. This allows us to compute the necessary terms in eq. (4.21), namely the three jacobians of the dynamics $\mathbf{f}_{,\mathbf{q}}$, $\mathbf{f}_{,\mathbf{u}}$, $\mathbf{f}_{,\mathbf{p}}$, the two jacobians of the control dynamics $\mathbf{g}_{,\mathbf{q}}$, $\mathbf{g}_{,\boldsymbol{\xi}}$, and the two jacobians of the control law $\mathbf{h}_{,\mathbf{q}}$, and $\mathbf{h}_{,\boldsymbol{\xi}}$. The differential system (4.21) can then be solved along a particular trajectory $\mathbf{a}$ that is submitted to the system, yielding the sought sensitivity matrix $\boldsymbol{\Pi}$.

This allows us to set up a trajectory optimization problem that aims at reducing this sensitivity, as announced at the beginning of this chapter. Given the system dynamics (4.23), a reference trajectory $\mathbf{r}_d(t) = \gamma(\mathbf{a},\, t)$ defined over a given time interval $t \in \mathbb{T} = [t_0,\, t_f]$, and the tracking controller (4.27–4.28) or (4.34–4.35), we consider two possible optimization problems of interest, both seeking the optimal trajectory vector $\mathbf{a}_{opt}$, namely:

- $\mathbf{a}_{opt} = \arg\min_{\mathbf{a}\in\mathcal{A}} \|\boldsymbol{\Pi}(t_f)\|^2$, or (4.36)
- $\mathbf{a}_{opt} = \arg\min_{\mathbf{a}\in\mathcal{A}} \int_{t_0}^{t_f} \|\boldsymbol{\Pi}(\tau)\|^2 \mathrm{d}\tau$ (4.37)

where $\|\cdot\|$ is a suitable norm for the state sensitivity matrix $\boldsymbol{\Pi}$, and $\mathcal{A}$ is the set of possible values for the optimization variables $\mathbf{a}$. The first problem (4.36) focuses on optimizing the perturbed tracking performance of $\mathbf{r}_d(t)$ at the final time t_f. This is the

most common robotic task, which is relevant when needing to reach a specific location — or, more generally, desired output — or for instance grasp an object at the final time t_f with high accuracy. On the other hand, the second problem aims at optimizing the average perturbed tracking performance of $\mathbf{r}_d(t)$ during the whole trajectory duration. This is more relevant when one wants to minimize deviations from the desired trajectory in the whole time interval $t \in \mathbb{T}$. It may be required for instance when needing to avoid collisions throughout the trajectory.

Problems (4.36–4.37) are constrained minimization problems that can be addressed with any suitable off-the-shelf solver. Note that among all the possible solving methods, the fastest ones require an analytical expression for the gradient of the cost w.r.t. the optimization variable $\mathbf{a}$. Heretofore, this gradient expression have not yet been derived. Though it would be possible to implement a solving method that does not use the gradient such as, e.g., simulated annealing [33], we remark that it is in fact possible to compute this gradient for the sensitivity cost, as we will detail in the next section.

In our case, a simple gradient descent algorithm with linear constraints will be used. To this end, we consider a scenario in which initial and final values are given for $\mathbf{r}_d(t)$ and a number of its time derivatives, e.g., given initial and final positions, velocities, accelerations, and so on. These constraints, defining the admissible set $\mathcal{A}$, can be written in a linear form as $\mathbf{M}\mathbf{a} = \mathbf{d}$, where the vector $\mathbf{d}$ is the given set of initial and final values for $\mathbf{r}_d(t)$, and the matrix $\mathbf{M}$ depends on the choice of the trajectory representation function γ.

The polynomial trajectories that we use in this Thesis are as described in Chap. 1, eq. (1.22).

Vector $\mathbf{a}$ can then be optimized with a null-space approach by starting from an initial guess satisfying the constraint, e.g.,

$$\mathbf{a}_0 = \mathbf{M}^\dagger \mathbf{d}, \tag{4.38}$$

and implementing the update law

$$\mathbf{a}_{n+1} = \mathbf{a}_n + k_1 \mathbf{M}^\dagger(\mathbf{d} - \mathbf{M}\mathbf{a}_n) + k_2(\mathbf{I} - \mathbf{M}^\dagger\mathbf{M})\nu \ , \quad \forall n > 0 \tag{4.39}$$

with the vector $\nu \in \mathbb{R}^{n_a}$ being the negative gradient of the cost functions in (4.36–4.37), and $k_1 > 0$, $k_2 > 0$ suitable gains. The update mechanism can be stopped whenever

the gradient norm becomes small enough. Note that since problems (4.36) and (4.37) are in general non-convex in $\mathbf{a}$, the update law (4.39) can only guarantee convergence towards a local minimum.

As for the choice of an appropriate matrix norm $\|\cdot\|$, many possibilities exist, e.g., determinant, trace, condition number and so on. In this chapter we chose to use the Frobenius matrix norm, *i.e.*, for a matrix $\mathbf{M} \in \mathbb{R}^{n\times m}$,

$$\|\mathbf{M}\|^2 = \sum_{i,j} m_{i,j}^2. \tag{4.40}$$

For a matrix $\mathbf{M}(\mathbf{x})$ function of a vector $\mathbf{x} \in \mathbb{R}^{n_x}$, this norm definition leads to the following derivative

$$\begin{aligned}\frac{\partial\|\mathbf{M}(\mathbf{x})\|^2}{\partial\mathbf{x}} &= 2\|\mathbf{M}(\mathbf{x})\| \cdot \frac{\partial\|\mathbf{M}(\mathbf{x})\|}{\partial\mathbf{M}(\mathbf{x})} \cdot \frac{\partial\mathbf{M}(\mathbf{x})}{\partial\mathbf{x}} \\ &= 2\|\mathbf{M}(\mathbf{x})\| \cdot \frac{\mathbf{M}(\mathbf{x})}{\|\mathbf{M}(\mathbf{x})\|} \cdot \frac{\partial\mathbf{M}(\mathbf{x})}{\partial\mathbf{x}} \\ &= 2\mathbf{M}(\mathbf{x}) \cdot \frac{\partial\mathbf{M}(\mathbf{x})}{\partial\mathbf{x}}\end{aligned} \quad . \tag{4.41}$$

Note that in this expression, $\frac{\partial\mathbf{M}(\mathbf{x})}{\partial\mathbf{x}} = \mathbf{T}$ is a tensor of dimensions (n, m, n_x) and thus, the product $\mathbf{M}(\mathbf{x}) \cdot \frac{\partial\mathbf{M}(\mathbf{x})}{\partial\mathbf{x}} \in \mathbb{R}^{1\times n_x}$ is a matrix-tensor product which obeys the rule

$$(\mathbf{M}\cdot\mathbf{T})_{1,k} = \sum_{i=1}^{n}\sum_{j=1}^{m} m_{i,j}\ t_{i,j,k} \ , \quad \forall k \leq n_x. \tag{4.42}$$

Defining the tensor $\mathbf{\Pi}_{,\mathbf{a}} = \frac{\partial\mathbf{\Pi}}{\partial\mathbf{a}} \in \mathbb{R}^{n_q\times n_p\times n_a}$, this results in the gradient expression

$$\nu = -\frac{\partial\|\mathbf{\Pi}(t_f)\|^2}{\partial\mathbf{a}} = -2\mathbf{\Pi}(t_f)\cdot\mathbf{\Pi}_{,\mathbf{a}}(t_f) \tag{4.43}$$

for Problem (4.36), and

$$\begin{aligned}\nu = &-\frac{\partial\int_{t_0}^{t_f}\|\mathbf{\Pi}(\tau)\|^2\mathrm{d}\tau}{\partial\mathbf{a}} = -\int_{t_0}^{t_f}\left(\frac{\partial\|\mathbf{\Pi}(\tau)\|^2}{\partial\mathbf{a}}\right)\mathrm{d}\tau = \\ = &-2\int_{t_0}^{t_f}\mathbf{\Pi}(\tau)\cdot\mathbf{\Pi}_{,\mathbf{a}}(\tau)\mathrm{d}\tau\end{aligned}$$

for Problem (4.37).

4.4.4 Gradient derivation

In the previous algorithm the required gradient $\mathbf{\Pi}_{,\mathbf{a}}$ may be estimated by numerical methods such as finite difference or complex step differentiation, see [59], however the dimension of the considered gradient is quite high, which makes such numerical computations too slow in practice. For example, in the case of the unicycle with trajectory polynomials of dimension 10, we have a gradient of dimension $3 \times 2 \times 10$, which means that a numerical approximation *via* finite differences may need approximatively 61 evaluations of the sensitivity at each step of the update law (4.39), which can be computationally heavy (and, of course, the situation can only get worse with more complex dynamics, parameters or finer trajectory parametrizations). Instead, we propose to derive a formal expression for the gradient of the sensitivity cost based on the analytical expressions of eq. (4.17).

As discussed before, $\partial\mathbf{\Pi}/\partial\mathbf{a}$ is a tensor quantity in $\mathbb{R}^{n_q \times n_p \times n_a}$. For simplifying the derivations we then work out the expression for the gradient $\partial\mathbf{\Pi}/\partial a_i$ w.r.t. each individual i-th component of the trajectory vector $\mathbf{a}$.

Let then

$$\mathbf{\Pi}_{,a_i}(t) = \left.\frac{\partial\mathbf{\Pi}(t)}{\partial a_i}\right|_{\mathbf{p}=\mathbf{p}_c} \in \mathbb{R}^{n_q \times n_p} \tag{4.44}$$

be the sought matrix gradient of the system state sensitivity w.r.t. the optimization variable a_i, and let also

$$\mathbf{\Pi}_{\boldsymbol{\xi},a_i}(t) = \left.\frac{\partial\mathbf{\Pi}_\xi(t)}{\partial a_i}\right|_{\mathbf{p}=\mathbf{p}_c} \in \mathbb{R}^{n_\xi \times n_p} \tag{4.45}$$

be the matrix gradient of the controller state sensitivity w.r.t. a_i. The matrix $\mathbf{\Pi}_{\boldsymbol{\xi}_{,a_i}}$ represents the gradient of the control state sensitivity w.r.t. the optimization variable a_i, and will be needed for evaluating $\mathbf{\Pi}_{,a_i}$.

Taking the raw time derivative of eq. (4.21), we obtain that

$$\begin{cases} \dot{\mathbf{\Pi}}_{,a_i} = \left[\dfrac{\partial \mathbf{f}_{,\mathbf{q}}}{\partial \mathbf{q}} \circ \dfrac{\partial \mathbf{q}}{\partial a_i} + \dfrac{\partial \mathbf{f}_{,\mathbf{q}}}{\partial \mathbf{u}} \circ \dfrac{\partial \mathbf{u}}{\partial a_i}\right] \cdot \mathbf{\Pi} + \mathbf{f}_{,\mathbf{q}} \cdot \mathbf{\Pi}_{,a_i} + \\ \qquad \left[\dfrac{\partial \mathbf{f}_{,\mathbf{u}}}{\partial \mathbf{q}} \circ \dfrac{\partial \mathbf{q}}{\partial a_i} + \dfrac{\partial \mathbf{f}_{,\mathbf{u}}}{\partial \mathbf{u}} \circ \dfrac{\partial \mathbf{u}}{\partial a_i}\right] \cdot \mathbf{\Theta} + \mathbf{f}_{,\mathbf{u}} \cdot \mathbf{\Theta}_{,a_i} + \\ \qquad \left[\dfrac{\partial \mathbf{f}_{,\mathbf{p}}}{\partial \mathbf{q}} \circ \dfrac{\partial \mathbf{q}}{\partial a_i} + \dfrac{\partial \mathbf{f}_{,\mathbf{p}}}{\partial \mathbf{u}} \circ \dfrac{\partial \mathbf{u}}{\partial a_i}\right] \\ \dot{\mathbf{\Pi}}_{\boldsymbol{\xi},a_i} = \left[\dfrac{\partial \mathbf{g}_{,\boldsymbol{\xi}}}{\partial \boldsymbol{\xi}} \circ \dfrac{\partial \boldsymbol{\xi}}{\partial a_i} + \dfrac{\partial \mathbf{g}_{,\boldsymbol{\xi}}}{\partial \mathbf{q}} \circ \dfrac{\partial \mathbf{q}}{\partial a_i} + \dfrac{\partial \mathbf{g}_{,\boldsymbol{\xi}}}{\partial a_i}\right] \cdot \mathbf{\Pi}_{\boldsymbol{\xi}} + \mathbf{g}_{,\boldsymbol{\xi}} \cdot \mathbf{\Pi}_{\boldsymbol{\xi},a_i} + \\ \qquad \left[\dfrac{\partial \mathbf{g}_{,\mathbf{q}}}{\partial \boldsymbol{\xi}} \circ \dfrac{\partial \boldsymbol{\xi}}{\partial a_i} + \dfrac{\partial \mathbf{g}_{,\mathbf{q}}}{\partial \mathbf{q}} \circ \dfrac{\partial \mathbf{q}}{\partial a_i} + \dfrac{\partial \mathbf{g}_{,\mathbf{q}}}{\partial a_i}\right] \cdot \mathbf{\Pi} + \mathbf{g}_{,\mathbf{q}} \cdot \mathbf{\Pi}_{,a_i} \\ \mathbf{\Theta}_{,a_i} = \left[\dfrac{\partial \mathbf{h}_{,\boldsymbol{\xi}}}{\partial \boldsymbol{\xi}} \circ \dfrac{\partial \boldsymbol{\xi}}{\partial a_i} + \dfrac{\partial \mathbf{h}_{,\boldsymbol{\xi}}}{\partial \mathbf{q}} \circ \dfrac{\partial \mathbf{q}}{\partial a_i} + \dfrac{\partial \mathbf{h}_{,\boldsymbol{\xi}}}{\partial a_i}\right] \cdot \mathbf{\Pi}_{\boldsymbol{\xi}} + \mathbf{h}_{,\boldsymbol{\xi}} \cdot \mathbf{\Pi}_{\boldsymbol{\xi},a_i} + \\ \qquad \left[\dfrac{\partial \mathbf{h}_{,\mathbf{q}}}{\partial \boldsymbol{\xi}} \circ \dfrac{\partial \boldsymbol{\xi}}{\partial a_i} + \dfrac{\partial \mathbf{h}_{,\mathbf{q}}}{\partial \mathbf{q}} \circ \dfrac{\partial \mathbf{q}}{\partial a_i} + \dfrac{\partial \mathbf{h}_{,\mathbf{q}}}{\partial a_i}\right] \cdot \mathbf{\Pi} + \mathbf{h}_{,\mathbf{q}} \cdot \mathbf{\Pi}_{,a_i} \end{cases}, \tag{4.46}$$

with the hypothesis that some partial derivatives are null, namely

- $\frac{\partial \mathbf{a}}{\partial \mathbf{p}} = \mathbf{0}$, the trajectory vector does not depend on the real system parameters,
- $\frac{\partial \mathbf{a}}{\partial \mathbf{p}_c} = \mathbf{0}$, the trajectory vector does not depend on the control system parameters (the measured or estimated ones),
- $\frac{\partial \mathbf{a}}{\partial t} = \mathbf{0}$, the trajectory vector is constant over time (as opposed to the desired position to be tracked $r_d(t) = \gamma(\mathbf{a}, t)$ for example),
- $\frac{\partial \mathbf{p}}{\partial t} = \mathbf{0}$, the real system parameters do not vary over time, and
- $\frac{\partial \mathbf{p}_c}{\partial \mathbf{p}} = \mathbf{0}$, the control parameters are constant w.r.t. variations in the true parameters.

Note that concerning the last one, it may not be true in practice if a parameter estimation is implemented on the system, in which case the internal states would include some parameters. In that case the gradient of the control parameters w.r.t. the real parameters should be a decreasing quantity over time, depending on the estimator performance. We do not consider such possible estimation scheme in this Thesis and thus keep the relation $\frac{\partial \mathbf{p}_c}{\partial \mathbf{p}} = \mathbf{0}$.

In these equations, several tensors intervene such as $\partial \mathbf{f}_{,\mathbf{q}} / \partial \mathbf{q} \in \mathbb{R}^{n_q \times n_q \times n_q}$. As a consequence the product $\frac{\partial \mathbf{f}_{,\mathbf{q}}}{\partial \mathbf{q}} \circ \frac{\partial \mathbf{q}}{\partial a_i}$, as well as other similar ones, are tensor-vector products. Let $\mathbf{T} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ be a tensor and $\mathbf{x} \in \mathbb{R}^{n_3}$ be a vector, we define this

tensor-vector product rule as

$$(\mathbf{T} \circ \mathbf{x})_{i,j} = \sum_{k=1}^{n_3} t_{i,j,k}\, x_k \;\; , \quad \forall i \leq n_1, j \leq n_2. \tag{4.47}$$

Moreover, we define the following quantities

$$\boldsymbol{\Gamma}_i(t) = \left. \frac{\partial \mathbf{q}(t)}{\partial a_i} \right|_{\mathbf{p}=\mathbf{p}_c} \in \mathbb{R}^{n_q} \tag{4.48}$$

$$\boldsymbol{\Gamma}_{\boldsymbol{\xi}_i}(t) = \left. \frac{\partial \boldsymbol{\xi}(t)}{\partial a_i} \right|_{\mathbf{p}=\mathbf{p}_c} \in \mathbb{R}^{n_\xi} \tag{4.49}$$

as the gradients of the system and control states w.r.t. changes in the optimization variable a_i. These quantities are also needed for evaluating the tensor $\boldsymbol{\Pi}_{,\mathbf{a}}$. As it was the case for $\boldsymbol{\Pi}$ and the other quantities introduced so far, both $\boldsymbol{\Gamma}_i$ and $\boldsymbol{\Gamma}_{\xi_i}$ do not admit, in general, an explicit expression. By adopting the same reasoning than the one leading to (4.21), one can show that the dynamics of $\boldsymbol{\Gamma}_i$ and $\boldsymbol{\Gamma}_{\xi_i}$ along the system trajectories take the expressions

$$\begin{cases} \dot{\boldsymbol{\Gamma}}_i = \dfrac{\partial \mathbf{f}}{\partial \mathbf{q}} \boldsymbol{\Gamma}_i + \dfrac{\partial \mathbf{f}}{\partial \mathbf{u}} \left(\dfrac{\partial \mathbf{h}}{\partial \mathbf{q}} \boldsymbol{\Gamma}_i + \dfrac{\partial \mathbf{h}}{\partial \boldsymbol{\xi}} \boldsymbol{\Gamma}_{\xi_i} + \dfrac{\partial \mathbf{h}}{\partial a_i} \right), \; \boldsymbol{\Gamma}_i(t_0) = \mathbf{0} \\ \dot{\boldsymbol{\Gamma}}_{\xi_i} = \dfrac{\partial \mathbf{g}}{\partial \mathbf{q}} \boldsymbol{\Gamma}_i + \dfrac{\partial \mathbf{g}}{\partial \boldsymbol{\xi}} \boldsymbol{\Gamma}_{\xi_i} + \dfrac{\partial \mathbf{g}}{\partial a_i}, \; \boldsymbol{\Gamma}_{\xi_i}(t_0) = \mathbf{0} \end{cases} , \tag{4.50}$$

which allows evaluating $\boldsymbol{\Gamma}_i(t)$ and $\boldsymbol{\Gamma}_{\xi_i}(t)$ by forward integration analogously to $\boldsymbol{\Pi}(t)$ and $\boldsymbol{\Pi}_\xi(t)$ in (4.21).

We also iterate again the notation $\mathbf{f}_{,\mathbf{u},\mathbf{q}}$ to refer to the derivative $\partial \mathbf{f}_{,\mathbf{u}} / \partial \mathbf{q}$ in the following, and we introduce the notation $\mathbf{f}_{,,\mathbf{q}}$ to refer to the derivative $\partial \mathbf{f}_{,\mathbf{q}} / \partial \mathbf{q}$. With all these settings, we can reformulate eq. (4.46) in a more compact way that only involves known

or computable quantities, *i.e.*,

$$\begin{cases} \dot{\mathbf{\Pi}}_{,a_i} = [\mathbf{f}_{,,\mathbf{q}} \circ \mathbf{\Gamma}_i + \mathbf{f}_{,\mathbf{q},\mathbf{u}} \circ \mathbf{u}_{,a_i}] \cdot \mathbf{\Pi} + \mathbf{f}_{,\mathbf{q}} \cdot \mathbf{\Pi}_{,a_i} + \\ \qquad [\mathbf{f}_{,\mathbf{u},\mathbf{q}} \circ \mathbf{\Gamma}_i + \mathbf{f}_{,,\mathbf{u}} \circ \mathbf{u}_{,a_i}] \cdot \mathbf{\Theta} + \mathbf{f}_{,\mathbf{u}} \cdot \mathbf{\Theta}_{,a_i} + \\ \qquad [\mathbf{f}_{,\mathbf{p},\mathbf{q}} \circ \mathbf{\Gamma}_i + \mathbf{f}_{,\mathbf{p},\mathbf{u}} \circ \mathbf{u}_{,a_i}] \\ \dot{\mathbf{\Pi}}_{\boldsymbol{\xi},a_i} = \left[\mathbf{g}_{,,\boldsymbol{\xi}} \circ \mathbf{\Gamma}_{\boldsymbol{\xi}_i} + \mathbf{g}_{,\boldsymbol{\xi},\mathbf{q}} \circ \mathbf{\Gamma}_i + \mathbf{g}_{,\boldsymbol{\xi},a_i}\right] \cdot \mathbf{\Pi}_{\boldsymbol{\xi}} + \mathbf{g}_{,\boldsymbol{\xi}} \cdot \mathbf{\Pi}_{\boldsymbol{\xi},a_i} + \\ \qquad \left[\mathbf{g}_{,\mathbf{q},\boldsymbol{\xi}} \circ \mathbf{\Gamma}_{\boldsymbol{\xi}_i} + \mathbf{g}_{,,\mathbf{q}} \circ \mathbf{\Gamma}_i + \mathbf{g}_{,\mathbf{q},a_i}\right] \cdot \mathbf{\Pi} + \mathbf{g}_{,\mathbf{q}} \cdot \mathbf{\Pi}_{,a_i} \\ \mathbf{\Theta}_{,a_i} = \left[\mathbf{h}_{,,\boldsymbol{\xi}} \circ \mathbf{\Gamma}_{\boldsymbol{\xi}_i} + \mathbf{h}_{,\boldsymbol{\xi},\mathbf{q}} \circ \mathbf{\Gamma}_i + \mathbf{h}_{,\boldsymbol{\xi},a_i}\right] \cdot \mathbf{\Pi}_{\boldsymbol{\xi}} + \mathbf{h}_{,\boldsymbol{\xi}} \cdot \mathbf{\Pi}_{\boldsymbol{\xi},a_i} + \\ \qquad \left[\mathbf{h}_{,\mathbf{q},\boldsymbol{\xi}} \circ \mathbf{\Gamma}_{\boldsymbol{\xi}_i} + \mathbf{h}_{,,\mathbf{q}} \circ \mathbf{\Gamma}_i + \mathbf{h}_{,\mathbf{q},a_i}\right] \cdot \mathbf{\Pi} + \mathbf{h}_{,\mathbf{q}} \cdot \mathbf{\Pi}_{,a_i} \end{cases} \tag{4.51}$$

with the initial conditions $\mathbf{\Pi}_{,a_i}(t_0) = \mathbf{0}$ and $\mathbf{\Pi}_{\boldsymbol{\xi},a_i}(t_0) = \mathbf{0}$.

Summarizing, in order to optimize some function of the state sensitivity matrix $\mathbf{\Pi}$, it is possible to benefit from an expression of its gradient $\mathbf{\Pi}_{,a_i}$ w.r.t. the optimization variables a_i. This gradient can be obtained by forward integrating (4.51) together with system (4.50) for obtaining $\mathbf{\Gamma}_i$ and $\mathbf{\Gamma}_{\boldsymbol{\xi}_i}$, and system (4.21) for obtaining $\mathbf{\Pi}$ and $\mathbf{\Pi}_{\xi}$. Note that, as explained before, one also needs to propagate the gradient of the controller state sensitivity $\mathbf{\Pi}_{\boldsymbol{\xi},a_i}$ in order to evaluate the terms $\mathbf{\Pi}_{,a_i}$ and thus the complete gradient $\mathbf{\Pi}_{,\mathbf{a}}$.

This allows us to implement a complete trajectory optimization algorithm that addresses problems (4.36) and (4.37) for any robot. In the following we will do it for the unicycle and the planar quadrotor, and discuss the obtained results.

4.4.5 Simulations

The described optimization scheme was implemented in MATLAB Simulink, where a Simulink model serves the purpose of integrating the model dynamics and sensitivity dynamics, and an 'outer loop' is implemented in Matlab that realizes the gradient descent algorithm. A set of simulations of the unicycle and quadrotor dynamics (and control) over the optimized trajectories was done. We used the robot simulator V-REP to visualize the resulting tracking of optimized and non-optimized trajectories.

The numerical errors that may occur due to numerical integration of the dynamics are measured by comparing the obtained values of the sensitivity matrix and its gradient with the corresponding finite differences computed from the variations of the state between two simulations run with infinitesimally close parameters. As a result our setup

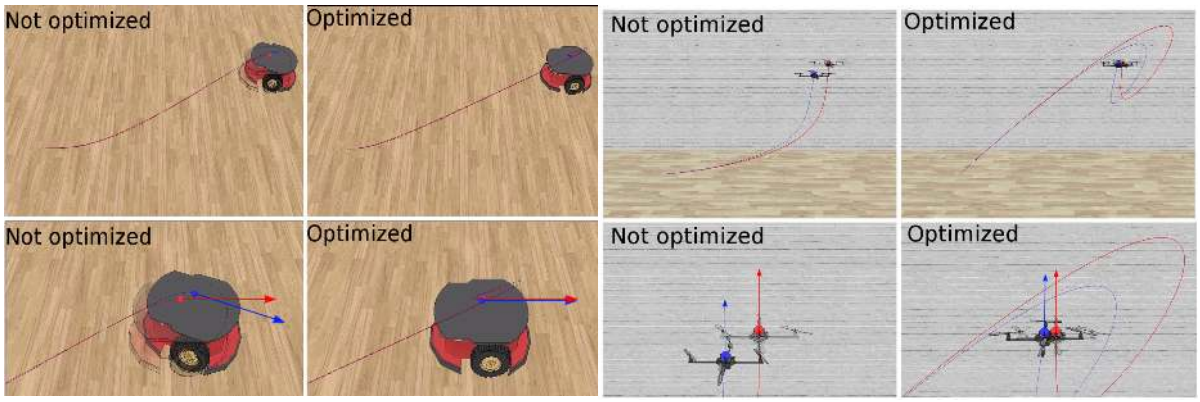

Figure 4.5 – Screenshots of the realized simulations for the unicycle robot and the quadrotor. The top line shows the trajectories that were submitted (red) and realized (blue) by the unicycle and the quadrotor with perturbed parameters. On the bottom line, one can see the detail of the end of the trajectories: the target positions (red) are reached with a better precision in the optimized cases (2nd and 4th columns) than in the initial guesses (1st and 3rd columns).

led to numerical errors smaller than $10^{-}6$ on each component of the sensitivity and its gradient.

A video was made which illustrates two simulations made for the unicycle and for the quadrotor [4]. The corresponding cases are the ones of Fig. 4.5, were one can see how the tracking of optimized trajectories (second and fourth columns) resulted in better precision at the final time than when tracking non-optimized trajectories (first and third columns).

Moreover, we show in Fig. 4.6 the tracking performance (dash lines) of the simulated unicycle when tracking trajectories that are optimized w.r.t. the state sensitivity or not. To illustrate well the behavior of the system, we considered a control law without integral action (two cases on the left), and with integral action (two cases on the right). After generating two optimal trajectories for these conditions (black lines on second and fourth plots), we made a simulated unicycle track them with a set of seven perturbed parameter vectors submitted to the controller, ranging from 80% to 120% of their nominal values. As a comparison, the same set of unicycle simulated dynamics was run on non-optimized trajectory with the same initial and final conditions (the initial guess), which resulted in the first and third plots. The error ellipsoids were computed from the

4. video at https://proxy.ens-rennes.fr/owncloud/index.php/s/nxFbMCRzD7wCzQp

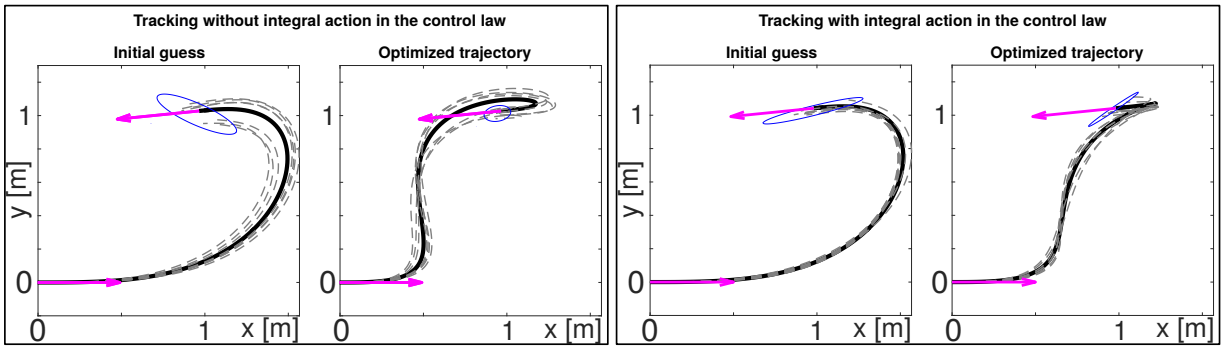

Figure 4.6 – Tracking of optimized vs. non-optimized trajectories by a unicycle with perturbed parameters. On the right, the control law includes an integral action whereas it does not on the left.

standard deviations of the obtained set of final tracking errors, and are displayed at the target final position (blue): we clearly see that the tracking of the optimized trajectories ended up in smaller errors in average than the initial guess, as wished. Note that the case with integral action in the control law seems to feature less improvement, which makes sense as the integral action already recovers part of the tracking error induced by the parametric uncertainty.

4.5 Validation through extended statistical analysis

In order to test the effectiveness of the previously described method, we conducted a statistical analysis whose goal is to check whether the application of our optimization scheme actually improves the performance of a tracking task or not, when the parameters are imprecisely known. To do so, the two system dynamics — unicycle and quadrotor — were simulated a large number of times with randomly perturbed parameters, each time following or not an optimized trajectory that had been previously generated with our method, as explained before. During this process, the evaluation of a sensitivity matrix of dimension 70 and the associated gradient of dimension 350 evaluated over 150 time samples could be computed in about 30 ms on an Intel Core i7-6600U at 2.60 GHz with 16 GB memory.

We now discuss this analysis, in which we considered four possible state sensitivities:

1. Unicycle — sensitivity of the state $\mathbf{q}$ w.r.t. the two wheel parameters $(r,\, b)$;

2. Quadrotor — sensitivity of the state $\mathbf{q}$ w.r.t. the two drag parameters $(b_x,\, b_y)$;
3. Quadrotor — sensitivity of the state $\mathbf{q}$ w.r.t. the 'mass' parameter k_f/m;
4. Quadrotor — sensitivity of the state $\mathbf{q}$ w.r.t. the 'inertia' parameter k_τ/I.

Other combinations are clearly possible, e.g., state sensitivity w.r.t. all the parameters $\mathbf{p}$ at the same time. For each of these cases we further considered four subcases, *i.e.*, DFL controller with or without integral action, which we denote as I and NI, and optimization of the 'final' problem (4.36) or of the 'integral' problem (4.37), which we denote as TF and TI. As a consequence there is a total of 16 test cases. For the ease of exposition we will then refer to an individual subcase with the code i-A-B, where $i = 1 \dots 4$ refers to the four considered state sensitivities, A∈{I, NI} to the presence or absence of the integral term in the DFL controller, and B∈{TF, TI} to the optimization problem (4.36) or (4.37). Therefore, as illustration, 3-I-TF will denote the quadrotor state sensitivity w.r.t. k_f/m evaluated for the DFL controller with integral action and optmized at t_f as described in (4.36).

Each of the sixteen combinations i-A-B was tested by running $N = 1000$ simulations in which the system under consideration, *i.e.*, unicycle or quadrotor, was either tracking a non-optimal reference trajectory $\mathbf{r}_d(t) = \gamma(\mathbf{a}_{n\_opt},\, t)$, or the optimal one $\mathbf{r}_d(t) = \gamma(\mathbf{a}_{opt},\, t)$. The non-optimized trajectory $\mathbf{a}_{n\_opt}$ was simply taken as the initial guess of the optimization algorithm (4.39), *i.e.*, $\mathbf{a}_{n\_opt} = \mathbf{M}^\dagger \mathbf{d}$, which only satisfies the linear constraints for the initial and final configuration. On the other hand, the optimized trajectory is obtained by solving the problem (4.36) or (4.37), depending on the considered subcase. In all cases, the trajectory representation function $\mathbf{r}_d(t) = \gamma(\mathbf{a},\, t)$ was chosen as a polynomial of order 15 in the variable t for each of the two components $x_d(t)$ and $y_d(t)$ of the desired position to be tracked $\mathbf{r}_d(t)$.

In each run, the model parameter(s) under consideration were generated by randomly perturbing the true system values. In particular, all nominal parameters, except the drag coefficients, were drawn from a uniform distribution with range 80% to 120% of the true system values. The drag parameters were instead drawn from a uniform distribution with range $[0\ 0.2]$.

The non-optimal reference trajectory $\mathbf{r}_d(t) = \gamma(\mathbf{a}_{n\_opt},\, t)$ was always the same across all subcases i-A-B. It was chosen as a rest-to-rest motion with the initial and final velocities, accelerations, jerks and snaps being null — the latter two only making sense for the quadrotor case —, and lasting 5 [s]. The optimized trajectory $\mathbf{r}_d(t) = \gamma(\mathbf{a}_{opt},\, t)$ was, instead, different for each subcase i-A-B because of the different conditions tested,

but it was obviously the same across the 1000 runs of each subcase. Finally, the DFL control gains were the same across all subcases, *i.e.*, one set for the I condition with $k_i > 0$, and another set for the NI condition with $k_i = 0$, and chosen so as to obtain real and negative closed-loop poles.

Consider now a particular subcase i-A-B tested over the N runs and let:

— $\mathbf{q}_{nom}^{n\_opt}(t)$ represents the closed-loop state evolution in the nominal case where $\mathbf{p}_c = \mathbf{p}$ when tracking the non-optimal reference trajectory $\mathbf{r}_d(t) = \gamma(\mathbf{a}_{n\_opt}, t)$, with $\mathbf{q}_{nom}^{n\_opt}(t)$ being the same for all the N runs;

— $\mathbf{q}_{pert,\,k}^{n\_opt}(t)$ represents the closed-loop state evolution in the k-th perturbed run where $\mathbf{p}_c \neq \mathbf{p}$, $\forall k \leq N$, when again tracking the non-optimal reference trajectory $\mathbf{r}_d(t) = \gamma(\mathbf{a}_{n\_opt}, t)$;

— $\mathbf{q}_{nom}^{opt}(t)$ represents the closed-loop state evolution in the nominal case where $\mathbf{p}_c = \mathbf{p}$, when tracking the optimal reference trajectory $\mathbf{r}_d(t) = \gamma(\mathbf{a}_{opt}, t)$, with $\mathbf{q}_{nom}^{opt}(t)$ being the same for all the N runs;

— $\mathbf{q}_{pert,\,k}^{opt}(t)$ represents the closed-loop state evolution in the k-th perturbed run where $\mathbf{p}_c \neq \mathbf{p}$, $\forall k \leq N$, when again tracking the optimal reference trajectory $\mathbf{r}_d(t) = \gamma(\mathbf{a}_{opt}, t)$.

Finally, we define the state evolution errors $\mathbf{e}_k^{n\_opt}(t) = \mathbf{q}_{nom}^{n\_opt}(t) - \mathbf{q}_{pert,\,k}^{n\_opt}(t)$ in the non-optimal case and $\mathbf{e}_k^{opt}(t) = \mathbf{q}_{nom}^{opt}(t) - \mathbf{q}_{pert,\,k}^{opt}(t)$ in the optimal case, and consider the quantities

$$\left\{\begin{array}{lcl} E_{TF,\,k}^{n\_opt} & = & \|\mathbf{e}_k^{n\_opt}(t_f)\| \\ E_{TI,\,k}^{n\_opt} & = & \int_{t_0}^{t_f} \|\mathbf{e}_k^{n\_opt}(\tau)\| \mathrm{d}\tau \\ E_{TF,\,k}^{opt} & = & \|\mathbf{e}_k^{opt}(t_f)\| \\ E_{TI,\,k}^{opt} & = & \int_{t_0}^{t_f} \|\mathbf{e}_k^{opt}(\tau)\| \mathrm{d}\tau \end{array}\right. \tag{4.52}$$

Let us focus on problem (4.36), the other one being equivalent. If tracking the optimized trajectory $\mathbf{r}_d(t) = \gamma(\mathbf{a}_{opt}, t)$ obtained from (4.36) results in a smaller value for $\|\mathbf{\Pi}(t_f)\|^2$ at the final time t_f as claimed, and thus in a smaller sensitivity norm $\|\mathbf{\Pi}(t_f)\|$, then one should expect the non-optimal state error norm $E_{TF,\,k}^{n\_opt}$ to be 'statistically larger' than the optimal $E_{TF,\,k}^{opt}$ over the N runs. In other words, the perturbed state evolution should consistently deviate less at t_f from the nominal one when following the optimal reference trajectory $\mathbf{r}_d(t) = \gamma(\mathbf{a}_{opt}, t)$. Analogous considerations clearly hold for problem (4.37) and the quantities $E_{TI,\,k}^{n\_opt}$, $E_{TI,\,k}^{opt}$ as well.

Figs. 4.7–4.10 illustrate the results of this statistical analysis for all the considered

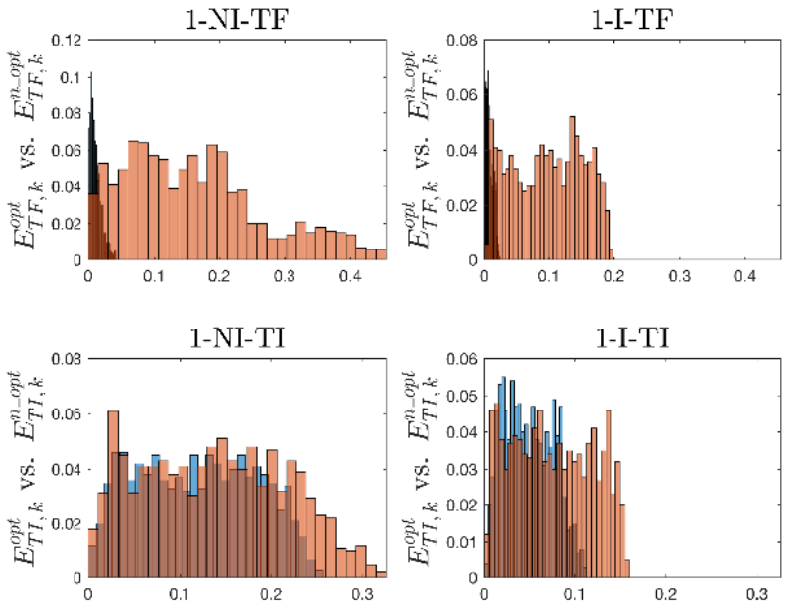

Figure 4.7 – Case 1: unicycle – sensitivity of $\mathbf{q}$ w.r.t. $(r,\, b)$. Top row: $E^{opt}_{TF,\,k}$ (blue histogram) vs. $E^{n\_opt}_{TF,\,k}$ (orange histogram) for the cases 1-NI-TF (left) and 1-I-TF (right). Bottom row: $E^{opt}_{TI,\,k}$ (blue histogram) vs. $E^{n\_opt}_{TI,\,k}$ (orange histogram) for the cases 1-NI-TI (left) and 1-I-TI (right).

subcases across the N runs. In particular, Fig. 4.7 reports the normalized histograms of $E^{opt}_{TF,\,k}$ in blue vs. $E^{n\_opt}_{TF,\,k}$ in orange, for the cases 1-NI-TF (top left) and 1-I-TF (top right), and of $E^{opt}_{TI,\,k}$ in blue vs. $E^{n\_opt}_{TI,\,k}$ in orange, for the cases 1-NI-TI (bottom left) and 1-I-TI (bottom right). The following Figs. 4.8–4.10 follow exactly the same pattern for the remaining cases 2, 3, 4. These histograms are normalized so that the height of each bin represents the probability of having a tracking error norm that falls within the bin bounds. As a consequence, these histograms can be seen as an approximation of the probability distribution of the tracking error norms resulting from the parameters being drawn from a uniform distribution as described before. Furthermore, Table 4.1 reports for all tested conditions the mean and standard deviation $(\mu^{opt},\, \sigma^{opt})$ and $(\mu^{n\_opt},\, \sigma^{n\_opt})$ of the various histograms shown in Figs. 4.7–4.10, together with the relative improvements in the optimal vs. non-optimal cases.

We can then note the following facts: the tracking state error norms in the optimal cases always resulted in smaller values w.r.t. the non-optimal cases, both in terms of mean and of variance, in all the tested conditions. Therefore, the proposed optimization

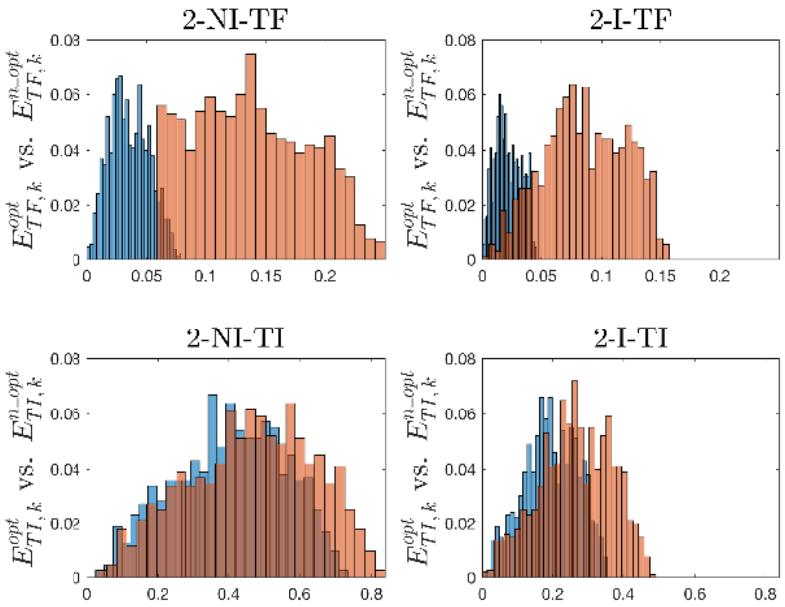

Figure 4.8 – Case 2: quadrotor – sensitivity of $\mathbf{q}$ w.r.t. (b_x, b_y). Top row: $E^{opt}_{TF,k}$ (blue histogram) vs. $E^{n\_opt}_{TF,k}$ (orange histogram) for the cases 2-NI-TF (left) and 2-I-TF (right). Bottom row: $Ei^{opt}_{TI,k}$ (blue histogram) vs. $E^{n\_opt}_{TI,k}$ (orange histogram) for the cases 2-NI-TI (left) and 2-I-TI (right).

of the reference trajectory $\mathbf{r}_d(t) = \gamma(\mathbf{a}, t)$ was able to reduce the average tracking error at t_f and over the whole trajectory — depending on the considered cases. It also made the tracking error more predictable by reducing its variance over the parameters control discrepancies. Note that this is true not only for all conditions NI, *i.e.*, without integral term in the controller, as one could have expected, but also for all conditions I, *i.e.*, with the integral term. Hence, despite the beneficial action of an integral term in compensating for parametric uncertainties, the proposed optimization is still able to further improve the overall tracking performance. Furthermore, one can also note how the improvements in the tracking error performance, both mean and variance, are always larger in the TF conditions than in the TI conditions. This can be explained as follows: in the TF conditions, the optimization has the possibility to generate a suitable 'maneuver' for eventually recovering the tracking error norm at the final time t_f, while possibly accepting an increased tracking error before t_f. On the other hand, the TI conditions weight the tracking error norm over the whole trajectory, thus leaving less room for the optimization to improve the tracking performance, e.g., contrarily to the TF

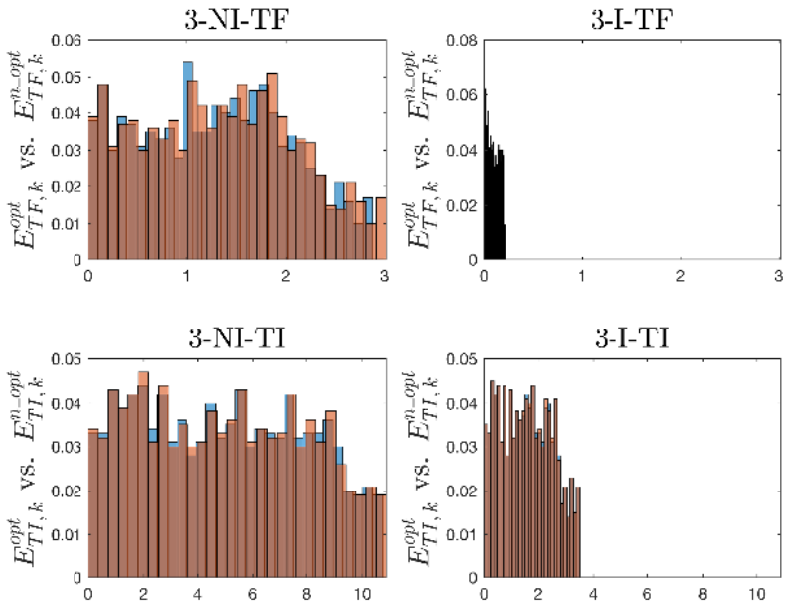

Figure 4.9 – Case 3: quadrotor – sensitivity of $\mathbf{q}$ w.r.t. k_f/m. Top row: $E^{opt}_{TF,k}$ (blue histogram) vs. $E^{n\_opt}_{TF,k}$ (orange histogram) for the cases 3-NI-TF (left) and 3-I-TF (right). Bottom row: $E^{opt}_{TI,k}$ (blue histogram) vs. $E^{n\_opt}_{TI,k}$ (orange histogram) for the cases 3-NI-TI (left) and 3-I-TI (right).

cases, a maneuver that temporarily increases the tracking error would result in a poor final performance.

Coming to the individual cases, we can note that in cases 1, 2 and 4 the sensitivity optimization is quite consistently able to produce a significant improvement in the tracking error norm performance in all conditions with higher or lower improvements depending on the specific cases as discussed. The same is not true, however, for case 3: here, only the I-TF condition resulted in a significant improvement of the tracking error norm, *i.e.*, 119% in mean and 87% in variance, while the other conditions had negligible improvements. This result can be explained by considering that case 3 involved the quadrotor state sensitivity w.r.t. the 'mass' parameter k_f/m, and variations in this parameter directly affect the possibility for compensating for the gravitational acceleration $[0 \ g]^T$ in (4.30) which is a constant drift term. In the NI conditions the DFL controller cannot compensate for $[0 \ g]^T$ whatever the shape of the reference trajectory. On the other hand, in the I-TF condition the optimization has the possibility to produce a 'maneuver' that suitably slows down the quadrotor before reaching the final pose at t_f.

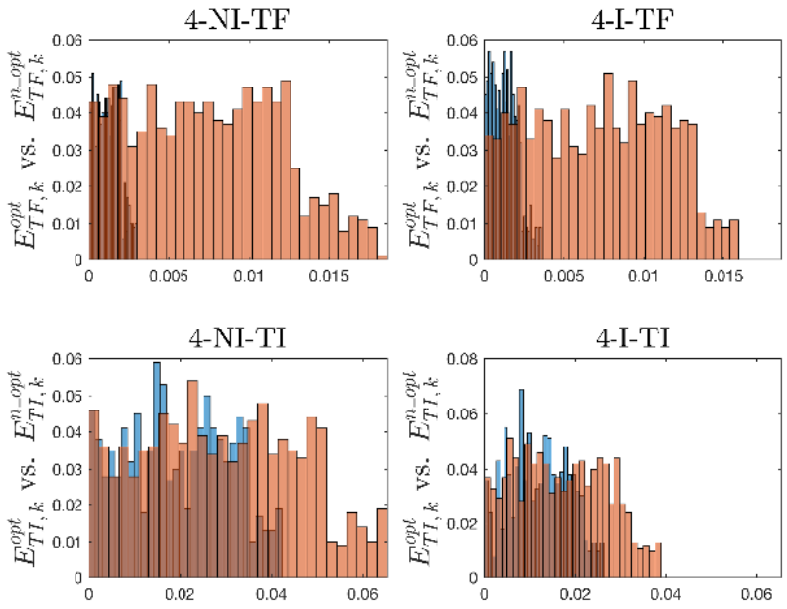

Figure 4.10 – Case 4: quadrotor – sensitivity of $\mathbf{q}$ w.r.t. k_τ/I. Top row: $E^{opt}_{TF,\,k}$ (blue histogram) vs. $E^{n\_opt}_{TF,\,k}$ (orange histogram) for the cases 4-NI-TF (left) and 4-I-TF (right). Bottom row: $E^{opt}_{TI,\,k}$ (blue histogram) vs. $E^{n\_opt}_{TI,\,k}$ (orange histogram) for the cases 4-NI-TI (left) and 4-I-TI (right).

Such a maneuver grants enough time to the integral term for compensating from the wrong k_f/m and, thus, allows to subsequently reach the correct pose at t_f. Indeed, note that in case 4, *i.e.*, sensitivity w.r.t. the 'inertia' parameter k_τ/I, the optimization could significantly improve the performance in all conditions since, in this case, no drift term is present in the states that is directly affected by k_τ/I.

4.6 Conclusion

We believe that the reported results provide a solid and successful validation of the proposed closed-loop sensitivity minimization for the sake of making a given system with its controller as insensitive as possible to parametric uncertainties. The most obvious improvement that can be foreseen from there is to consider optimization problems more complex than (4.36–4.37) by, e.g., taking also into account limited actuation or other concurrent objectives, e.g., minimize energy or time.

Case 1	μ^{opt}	μ^{n_opt}	%	σ^{opt}	σ^{n_opt}	%
1-NI-TF	0.011	0.162	1373.8%	0.008	0.1065	1237.3%
1-I-TF	0.0077	0.097	1150.2%	0.0053	0.053	910%
1-NI-TI	0.119	0.138	16%	0.064	0.078	21.5%
1-I-TI	0.049	0.075	54.1%	0.026	0.044	66.1%

Case 2	μ^{opt}	μ^{n_opt}	%	σ^{opt}	σ^{n_opt}	%
2-NI-TF	0.035	0.138	290.6%	0.016	0.048	192.4%
2-I-TF	0.022	0.086	292%	0.011	0.034	209.4%
2-NI-TI	0.40	0.462	15.2%	0.155	0.176	13.6%
2-I-TI	0.192	0.264	37.2%	0.0759	0.102	34%

Case 3	μ^{opt}	μ^{n_opt}	%	σ^{opt}	σ^{n_opt}	%
3-NI-TF	1.286	1.332	3.4%	0.752	0.781	3.7%
3-I-TF	0.046	0.103	119.5%	0.032	0.06	87.9%
3-NI-TI	4.96	5	0.91%	2.97	3	0.98%
3-I-TI	1.572	1.575	0.18%	0.941	0.942	0.17%

Case 4	μ^{opt}	μ^{n_opt}	%	σ^{opt}	σ^{n_opt}	%
4-NI-TF	0.001	0.007	523.8%	0.0007	0.0045	515.8%
4-I-TF	0.0013	0.007	449.3%	0.0008	0.004	396%
4-NI-TI	0.0193	0.029	49.8%	0.0113	0.017	49.6%
4-I-TI	0.012	0.017	41%	0.0065	0.001	53%

Table 4.1 – Mean/standard deviations (μ^{opt}, σ^{opt}) and (μ^{n_opt}, σ^{n_opt}) of the various histograms shown in Figs. 4.7–4.10 together with the relative improvements in the optimal vs. non-optimal cases.

CHAPTER 5

IMPROVEMENTS AND GENERALIZATION OF THE SENSITIVITY MINIMIZATION FRAMEWORK

5.1 Introduction

In this chapter, we propose a more general formalization of the sensitivity framework introduced before. Indeed, we have successfully applied the theory to a statistical analysis campaign with satisfying results, showing that the use of the closed-loop state sensitivity as metric to minimize the tracking error due to imperfections in the measured or estimated control parameters is a good choice. Despite the success of this first analysis, we propose to extend the theory with a rigorous approach which better justifies the choice of the sensitivity matrix. We will see that this approach allows us to extend the covered robotic applications to a new kind of situations that have not yet been taken into account, namely the cases where the controller of the system is not able to track the desired trajectory perfectly, even in the ideal case where the parameters are perfectly known, *i.e.*, $\mathbf{p}_c = \mathbf{p}$. For instance, a system with stacked servoing loops, or with any actuation limitation (thus almost all controlled systems) cannot reach the desired trajectory without a lag in general.

Hence, the minimization of the sensitivity may not be sufficient in general to overcome the tracking error, because the parametric uncertainty is not the sole responsible for it. In this respect, in this chapter we propose an extension of the algorithm which takes into account as much as possible the performance of the controller instead of considering it perfect.

As stated in Chapter 4, this approach also falls within the spirit of robust control with an improved planning strategy. However, note that robust control tends to push

the trade-off between stability and performance or precision towards stability, while the method we develop in this Thesis elaborates a layer of robustness in planning. Therefore, this allows to head the choice of a controller towards high performance, while benefiting from intrinsic robustness thanks to the properties of the feedforward term generated before.

As explained, though, the choice of the controller in our framework remains free and thus it is possible to implement robust control strategy inside the minimum-sensitivity trajectory generation.

5.2 Generalization to arbitrary outputs

As we have seen throughout this Thesis, in a realistic scenario we are not interested in controlling the state of the robot directly, but some output function of it. In the developments of Chapter 4, the output was always a subset of the state. Hence, the computation of the state sensitivity matrix was intrinsically sufficient for the aim of minimizing the tracking error. However, the relation between the output and the state may not be as simple in general. For instance an output of interest for a unicycle robot could the configuration of an on-board component, like a sensor or an actuator, which is attached to the robot with an offset in the body-frame. Such an output would typically consist in the world position and orientation of that component, as one can see from the illustration of Fig. 5.1. Defining the body-frame positioning offset vector $\delta\mathbf{q}$ and the

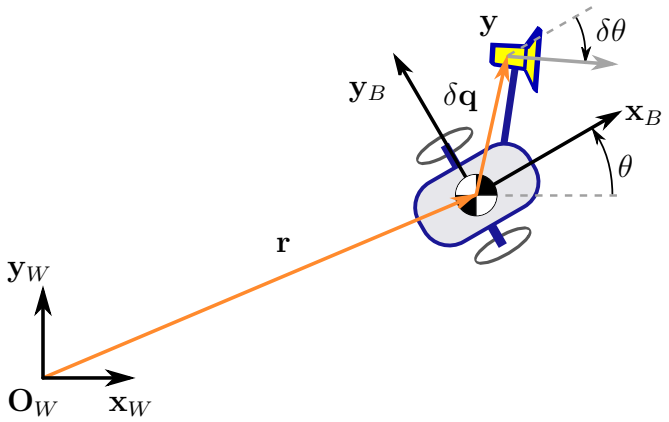

Figure 5.1 – Unicycle with a special output vector adjusted on an on-board component.

body-frame angle $\delta\theta$, the output would have the form

$$\mathbf{y}(t) = \begin{bmatrix} \begin{bmatrix} q_x \\ q_y \end{bmatrix} + \mathbf{R}(q_\theta) \cdot \delta\mathbf{q} \\ q_\theta + \delta\theta \end{bmatrix}. \tag{5.1}$$

This is of course one possibility among a lot of others. Interestingly, we see here that trigonometric functions of the state angle appear in the output expression (rotation matrix $\mathbf{R}$), which impacts the derivation of the sensitivity minimization technique as we will see next.

The second element we will focus on in this development is the possible poor tracking of the desired trajectory plugged in the controller, *i.e.*, $\mathbf{y}^*(t) = \gamma(\mathbf{a}, t)$, in particular the mismatch at the final time. Indeed, let $\mathbf{y}_f^*$ be the target final output, and let $\mathbf{y}(t_f)$ be the real output reached by the system at the final time t_f. In general, it may happen that $\mathbf{y}_f^* \neq \mathbf{y}(t_f)$ since the controller may *not* be able to track the desired trajectory $\mathbf{y}^*(t)$ because of either an inherent structural lag in some cases or because of an internal loop in other cases (or even a combination of both), but also possibly because of actuation limits. It is important to note that this new distinction was not relevant for the previous study because for the two considered systems, *i.e.*, unicycle and quadrotor, the employed *DFL controllers* with 'infinite' control authority were actually able to compensate perfectly the dynamics in order to track any possible desired trajectory (in the case where the parameters were correctly known, $\mathbf{p}_c = \mathbf{p}$). In other words, the DFL controllers do not present any tracking lag when well calibrated, *i.e.*, when $\mathbf{p}_c = \mathbf{p}$, and when not subject to input constraints, a fact that guarantees that the only source of tracking error can only come from the discrepancy in the parameters. However, in the general case the controller may not be able to track perfectly the desired trajectory because of, e.g, the use of a linear feedback on a non-linear system or, because of the presence of a low-level internal loop that servoes an underlying quantity, or any control saturation. For example, in the unicycle case that we are interested in, the robot input is usually considered to be the two spinning wheel velocities — as we have been doing throughout this Thesis — but in practice this is achieved through a low-level servoing loop that controls the electrical currents in the motors such that the desired speeds are reached as fast and precisely as possible. As a consequence, it may also happen that the final output submitted to the controller $\mathbf{y}^*(t_f)$ differs from the target output $\mathbf{y}_f^*$ which

we actually want the robot to reach. Indeed, since it is known that the controller cannot bring the real closed-loop output to this submitted final output in general, there is no reason to try to guarantee that the final submitted output reaches the target $\mathbf{y}_f^*$ at the planning stage. Instead, the planner can do whatever is considered relevant with the final submitted output in order to maximize the precision, as we will see.

Coming back to the general problem, the obvious goal we seek here as a standard robotic application is to make the real output reach the desired one, i.e., to guarantee that $\mathbf{y}(t_f) = \mathbf{y}_f^*$. As discussed in Chapter 4, Sec. 4.2, two kind of perturbations may in general lead to discrepancies between the desired output and real one apart from the controller limitations discussed before (due to lag or actuation limits). The first kind consists in all the unmodeled phenomena — it cannot be dealt with at the planning stage by definition. The second kind consists in the small but unavoidable errors in the measured parameters of the model. In fact, the model parameters are generally measured and/or estimated in a preliminary stage of the robotic application, see, e.g., [35, 68, 18].

From here, our problem of interest can be formalized as the following general optimization problem

$$\begin{aligned} \min_{\mathbf{a}} \quad & J(\mathbf{a}) \\ \text{s.t.} \quad & \mathbf{c}(\mathbf{a}) = \mathbf{0} \end{aligned} \tag{5.2}$$

where the function $\mathbf{c}(\mathbf{a})$ is a vector of constraints to be respected along the trajectory, such as actuation limits or/and initial and final values of the output/states. Following the sought goal of reaching a target output at the end of the trajectory, we define the scalar cost J as the function

$$J(\mathbf{a}) = ||\mathbf{y}_{cl}(t_f, \mathbf{p}) - \mathbf{y}_f^*||^2 \tag{5.3}$$

where the subscript cl denotes the fact that we consider, as usual, the output for the *closed-loop* system (robot+controller). Finding a trajectory $\mathbf{a}$ that minimizes this cost would then mean that the robot behaves in a way that guarantees the error to be as small as possible at the final time t_f also in presence of parametric uncertainties. Said differently, this optimization problem is the one of maximizing the precision of the robot for the task of reaching the target output $\mathbf{y}_f^*$, as wished.

Note that as in Chapter 4, one may be interested in an alternative task that consists in optimizing the performance of the robot along the whole trajectory. This could be translated into the objective of minimizing the integral of the error between the realized

output and the target one along the trajectory. However, for simplicity, we only consider the minimization problem at t_f, since the integral case can be treated in an analogous way.

Looking back at the minimization problem set in eq. (5.2), some elements can be commented. First, the real parameters $\mathbf{p}$ of the robot are obviously not known, which makes it impossible to directly solve this problem by trajectory planning. Second, however, the closed-loop output of the real system $\mathbf{y}(\mathbf{p})$ is supposed to be *close* to the one of the modeled system $\mathbf{y}(\mathbf{p}_c)$. More precisely, as soon as the output is a regular enough function of the parameters, which we assume to be the case, the small error between the real and estimated parameters $\Delta\mathbf{p} = \mathbf{p} - \mathbf{p}_c$ induces a variation of the closed-loop output which is quantifiable through the following Taylor development:

$$\mathbf{y}(\mathbf{p}_c + \Delta\mathbf{p}) = \mathbf{y}(\mathbf{p}_c) + \frac{\partial \mathbf{y}(\mathbf{p}_c)}{\partial \mathbf{p}} \cdot \Delta\mathbf{p} + \mathbf{o}(||\Delta\mathbf{p}||). \tag{5.4}$$

We recognize the term $\frac{\partial \mathbf{y}(\mathbf{p}_c)}{\partial \mathbf{p}}$ which is the sensitivity of the closed-loop output w.r.t. the parameters, evaluated at the control parameters $\mathbf{p}_c$. This *closed-loop output sensitivity* will be denoted $\mathbf{\Pi}_\mathbf{y}(\mathbf{p}_c) = \frac{\partial \mathbf{y}(\mathbf{p}_c)}{\partial \mathbf{p}}$ in the following developments.

This expression allows us to inject the closed-loop output evaluated at the known parameters $\mathbf{p}_c$ in the initial optimization problem. The cost function to be minimized can then be rewritten as

$$J(\mathbf{a}) = ||\mathbf{y}_{cl}(t_f, \mathbf{p}_c) - \mathbf{y}_f^* + \mathbf{\Pi}_\mathbf{y}(\mathbf{p}_c) \cdot \Delta\mathbf{p} + \mathbf{o}(||\Delta\mathbf{p}||)||^2. \tag{5.5}$$

In this expression, the term $\mathbf{o}(||\Delta\mathbf{p}||)$ is assumed to be small enough to be neglected in the following because of the assumed small parameter estimation error $\Delta\mathbf{p}$. In fact, for illustration, this can be checked on a simple unicycle case: let $\mathbf{e}_f = \mathbf{y}_{cl}(t_f, \mathbf{p}_c) - \mathbf{y}_f^*$ be the modeled final control error that is due to incapacity of the controller to perfectly track the desired trajectory in the general case, as discussed before. Let a unicycle with a DFL controller track a desired trajectory $\mathbf{y}^*(t)$, as done in Chapter 4. However, consider now that the input (ω_R^*, ω_L^*) of the unicycle is processed by a low-level 'hardware' control loop that manages the actual wheel spinning speeds (ω_R, ω_L) by controlling the applied torques *via* the electrical current passing through the motors. This low-level servoing loop affects the dynamics of the unicycle, by limiting the bandwidth of the actual spinning velocities — w.r.t. the ideal control input. Now let us simulate the behavior

of such a unicycle dynamics with limited bandwidth several times, across a set of parameter values ranging from 80% to 120% of the true system parameters. To do so we consider a bandwidth of 5 [rd/s] — which corresponds to the measured behavior of our real unicycle robot used for experiments. This bandwidth limitation is implemented in the simulation by inserting a first order low-pass filter of cutting frequency 5 [rd/s] in the dynamics, as a pre-processing of the two submitted wheel spinning velocities. Doing so results in the tracking performance depicted on Fig. 5.2. Note that the two parameters r and b, *i.e.*, radius of the wheels and distance between them, were changed together with the same perturbation rate.

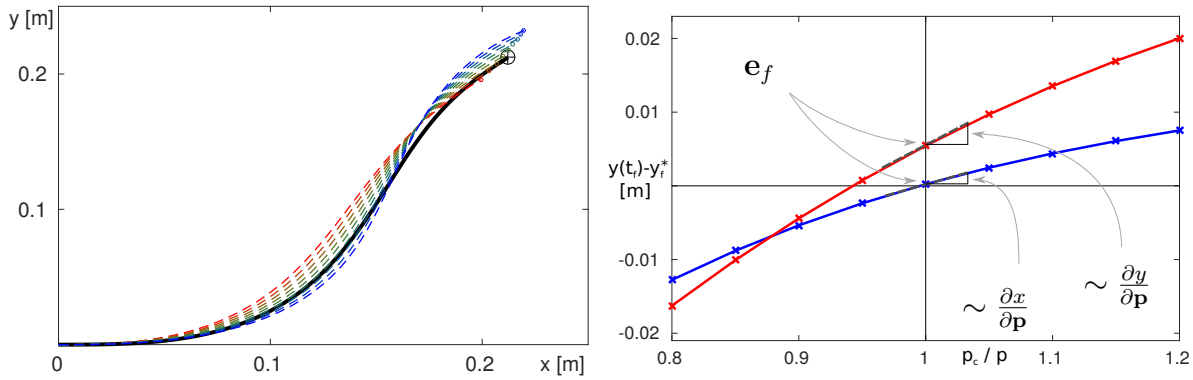

Figure 5.2 – On the left, the reference trajectory for a simulated unicycle with limited input bandwidth in black, and the resulting tracking trajectories with perturbed parameters in colors. On the right, the corresponding profiles of final tracking error components x (blue) and y (red) as functions of the parameter perturbation rate.

On this figure, one can see on the left the resulting trajectories for each parameter set in colors, and the desired trajectory fed into the DFL controller in black. We clearly see that the tracking performance is affected both by the wrong value for the control parameters, which results in the variations of the tracking error shape among the colors, and by the hardware limited input bandwidth, resulting in the lag one can observe for all the colored trajectories compared to the desired black one. The plot on the right shows the tracking error at the final time t_f, *i.e.*, $\mathbf{y}(t_f) - \mathbf{y}^*_f$, decomposed in the two spatial components x in blue and y in red. Two important facts can be observed from these plots:

1. the tracking error is not zero even when employing the right parameter values $\mathbf{p}$, because of the modeled control lag. This can be seen in the right plot by looking at the y-intercept of the tracking error values, which are non-zero;

2. the tracking error components evolve in an almost linear way around the point $\mathbf{p}_c = \mathbf{p}$.

This illustrates well the necessity of taking into account the previously defined control error $\mathbf{e}_f$ in the definition of a suitable sensitivity minimization framework. Secondly, this shows that in this case, the sensitivity gives a good hint of how the tracking error varies with the parameters, *i.e.*, in the affine way that is described by the truncated Taylor development of eq. (5.4). Indeed, the curvature of the error profile seems in this case low enough to justify the approximation of neglecting the remainder of the Taylor development in (5.4). Obviously, this should be checked with a thorough study for more complex robots as well. One way to assess whether this holds in general would be to give an explicit characterization of the second order and higher terms in the Taylor expansion for ensuring that they remain low in some range $\Delta\mathbf{p}$ of parametric variation [1]. However, we assume in this Thesis the hypothesis that the curvature remains low enough for us to ignore the effect of the higher order terms in the Taylor development, as a generalization of our observations concerning the unicycle case.

Thereafter, putting all the pieces together and by taking $\mathbf{o}(||\Delta\mathbf{p}||) \approx 0$ allows us to rewrite the cost (5.5) in a slightly approximated version

$$J(\mathbf{a}) \approx \tilde{J}(\mathbf{a}) = ||\mathbf{e}_f + \mathbf{\Pi}_\mathbf{y}(\mathbf{p}_c) \cdot \Delta\mathbf{p}||^2. \tag{5.6}$$

In this norm to be minimized, only one term that depends on the unknown real parameters $\mathbf{p}$ remains, namely the parameter error $\Delta\mathbf{p}$, while all the rest is related to known or computable quantities. Ideally, we seek a method that would be able to completely cancel the cost $\tilde{J}(\mathbf{a})$, i.e., finding a trajectory $\mathbf{a}$ such that $\tilde{J}(\mathbf{a}) = 0$. This would be done by ensuring that the final control error $\mathbf{e}_f$ exactly compensates the first order term $\mathbf{\Pi}_\mathbf{y}(\mathbf{p}_c) \cdot \Delta\mathbf{p}$ with the relation $\mathbf{e}_f = -\mathbf{\Pi}_\mathbf{y}(\mathbf{p}_c) \cdot \Delta\mathbf{p}$. Of course, as the first order term is not completely known because of the true system parameters $\mathbf{p}$ being involved, it is not possible to directly ensure this relation.

A possible workaround consists in making sure that both of these terms are close to zero. Indeed, ensuring that both $\mathbf{e}_f = \mathbf{0}$ *and* $\mathbf{\Pi}_\mathbf{y}(\mathbf{p}_c) \cdot \Delta\mathbf{p} = \mathbf{0}$ would solve the minimization problem (5.2) — it is simply a particular case of the equality $\mathbf{e}_f = -\mathbf{\Pi}_\mathbf{y}(\mathbf{p}_c) \cdot \Delta\mathbf{p}$.

1. At the moment, we consider that this is a difficult problem with the current formalization introduced in this Thesis, because of the high dimensionality of the involved quantities, and because of the complexity of the successive derivatives. As a future work, we plan to study the possibility of leveraging model simplification techniques to make these assumptions more testable, as well as to improve the computational performance of our trajectory optimization scheme in general.

The difference is that this reformulation allows us to get rid of the unknown term $\Delta\mathbf{p}$, thanks to the implication

$$[\mathbf{\Pi_y}(\mathbf{p}_c) = \mathbf{0}] \Rightarrow [\mathbf{\Pi_y}(\mathbf{p}_c) \cdot \Delta\mathbf{p} = \mathbf{0}] \tag{5.7}$$

Thus, a way to solve our general problem can be to ensure that $\mathbf{e}_f = \mathbf{0}$ *and* $\mathbf{\Pi_y}(\mathbf{p}_c) = \mathbf{0}$. Note, however, that 'around' zero, *i.e.*, when $\mathbf{\Pi_y} \neq \mathbf{0}$,

$$\arg\min_{\mathbf{a}} ||\mathbf{\Pi_y}(\mathbf{p}_c) \cdot \Delta\mathbf{p}||^2 \neq \arg\min_{\mathbf{a}} ||\mathbf{\Pi_y}(\mathbf{p}_c)||^2,$$

even though $\Delta\mathbf{p}$ does not depend on the trajectory vector $\mathbf{a}$. Indeed, the arbitrary values in $\Delta\mathbf{p}$ weight the elements of the matrix $\mathbf{\Pi_y}(\mathbf{p}_c)$ such that the cost $||\mathbf{\Pi_y}(\mathbf{p}_c)||$ does not capture the same information than (5.6). This means that in the case where the matrix $\mathbf{\Pi_y}$ cannot be completely cancelled, it is not possible to optimally solve the general problem (5.2) as wanted.

Incidentally, there are few chances in practice that a trajectory $\mathbf{a}$ guaranteeing both $\mathbf{e}_f = \mathbf{0}$ and $\mathbf{\Pi_y} = \mathbf{0}$ exists, because it is quite constraining to keep such equalities respected while at the same time complying with the inevitable actuation limits of the system. Moreover, it may not even be possible to reach these conditions for a given arbitrary dynamics with actuation saturations and for a trajectory that is limited in duration as we do consider here. However, in most robotic applications the actuation limits will still let enough room in the trajectory space for the first equality $\mathbf{e}_f = \mathbf{0}$ to be reached through an optimization of the trajectory vector $\mathbf{a}$. The other *sensitivity* term, $\mathbf{\Pi_y}(\mathbf{p}_c)$, may not be completely cancelled, but can still be reduced to the minimum possible via optimization. Though not optimal as discussed, we think that this approach is good compromise for formulating a solvable problem that is as close as possible to the desired general problem (5.2).

Summarizing, we propose a suboptimal approach to the optimization problem (5.2), which consists in solving the alternative optimization problem

$$\begin{aligned} \min_{\mathbf{a}} \quad & ||\mathbf{\Pi_y}(\mathbf{p}_c)||^2 \\ \text{s.t.} \quad & \mathbf{e}_f = \mathbf{0} \\ \text{s.t.} \quad & \mathbf{c}(\mathbf{a}) = \mathbf{0} \end{aligned} \quad , \tag{5.8}$$

where the constraint function $\mathbf{c}(\mathbf{a})$ does not include any imposed value for the submitted

final output $\mathbf{y}^*(t_f)$ to be tracked by the controller, contrarily to Chapter 4 where we had the constraint $\mathbf{y}^*(t_f) - \mathbf{y}_f^* = \mathbf{0}$. Indeed, as discussed, the task of dealing with the final output of the closed-loop system is now handled by the other constraint, *i.e.*, $\mathbf{e}_f = \mathbf{0}$.

Resulting from the foregoing construction, this alternative problem features some interesting properties:

(i) in the ideal best case where the sensitivity norm $||\mathbf{\Pi}_\mathbf{y}(\mathbf{p}_c)||^2$ can be exactly cancelled to zero, the solutions of this problem are solutions of the general problem (5.2);

(ii) as explained before, problem (5.8) is constructed such that its solutions in the other general cases, *i.e.*, when $\|\mathbf{\Pi}_\mathbf{y}(\mathbf{p}_c)\|^2 > 0$, are the closest possible to the ones of the general problem (5.2);

(iii) as in Chapter 4, addressing this problem allows to reduce at best the error due to parameter imprecision, even though the real parameters $\mathbf{p}_c$ are not known;

(iv) unlike problem (5.2), it is addressable because all the quantities appearing in it are either known or computable.

Concerning this last point, we note that the final control error $\mathbf{e}_f$ is evaluated at the control parameters $\mathbf{p}_c$. Of course, it can be computed if and only if the control loop is modeled in depth with enough precision, *i.e.*, if the effects of the potential internal control loops, and the potential closed-loop lag due to any inherent controller imperfection, are all known. The closed-loop output sensitivity $\mathbf{\Pi}_\mathbf{y}$ is also evaluated at the control parameters $\mathbf{p}_c$, which makes it computable based on the previously derived state sensitivity as we will see next.

One can notice that problem (5.8), though it may seem similar to the problem (4.36) that was constructed from intuition and asserted in Chapter 4, is meaningfully improved in terms of scope. Indeed, this problem allows to consider any output of the considered dynamical system, which makes its processing closer to the task perspective. Finally, it refines the accuracy of the theory by considering the control error $\mathbf{e}_f$ in the construction of the cost function to be minimized. In addition, this problem allows, as problem (4.36), to keep a control-aware approach that integrates at best the effect of the control loop together with the dynamics into the planning stage.

Let $\mathbb{P} \subset \mathbb{R}^{n_p}$ be a neighbourhood of $\mathbf{p}_c$ in the parameters space, that also contains the true parameters $\mathbf{p}$. Fig. 5.3 shows a formal representation of the effect of our optimization problem on the error profile. In this plot, a unidimensional case is considered,

which means that the final output tracking error $\mathbf{e}_f$ is represented as a scalar quantity in $\mathbb{R}$, as well as the true parameter $\mathbf{p}$ and the control parameter $\mathbf{p}_c$, which makes it easier to understand the concept. However, all these quantities clearly behave in the same way in higher dimensions. The abscissa of the graph corresponds to the parameter value, centred on the control parameter $\mathbf{p}_c$ which is known. The real parameter $\mathbf{p}$ is represented somewhere around $\mathbf{p}_c$, though it is not known in practice. On the ordinate we find the final output tracking values.

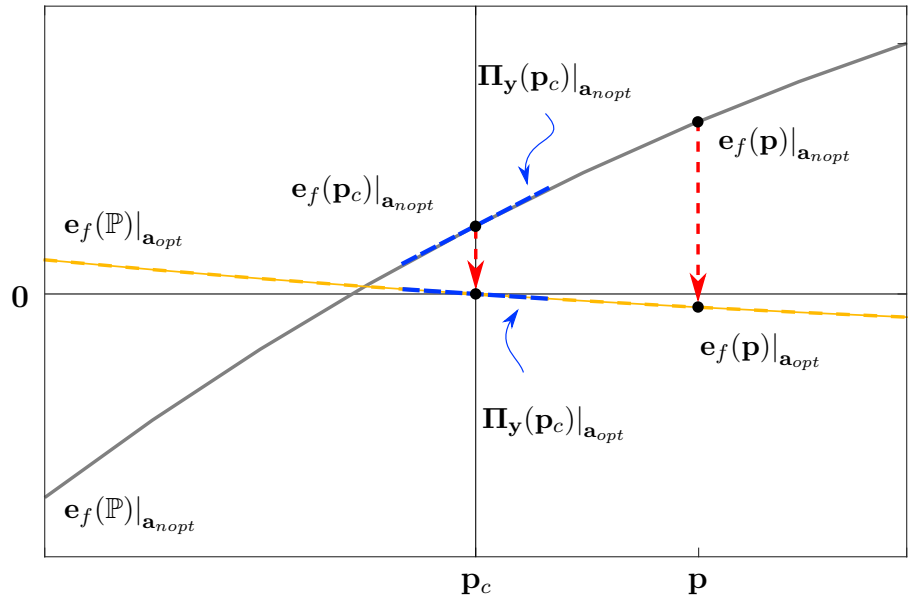

Figure 5.3 – Graphical representation of the effect of solving problem (5.8) in a unidimensional case. The two vertical red arrows show the effect of the optimization on the control error $\mathbf{e}_f(\mathbf{p}_c)$ and on the real error $\mathbf{e}_f(\mathbf{p})$. In this plot, $\mathbf{p}$ is shown indicatively but is not known in practice. Still, the resulting optimized profile guarantees that whatever the value of $\mathbf{p}$, the error $\mathbf{e}_f(\mathbf{p})|_{\mathbf{a}_{opt}}$ remains small.

One can see that the curve representing the final error $\mathbf{e}_f$ as a function of the parameters for the initial trajectory vector $\mathbf{a}_{nopt}$ showcases a certain y-intercept value, which corresponds to the control error $\mathbf{e}_f(\mathbf{p}_c)$ that can be completely predicted from the models of the system dynamics and controller. This is the quantity that is denoted by $\mathbf{e}_f$ and that we seek to cancel in the optimization problem formulation (5.8). If a solution $\mathbf{a}_{opt}$ can be found, solving this problem makes that control error null, because it is one of the constraints of the optimization. This is graphically translated into the red arrow that vertically brings the control error $\mathbf{e}_f(\mathbf{p}_c)$ to zero. Jointly, the profile of

the output tracking error is changed from $\mathbf{e}_f(\mathbb{P})|_{\mathbf{a}_{nopt}}$ in gray to $\mathbf{e}_f(\mathbb{P})|_{\mathbf{a}_{nopt}}$ in orange. This change happens because the new considered trajectory vector $\mathbf{a}_{opt}$ is different, and thus makes the tracking performance — especially at the ending time t_f of the trajectory — differently affected by a perturbation of the control parameter.

At this stage the real output error $\mathbf{e}_f(\mathbf{p})$ might already be reduced in certain cases. Nevertheless, the most important change in the error profile is the second one, *i.e.*, the reduction of the slope of the tangent at $\mathbf{p}_c$ (the dashed blue lines), which is guaranteed by the minimization of the closed-loop output sensitivity $\mathbf{\Pi}_{\mathbf{y}}(\mathbf{p}_c)$. This minimization makes the new error profile the most flat possible around $\mathbf{p}_c$, as one can see from the illustration of the orange curve. As derived theoretically just before, we see here that the combination of both 1) the minimization of the sensitivity norm, and 2) the cancellation of the control error, makes the real error $\mathbf{e}_f(\mathbf{p})|_{\mathbf{a}_{opt}}$ in the optimized case lower than the initial one $\mathbf{e}_f(\mathbf{p})|_{\mathbf{a}_{nopt}}$, and actually the lowest possible. Even better, this is all done without knowing the value of the real parameter $\mathbf{p}$, and still it holds whatever value it has, by construction of the optimization problem.

5.2.1 Solving procedure in the general case

In this section we propose a general method for solving the optimization problem (5.8) set up previously. Unlike in Sec. 4.3.2 of Chapter 4, we now make a distinction between the *submitted* target output $\mathbf{y}^*(t)$ that the controller of the robot will try to track over the time interval $\mathbb{T}$, and the *desired* final output $\mathbf{y}_f^*$ that we really want the robot to arrive at. Indeed, as discussed before, the controller imperfection might induce a small but non-negligible tracking lag in the general case. In particular, we may thus have that $\mathbf{y}^*(t_f) \neq \mathbf{y}_f^*$, *i.e.*, the trajectory that is submitted to the controller may optimized in such a way that it does not end at the target in order to compensate the effect of the lag of the controller.

Still considering the generic dynamics of eq. (4.1), let the upgraded control laws be

$$\begin{cases} \boldsymbol{\xi}(0) = \boldsymbol{\xi}_0 \\ \dot{\boldsymbol{\xi}}(t) = \mathbf{g}(\mathbf{q}(t), \boldsymbol{\xi}(t), \mathbf{a}, t, \mathbf{p}_c) \\ \mathbf{u}(t) = \mathbf{h}(\mathbf{q}(t), \boldsymbol{\xi}(t), \mathbf{a}, t, \mathbf{p}_c) \\ \mathbf{y}(t) = \mathbf{z}(\mathbf{q}(t), \mathbf{p}_c) \end{cases} \tag{5.9}$$

with the output being a function of the control parameters $\mathbf{p}_c$ rather than the real pa-

rameters $\mathbf{p}$, in order to take into account the fact that the output is usually estimated over time by leveraging some model of the system, which can thus only be function of the control parameters $\mathbf{p}_c$. Note that here, the trajectory $\mathbf{a}$ determines the target *output* $\mathbf{y}^*(t)$ through the representation function $\mathbf{y}^*(t) = \gamma(\mathbf{a}, t)$, instead of some desired state or part of it as it was the case in eq. (4.14). Clearly, it would be the same if $\mathbf{y}$ was actually a subset of the state $\mathbf{q}$, but in the general case it may be more complex as mentioned in Sec. 5.2.

As seen before, we seek at minimizing the cost function of problem (5.8), defined as some norm of the closed-loop output sensitivity, *i.e.*, $||\mathbf{\Pi}_\mathbf{y}(\mathbf{p}_c)||^2$. To do so, we first need to compute this quantity, which will be detailed now. Furthermore, the gradient of this cost w.r.t. the optimization vector $\mathbf{a}$ is of paramount importance in the optimization process and will thus be computed afterwards. Note that the choice of the norm does not affect the way the sensitivity is computed, though it does intervene in the gradient derivation. As in Chapter 4, we use the Frobenius matrix norm, which can basically be described as the square root of the sum of the squared components of the matrix, by extension of the Euclidean norm on the matrix space $\mathbb{R}^{n_y \times n_p}$.

The closed-loop output sensitivity can be derived from eq. (5.9) by simply taking the derivative of the output function w.r.t. the parameters which yields

$$\begin{aligned} \mathbf{\Pi}_\mathbf{y}(\mathbf{p}_c) &= \frac{\partial \mathbf{y}}{\partial \mathbf{p}}(\mathbf{p}_c) = \frac{\partial \mathbf{z}}{\partial \mathbf{q}}(\mathbf{p}_c) \cdot \frac{\partial \mathbf{q}}{\partial \mathbf{p}}(\mathbf{p}_c) \\ &= \mathbf{z}_{,\mathbf{q}}(\mathbf{p}_c) \cdot \mathbf{\Pi}(\mathbf{p}_c) \end{aligned} \quad . \tag{5.10}$$

In this condensed form that makes use of the notation $\mathbf{z}_{,\mathbf{q}} = \frac{\partial \mathbf{z}}{\partial \mathbf{q}}$ again for the differentiation, we clearly see the closed-loop state sensitivity $\Pi(\mathbf{p}_c)$ evaluated at the control parameters $\mathbf{p}_c$ emerging. Note that the time variable t does not appear in the arguments of the output function $\mathbf{z}$, and so the Jacobian $\mathbf{z}_{,\mathbf{q}}$ is not a function of the time directly, though it is a function of the state $\mathbf{q}$ which may vary with t. Also, note that the output function $\mathbf{z}$ do not depend on the true parameters $\mathbf{p}$ but on the control parameters $\mathbf{p}_c$, which is why there is no term $\mathbf{z}_{,\mathbf{p}}$ in this expression. Indeed, as before we have $\frac{\partial \mathbf{p}_c}{\partial \mathbf{p}} = \mathbf{0}$.

As a consequence, the evaluation of the closed-loop output sensitivity can be done by leveraging the differential system (4.21) that has been set up in Chapter 4. As seen before, solving this system enables us to numerically compute the values of the components of the state sensitivity $\mathbf{\Pi}(t)$, which can then be injected in eq. (5.10) to get the

desired output sensitivity $\mathbf{\Pi}_{\mathbf{y}}(t)$.

If we go back to the simple unicycle case where the output $\mathbf{y}$ just consists in the position of the robot $(q_x,\, q_y)$, we get the simple Jacobian matrix

$$\mathbf{z}_{,\mathbf{q}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{5.11}$$

which is a constant matrix. It then affects very slightly the computations as they were done before. Note also that pre-multiplying the state sensitivity $\mathbf{\Pi}$ by this matrix is roughly equivalent to selecting only certain elements of the sensitivity matrix in the norm to be minimized, as we did in our statistical analysis of Sec. 4.5.

As a comparison, when considering the alternative three-dimensional output example of eq. (5.1), we get a richer expression for this Jacobian. First note that in this case the body-frame offset position and orientation of the on-board component, are geometrical properties of the system that need to be measured or estimated in practice, and thus they are parameters of our model. We then have to extend the parameter vector from $(r,\, b)$ to $(r,\, b,\, \delta q_x,\, \delta q_y,\, \delta\theta)$. As a consequence, our Jacobian of interest writes

$$\mathbf{z}_{,\mathbf{q}} = \begin{bmatrix} 1 & 0 & -\sin(q_\theta)\delta q_{xc} - \cos(q_\theta)\delta q_{yc} \\ 0 & 1 & \cos(q_\theta)\delta q_{xc} - \sin(q_\theta)\delta q_{yc} \\ 0 & 0 & 1 \end{bmatrix} . \tag{5.12}$$

In this case, there is a dependency of the Jacobian to the state $\mathbf{q}$ and to the parameters $\mathbf{p}_c$. Note however that, as explained, the parameters that intervene here are the control parameters and not the real parameters $\mathbf{p}$.

The second important change that occurs when switching from the first formulation of Chapter 4 to the new one of problem (5.8), is the gradient derivation. Indeed, this optimization problem under non-linear constraints requires us to provide both the gradient of the cost to be minimized, and of the non-linear constraint $\mathbf{e}_f = \mathbf{0}$. This is of course possible, because we have constructed this problem while carefully checking the feasibility of all the involved computations, see Sec. 5.2.

Let $\mathbf{\Pi}_{\mathbf{y},\mathbf{a}} = \frac{\partial \mathbf{\Pi}_{\mathbf{y}}}{\partial \mathbf{a}} \in \mathbb{R}^{n_y \times n_p \times n_a}$ be the tensor gradient of the output sensitivity w.r.t. the

trajectory vector $\mathbf{a}$. The gradient of the cost function w.r.t. $\mathbf{a}$ then writes

$$\frac{\partial \|\mathbf{\Pi_y}\|^2}{\partial \mathbf{a}} = 2\mathbf{\Pi_y} \cdot \mathbf{\Pi_{y,a}}. \tag{5.13}$$

which is naturally similar to the expression we had for the state sensitivity. However, the complete derivation of the expression for the gradient $\mathbf{\Pi_{y,a}}$ is a bit more complicated. In detail, we have that

$$\begin{aligned}\mathbf{\Pi_{y,a}} =& \frac{\partial \mathbf{z_{,q}}}{\partial \mathbf{a}} \circ \frac{\partial \mathbf{q}}{\partial \mathbf{p}} + \mathbf{z_{,q}} \circ \frac{\partial \mathbf{\Pi}}{\partial \mathbf{a}} \\ =& \left(\frac{\partial \mathbf{z_{,q}}}{\partial \mathbf{q}} \circ \frac{\partial \mathbf{q}}{\partial \mathbf{a}} \right) \circ \mathbf{\Pi} + \mathbf{z_{,q}} \circ \mathbf{\Pi_{,a}} \\ =& \mathbf{z_{,,q}} \circ \mathbf{\Gamma} \circ \mathbf{\Pi} + \mathbf{z_{,q}} \circ \mathbf{\Pi_{,a}}\end{aligned} \tag{5.14}$$

where we recall that the matrix quantity $\mathbf{\Gamma} = \frac{\partial \mathbf{q}}{\partial \mathbf{a}}$ can be computed by integrating the differential system (4.50). On the last line, on can see that the products between the involved elements are either tensor-matrix, or matrix-tensor products. These are such that the following matrix equality holds for each component of the trajectory vector $\mathbf{a}$

$$\mathbf{\Pi_{y},}_{a_i} = (\mathbf{z_{,,q}} \circ \mathbf{\Gamma}_i) \cdot \mathbf{\Pi} + \mathbf{z_{,q}} \cdot \mathbf{\Pi_{,}}_{a_i} \ , \ \forall i \leq n_a \tag{5.15}$$

where the symbol $\circ$ denotes the same tensor-vector product as defined and used lately in this Thesis. As wished, all the quantities in eqs. (5.10–5.15) are known or computable, and thus the cost minimization (without the non-linear constraint) is implementable.

Concerning the non-linear constraint $\mathbf{e}_f = \mathbf{0}$, note that the scalar formulation $\|\mathbf{e}_f\|^2 = 0$ is equivalent, thanks to the separation property of the Euclidean vector norm over $\mathbb{R}^{n_y}$. Adopting this formulation allows to reduce the constraint to a unidimensional space, which makes more sense w.r.t. the task we actually seek, *i.e.*, cancelling the distance between the final output $\mathbf{y}(t_f)$ and the target $\mathbf{y}_f^*$. In the same way that the gradient of the cost function was required, it is also the case for the gradient of the constraint function. Indeed, the solver, e.g., an interior-point algorithm, will use this gradient to guarantee that the solution respects the constraint. Recalling that the control error equals

$\mathbf{e}_f = \mathbf{y}(t_f) - \mathbf{y}_f^*$, the differentiation of this norm yields the expression

$$\begin{aligned}\frac{\|\partial \mathbf{e}_f\|^2}{\partial \mathbf{a}} =& 2\mathbf{e}_f^\top \cdot \frac{\partial \mathbf{e}_f}{\partial \mathbf{a}} \\ =& 2\mathbf{e}_f^\top \cdot \mathbf{z}_{,\mathbf{q}} \cdot \mathbf{\Gamma}\end{aligned} \quad . \tag{5.16}$$

Again, we see that all the terms involved in this expression are either known or computable through integration of the differential systems established so far. Thus, one can compute the value of this gradient matrix given a certain trajectory vector $\mathbf{a}$ to be tracked by the system.

5.2.2 Final error compensation case study

As a benchmark of our previous theoretical derivations, we applied the sole control error compensation idea to a simple case, when there is no parameter perturbation, *i.e.*, $\mathbf{p}_c = \mathbf{p}$. Here, we consider again a simple unicycle with limited input bandwidth, in order to simulate a control lag that corresponds to some hardware handling of the wheel spinning velocities, as usual in this chapter. Then, to implement the control error compensation, we set a simple optimization problem that consists in minimizing the control error $\|\mathbf{e}_f\|^2$, instead of using it as a constraint with the sensitivity norm cost. Hence, we treat the optimization problem

$$\begin{aligned}\min_{\mathbf{a}} \quad & \|\mathbf{e}_f\|^2 \\ \text{s.t.} \quad & \mathbf{c}(\mathbf{a}) = \mathbf{0}\end{aligned} \quad , \tag{5.17}$$

where the constraint function $\mathbf{c}(\mathbf{a})$ only imposes the initial configuration of the robot, without integrating the final position as discussed before.

After implementing the cost function and gradient as described in eq. (5.17) and eq. (5.16), we solve this problem by means of the wide spread solver fmincon within MATLAB. Fig. 5.4 shows the result of this optimization. The blue trajectory corresponds to the tracking of a non-optimized desired trajectory, which ends at the target point $\mathbf{y}_f^*$. We clearly see that the control lag results in the robot not being able to bring itself to the target (circled cross). In contrast, one can see that the generated trajectory (the dash line) does not end at the target $\mathbf{y}_f^*$, as expected. This allows the robot to realize the red trajectory, which is able to reach that target because the control lag has been

anticipated perfectly by the optimizer, and results in a deviation of the tracking that is completely mastered. Note that the solver was actually able to reach the minimal cost value of zero, which confirms in this case our sayings that this control error can be completely cancelled.

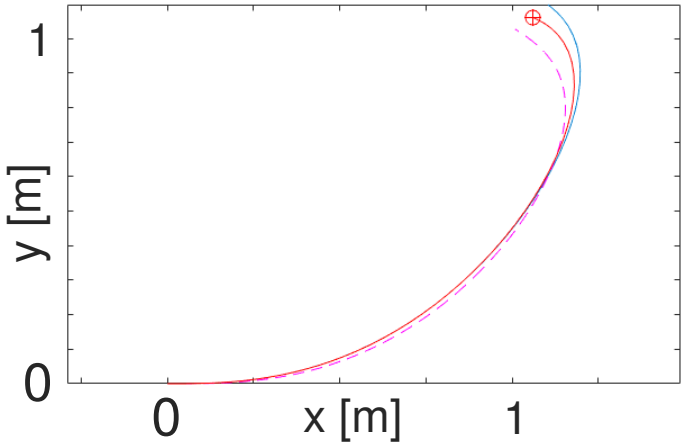

Figure 5.4 – Illustration of the concept of tracking error pre-compensation. The dash line is the trajectory that was generated thanks to the knowledge of the controller's bandwidth, which ends up in the red trajectory when submitted to the controlled unicycle. As opposed to the initial guess tracking, in blue, the final target is reached.

5.3 Other sensitivity metrics

Heretofore, we have been interested in the closed-loop output sensitivity, because this metric allows us to quantify the variations induced by the discrepancy between the control parameters $\mathbf{p}_c$ and the true system parameters $\mathbf{p}$, as demonstrated in Sec. 5.2. However, it can be noticed that in practice the realization of the tracking task with a high accuracy in the output may not be the sole objective roboticists are interested in. Indeed, among others, the guarantee that the system internal states remains controlled, as well as the repeatability of the task, are often essential for development, economic and security reasons. In practice, these considerations are completely bound to the behavior of the inputs that are submitted to the system dynamics. Therefore, we interested ourselves in the dynamics of the input $\mathbf{u}(t)$ along the trajectories that are realized in our work.

One way to control that the inputs do what they are expected to do, *i.e.*, remain in their bounds, and more generally remain predictable despite the presence of parametric uncertainty, is to look at the values of a different quantity that we call here the *closed-loop input sensitivity*

$$\mathbf{\Theta} = \frac{\partial \mathbf{u}}{\partial \mathbf{p}}.$$

Indeed, the elements of this matrix represent the amount of variations that would occur on the inputs, consequentially to a perturbation of the parameters. In other words, the closed-loop input sensitivity is able to quantify how much the inputs are affected by the discrepancy between the control parameters $\mathbf{p}_c$ and the true ones $\mathbf{p}$. It is the input counterpart of the previously derived output sensitivity $\mathbf{\Pi_y}$.

The main difference with the output sensitivity, though, is that the inputs are generally not meant to reach a certain pre-determined target. They are in contrast meant to be predictable and feasible. Hence, it does not make sense to apply the exact same reasoning that we developed in Sec. 5.2 for the construction of a resembling optimization problem that would be adapted to the input. Nevertheless, what makes sense w.r.t. the elements we just discussed in this section, is to ensure that the inputs behave in a way that is in some sense 'similar' to what is expected at the planning stage by exploiting the closed-loop input sensitivity. Making so would henceforth guarantee, or at least increase the probability, that the hardware is operated in the conditions it was designed for. Furthermore, we observed that in practice the behavior of the input when tracking an output-optimized trajectory (following the algorithm described before) is susceptible of great variations against parameter deviations. Actual measures of this effect will be shown in the analysis of the next section.

Encouraged by this reflection, we propose another use of the sensitivity framework, that is completely different in terms of goal and results. More specifically, we propose to address the matter exposed just before by setting up a new optimization problem,

$$\begin{aligned} \min_{\mathbf{a}} \quad & J_{\mathbf{\Theta}}(\mathbf{a}) \\ \text{s.t.} \quad & \mathbf{c}(\mathbf{a}) = \mathbf{0} \end{aligned} \tag{5.18}$$

that is independent from the one of eq. (5.8), and where the cost function $J_{\mathbf{\Theta}}$ is constructed with the input sensitivity. The constraint function $\mathbf{c}(\mathbf{a})$ is, however, similar to the one of Chapter 4, *i.e.*, it defines the initial and final configurations of the trajectory, and possibly the input bounds.

Considering the task of making the input globally predictable, as wished here, we construct the following cost function that we think captures well the envisioned task

$$J_{\mathbf{\Theta}}(\mathbf{a}) = \int_{t_0}^{t_f} \|\mathbf{\Theta}(\tau, \mathbf{a})\|_{\mathbf{W}}^2 \, \mathrm{d}\tau \tag{5.19}$$

where the norm $\|\cdot\|_{\mathbf{W}}$ is a special weighted norm. Given a weighting matrix $W \in \mathbb{R}^{n_u \times n_p}$ and the matrix $\mathbf{\Theta} \in \mathbb{R}^{n_u \times n_p}$ which we want to evaluate the norm of, we define this weighted norm with the rule

$$\|\mathbf{\Theta}\|_{\mathbf{W}} = \sqrt{\sum_{i,j} m_{i,j}^2 \, w_{i,j}} \tag{5.20}$$

which is the same as the one used in Chapter 2 for the open-loop state sensitivity.

This choice for the cost function allows to take into account the inputs along the whole trajectory equally in time, via the integral, and by means of the weighting matrix $\mathbf{W}$ to freely chose to put more attention on a particular input than on the others, which may be useful for complex systems where the inputs do not all have the same importance. Thanks to the relation between $\mathbf{\Theta}$ and the other sensitivities $\mathbf{\Pi}$ and $\mathbf{\Pi}_\xi$ that is given by eq. (4.21), the cost $J_{\mathbf{\Theta}}$ is completely computable by the same way than the state and output sensitivities, *i.e.*, by integrating the differential systems that we provided.

Before going further with problem (5.18), note that one can apply the cost $J_{\mathbf{\Theta}}$ to any of the trajectories that were previously studied. In particular, this enables us to evaluate the impact of changing a trajectory from an initial guess to another one that is optimized w.r.t. some criterion that is not the one of problem (5.18). Doing so on the initial trajectory and optimized one of Fig. (4.6)-left, and with a unitary weighting, *i.e.*, $w_{i,j} = 1, \forall i, j$, we obtain that respectively $J_{\mathbf{\Theta}}(\mathbf{a}_{nopt}) = 65$ and $J_{\mathbf{\Theta}}(\mathbf{a}_{opt}) = 4090$ (rad^{2}m^{-2}s^{-1}). This means that in that particular case, the optimization of the trajectory w.r.t. the state sensitivity cost function was detrimental for the closed-loop input sensitivity norm, and thus for the input predictability. Note that the fact it varies in this direction is completely incidental, because the value the input sensitivity was not controlled during that optimization process.

With this in mind, let us now derive the gradient of the cost function $J_{\mathbf{\Theta}}$. Similarly to

the previous cases, we have that

$$\begin{aligned}\frac{\partial \|\boldsymbol{\Theta}\|^2_{\mathbf{W}}}{\partial \mathbf{a}} =& \frac{\partial}{\partial \mathbf{a}} \left[\sum_{i,j} w_{i,j}\, \theta(\mathbf{a})^2_{i,j} \right] \\ =& 2 \sum_{i,j} w_{i,j}\, \theta(\mathbf{a})_{i,j}\, \frac{\partial \theta(\mathbf{a})_{i,j}}{\partial \mathbf{a}} \\ =& 2(\mathbf{W} \star \boldsymbol{\Theta}) \circ \boldsymbol{\Theta}_{,\mathbf{a}}\end{aligned} \tag{5.21}$$

where $\star$ is the Hadamard product, *i.e.*, element-wise, and where $\circ$ is the matrix-tensor product defined before. In this expression one can recognize the raw input sensitivity gradient $\boldsymbol{\Theta}_{,\mathbf{a}}$ that was already introduced and computed in eq. (4.51).

We now have all the required elements to produce an optimal trajectory in the sense of problem (5.18).

5.4 Statistical analysis

In this section we present a new statistical analysis of larger scale that aims at testing the soundness of the optimization problems (5.8) and (5.18) when applied to various trajectory global shapes. In particular we wish to verify that the minimization of the costs that were obtained from the sensitivity quantities actually result in improvements of the performances at the task level, *i.e.*, for problem (5.8) a reduction of the output tracking error in case of perturbed parameters and for problem (5.18) a global reduction of the discrepancy between the expected input (along the submitted trajectory) and the realized input (along the actual trajectory). Given the number of possible test cases, we chose to focus on the unicycle case, which already generates a lot of cases as we will see. The application of the same analysis to others robots is part of our future work, although we can notice that we do not expect significant differences in the behavior of our framework against a quadrotor for example, which features a close dynamics and control overall.

The concept of this analysis is to 1) generate a bunch of N_{traj} non-optimized trajectories and corresponding optimized trajectories on which we would like to test the framework, and 2) to evaluate the resulting performance for each trajectory case, by means of statistical analysis of the dynamical behavior against N parameter perturbations. Thus, after having selected meta parameters that we think are relevant for these

tests, we conducted a trajectory generation phase followed by an evaluation campaign of each conditions against a set of randomly perturbed parameters. In order to be general and thus more realistic in the testing conditions, we wanted to test the effectiveness of the framework against variations in the trajectory limit conditions. Changing both the initial and final position is equivalent to changing only the final position and moving the whole trajectory. Hence, only the final position is varied. Similarly, the orientation of the initial tangent (given by the initial velocity) is constantly set to zero (aligned with the $\mathbf{x}$ axis) while the final position and orientation are varied relatively, without loss of generality. As there is a spatial symmetry in the behavior of the unicycle dynamics w.r.t. the initial orientation, we also consider only final positions that are in the top half disk above the initial position. Finally, the initial guesses are also randomized to avoid, or at least to limit, any bias that would be induced by the specificities of the polynomial representation function γ that is used. This is done by adding a random vector in the nullspace of the initial and final linear constraint,

$$\mathbf{a}_{nopt} = \mathbf{M}^{\dagger} \cdot \mathbf{d} + (\mathbf{I} - \mathbf{M}^{\dagger} \cdot \mathbf{M})\mu \tag{5.22}$$

where $\mu \in \mathbb{R}^{n_a}$ is a random vector, which makes the polynomial coefficients arbitrary while still making the trajectory respect the initial and final conditions.

Note that in these simulations, the control lag was not simulated, because it would have complexified the computations a lot while not changing significantly the results. Thus, in the coming analysis, there was no final error constraint $\mathbf{e}_f$ in the simulations because as shown before, the DFL controller already guarantees that $\mathbf{e}_f(\mathbf{p}_c) = \mathbf{0}$, so there is no need to compensate.

Concretely, the first phase of generating trajectory conditions is done by picking a target $\mathbf{y}_f^* = \rho(\cos(\phi), \sin(\phi))$ with $0 \leq \phi \leq \pi$ and $\rho \leq 0.5$ [m]. The initial orientation is set to zero, while the final one is chosen randomly by picking an angle value from $-\pi$ to π.

Then in the optimization constraints $\mathbf{c}(\mathbf{a})$, we add the restriction that the speed $v(t)$ of the unicycle has to remain strictly above zero along the trajectory. This condition originates from the structure of the DFL controller, which requires in its computations to divide by the internal state ξ_v that represents the velocity norm. As a consequence we seek trajectories that won't excite this intrinsic singularity with, e.g., a sharp change of moving direction. The constraint of keeping the inputs bellow some saturations is

also added, *i.e.*, $|\omega_R(t)| \leq \omega_{max}$ and $|\omega_L(t)| \leq \omega_{max}$, because it is absolutely necessary in practice and thus makes the study closer to the reality of the application of the theory.

Then, for each initial trajectory that is generated according to these rules, we first test the output sensitivity minimization of problem (5.8). To do so, we first solve the optimization problem for these specific conditions, *i.e.*, with the initial guess and constraints that were produced as we described. Afterwards, we run N simulations of the unicycle tracking 1) the initial guess and 2) the optimized trajectory, while randomly varying each time the parameters $\mathbf{p}$ in range from 80% to 120% of their nominal values. Once these $2N$ simulations are run, we measure the corresponding final errors $\|\mathbf{e}_f\|^2$ for the initial guess and for the optimized trajectory. Then, on each of these two resulting sets of error values, we measure the mean value and the standard deviation. As a synthesis, starting from a single initial guess trajectory we end up with four numbers, namely the means and standard deviations of the final tracking errors for the randomized non-optimized trajectory and for the optimized trajectory.

Finally, we aggregate these numbers over the whole set of N_{traj} trajectories that were generated in phase 1, by computing the boxplot 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 N_{traj} error means. The same is also done for the standard deviations on the initial guesses, as well as for error means and standard deviations of the optimized trajectories.

Fig. 5.5-left shows the resulting boxplots, with the error means boxplots on the top plot, and the error standard deviations boxplots on the bottom plot. For this statistical analysis we took $N_{traj} = 100$ trajectory conditions, and $N = 100$ simulations for each case. On these graphs, it appears that the repartition of the error means is clearly reduced both in terms of height and of spreading, for the optimized trajectories compared to the initial ones. The direct conclusion we can formulate is that the errors are indeed reduced when applying our optimization process, and they are also made more predictable. This firmly demonstrates the effectiveness of the proposed method, at least in the conditions we described.

The second test we conducted consists in the evaluation of the input sensitivity. More precisely, for each of the trajectories described above, *i.e.*, optimized and non-optimized with different final conditions, we evaluated the quantity

$$E_{\mathbf{u}} = \int_{t_0}^{t_f} \|\mathbf{u}(\mathbf{p}_c) - \mathbf{u}(\mathbf{p})\|^2, \mathrm{d}\tau \tag{5.23}$$

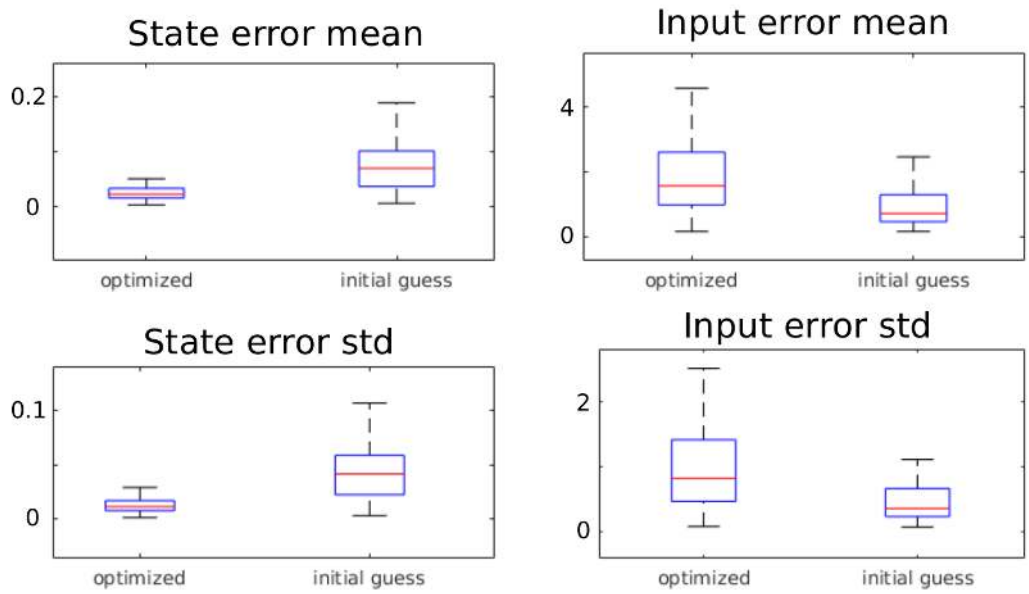

Figure 5.5 – Boxplots of the evaluated performances for the conducted statistical campaign. On the left, repartition of the tracking error means (top) and standard deviations (bottom) for the non-optimized and optimized trajectories w.r.t. problem (5.8). On the right, repartition of the integral input errors (top) and standard deviations (bottom) for the non-optimized and optimized trajectories w.r.t. problem (5.8).

which measures the discrepancy between the inputs that are planned based on the model parameters $\mathbf{p}_c$, and the realized inputs that occur when applying the control laws on the real system with parameters $\mathbf{p}$. This input error metric is suitable to check the effectiveness of problem (5.18). That being said, we found interesting to wonder how this metric is affected by the solving of the other problem (5.8). For this purpose, we measured the values of the input error $E_{\mathbf{u}}$ on the same set of non-optimized and optimized trajectories that were generated for the evaluation of the output tracking performance just before. From this data we generated new boxplots depicting the input errors characteristics. Fig. 5.5-right shows these new boxplots, with the two boxplots of the means of $E_{\mathbf{u}}$ on the top plot, and the two boxplots of the standard deviations of $E_{\mathbf{u}}$ on the bottom plot. What can be seen here, is that this time the optimized case is worse than the non-optimized case, *i.e.*, the input errors were higher in the optimized case than in the non-optimized case, and more spread. This means that the generation of trajectories that minimize the output tracking error (problem (5.8)) has the side effect of statistically making the input *less* predictable, and more subject to variations due to parameters error.

Therefore, this justifies even more the investigation of problem (5.18). The same method is thus applied to this problem, *i.e.*, for the same set of N_{traj} initial trajectories, 1) optimize them w.r.t. problem (5.18), and 2) run N simulations with perturbed parameters. Fig. 5.6-right shows the results of this second statistical analysis. We can

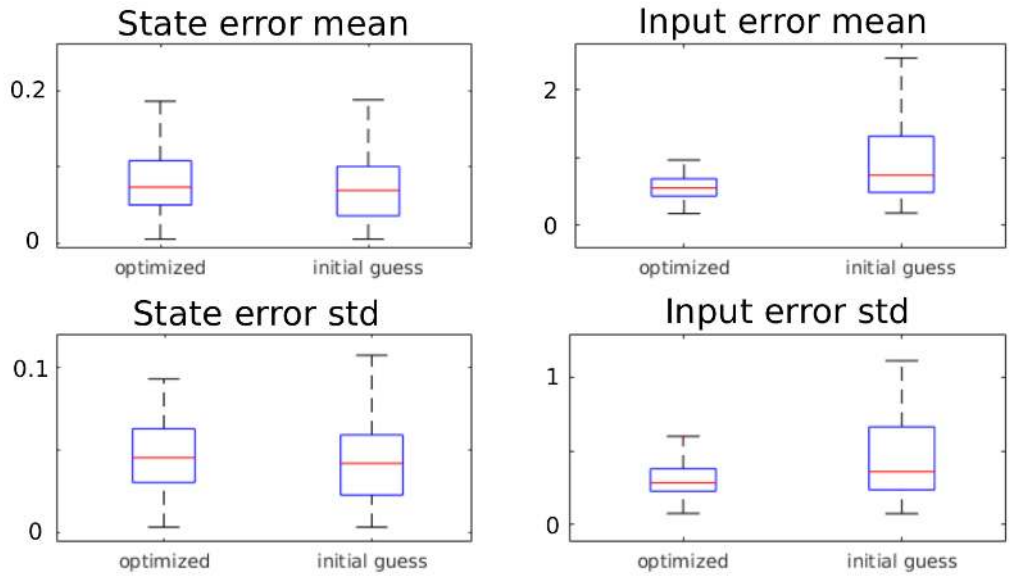

Figure 5.6 – Boxplots of the evaluated performances for the second statistical analysis. On the left, repartition of the tracking error means (top) and standard deviations (bottom) for the non-optimized and optimized trajectories w.r.t. problem (5.18). On the right, repartition of the integral input errors (top) and standard deviations (bottom) for the non-optimized and optimized trajectories w.r.t. problem (5.18).

observe that, as wished, the inputs are made less subject to perturbations in the parameters and more predictable by the optimization. This validates the effectiveness of our method for input sensitivity reduction. Then, Fig. 5.6-left shows the boxplots of the output tracking errors for the same trajectories. This time, we see that the optimization of the input almost did not affect the characteristics of the output, and just barely worsened them. This means that the output errors are not made more predictable by this optimization method.

Consequently, we observe that when optimizing the output errors, the input behaves in a more unpredictable way, and conversely when optimizing the input, the output errors are (slightly) deteriorated. The objectives of problem (5.8) and (5.18) thus seem conflicting. As a first approach to treat this issue, we propose, among other possibilities like Pareto front, to design a cost function to be minimized that integrates both the

information of the output and input problems.

To do so, we define the new cost function

$$J_w = \alpha \|\mathbf{\Pi}_\mathbf{y}(t_f)\|^2 + \beta \int_{t_0}^{t_f} \|\mathbf{\Theta}(\tau)\|^2_\mathbf{W} \, \mathrm{d}\tau \tag{5.24}$$

where $\alpha \in \mathbb{R}$ and $\beta \in \mathbb{R}$ are weights that determine the relative importance of each component. These weights are chosen as follows: for a given set of trajectory conditions (final position and so on), first solve the optimization problem (5.8). The resulting optimal trajectory vector $\mathbf{a_\Pi}$ results in a cost $J_1 = \|\mathbf{\Pi}_\mathbf{y}(\mathbf{a_\Pi}, t_f)\|^2$ which is supposedly minimal. We then use this minimal cost value as a reference to define the cost weight $\alpha = 1/J_1$. Doing the same for the other problem of the input optimization, we get an optimal trajectory vector $\mathbf{a_\Theta}$ associated with a minimal cost value $J_2 = J_\mathbf{\Theta}(\mathbf{a_\Theta})$, which results in the second weight value $\beta = 1/J_2$. These choices for the weights allow to somehow normalize the importance of both contributions. As an example to show the interest of such weights, if the solutions to both problem were the same, *i.e.*, $\mathbf{a_\Theta} = \mathbf{a_\Pi}$, we would have the weighted cost $J_w = 2$. Then if a slight difference exists between the solutions, the cost resembles $J_w(\mathbf{a_\Pi}) = 1 + \epsilon_1$ on the optimal trajectory vector $\mathbf{a_\Pi}$ of problem (5.8), and $J_w(\mathbf{a_\Theta}) = \epsilon_2 + 1$ on the optimal trajectory vector $\mathbf{a_\Theta}$ of problem (5.18). Thus, there should exist a trajectory vector 'between' those solutions that minimizes the weighted cost.

The solving of this method resulted in the boxplots of Fig. 5.7. On this third boxplot graph, we see that, as expected, both the input and output errors were reduced along the optimized trajectories compared to the non-optimized ones. Anyway, we note that the effectiveness of the input optimization was modest, with only a little reduction of the median input error over all the N_{traj} considered trajectory conditions. This is of course highly dependant on the choice of the weights in the weighted cost function (5.24). From the task perspective, if one wants to give more importance to the predictability of the input, then the weighted cost can be adjusted by increasing the weight β so that this component affects more the optimization.

As a side note, one can notice how in all the presented boxplots, the standard deviations always evolve in the same manner than the means in terms of height and spreading, *i.e.*, both reduced when the means do so, and conversely. This apparent correlation can be interpreted as a validation of the fact that the curvature of the error

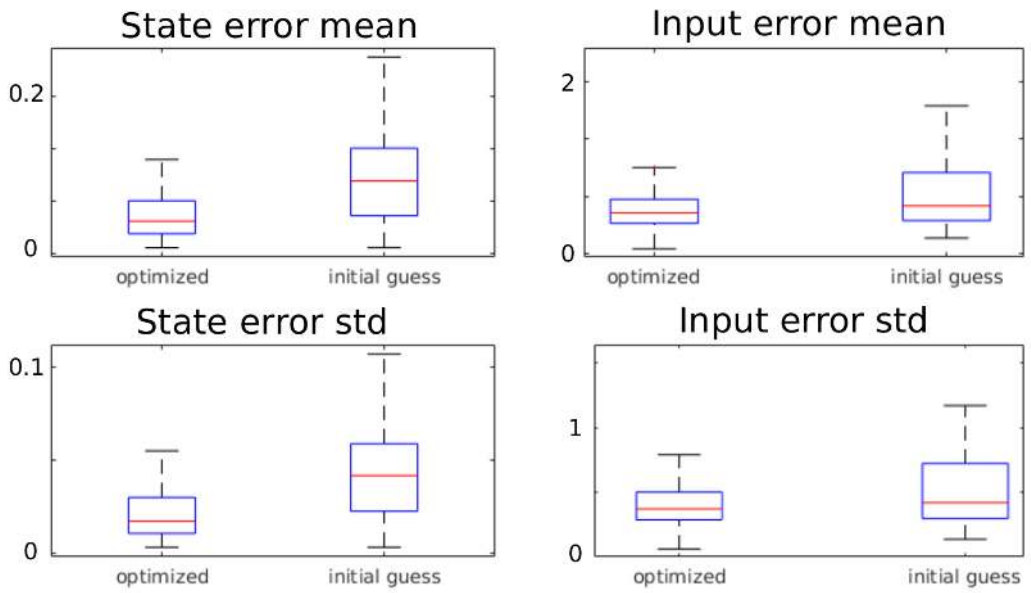

Figure 5.7 – Boxplots of the evaluated performances for the third statistical analysis. On the left, repartition of the tracking error means (top) and standard deviations (bottom) for the non-optimized and optimized trajectories w.r.t. the weighted cost (5.24). On the right, repartition of the integral input errors (top) and standard deviations (bottom) for the non-optimized and optimized trajectories w.r.t. the weighted cost (5.24).

profiles (led by the second order sensitivity and higher order terms) does not prevent our methods from working correctly. In fact, the standard deviation reduction tends to prove that the application of the optimizations effectively makes the error profiles more flat, as formally depicted on Fig. 5.3. Note that this holds for the input sensitivity as well, even if in that case there is only a relative error reduction, and no absolute error compensation such as $\mathbf{e}_f$.

5.5 Experimental validation

After having tested the effectiveness of the theory that was developed in this chapter through a large statistical analysis (see Sec. 5.4), we now present an experimental validation that we conducted with a real robot for the sake of application the described methods in practice. The experiments have been run on a Pioneer 3DX, see Fig. 5.8, that is standard commercial unicycle robot able to achieve any tracking task. Though it is capable of estimating its relative position in space and control it by itself with an

Figure 5.8 – The pioneer 3DX, a unicycle robot used for our experiments.

integrated servoing command, for this set of experiments we commanded this robot with low-level wheel velocity inputs (ω_R, ω_L), which allows us to perfectly master the behavior of the position control loop. Remaining consistent with the rest of our work, we implemented the DFL controller that was described and used in simulation so far. The hardware wheel velocity control is supposed to affect the dynamics as a bandwidth limitation that can be modeled with a first order linear low-pass filter, as in Sec. 5.2.

Although we could have leveraged the internal odometry, the measure of the position and orientation of the robot is done through the Vicon pose estimation system. This allows absolute positioning in the room frame, and prevents from integration biases that would affect the tracking performance in a non-modeled way. The DFL controller is implemented in MATLAB Simulink, on a laptop that is put on the robot and connected to it via USB in order to feed the velocity inputs to the hardware. Data is exchanged over Wi-Fi between the Vicon ground-station and the centralizing laptop.

The software architecture is based on the middleware genom (see Part I), which allows to interconnect the multiple programs that are involved for the whole machinery to operate. To maximize the repeatability of the experiments, the initialization of the trajectory is automated with a simple servoing that only authorizes the start of a certain trajectory tracking when both the position and orientation of the robot correspond to the wished initial configuration of the trajectory. To do so, we use the DFL controller for position initialization, followed by a simple differential PI control law for the orientation.

The calibration of the parameters is done by running the robot in open-loop over inputs that excite the dynamics enough [19, 10, 62]. We then solve the optimization prob-

lem that consists in finding the parameters that ensure the best fitting of the recorded behavior with a simulation of the dynamics that rely on the same open-loop inputs. Doing so allows us to get the wheel radius $r = 0.09$ [m], the distance between the wheel $b = 0.39$ [m], and also the bandwidth of the hardware control loop $b_w = 5$ [rd/s]. We also take note of the actuation saturations $|\omega_L|, |\omega_L| \leq 15$ [rd/s].

The tested cases are the following:

1. minimization of the output sensitivity (as in Chapter 4),
2. minimization of the input sensitivity as defined in Sec. 5.3,
3. minimization of the weighted sum,
4. minimization of the output sensitivity with compensation of the final control error as in problem (5.8).

As the theory is based on the idea that the parameters $\mathbf{p}$ of the system cannot be known with a better approximation than $\mathbf{p}_c$, we do not have in practice the reference case to compare the performance when applying the method. What can be measured instead is the behavior of the robot when tuning its controller with parameters that are different w.r.t. the best approximate available. In other words, we measure the validity of our method by checking that the wished reduction of the dependence of the errors to the parameters, *i.e.*, of the sensitivities, actually happens. To do so, we run the robot on our trajectory tracking tasks with each time two different configurations: the one where the parameters are set to the best estimation, and another one where we 'fake' the controller with an emulation of perturbed parameters that are 120% of their true values. Concretely, this is done by emulating a different set of parameters between the controller and the actual dynamics. This emulation of different parameter sets allows us to evaluate the behavior of the robot as if we had changed a bit its physical properties — but still all from a software tuning approach.

Note that for the three first experiments, the limitation of the input bandwidth of our real robot was not taken into account in the model of the dynamics. This allows us to test separately the method as described in Chapter 4 and the control error compensation envisioned for problem (5.8). As a consequence, the sensitivity equations used for solving the optimization problems of the three first experiments were based on the simple unicycle dynamics as derived in Sec. 4.4.1. In contrast, the last experiment was done on a trajectory generated with extended equations for the unicycle, that take into account the input bandwidth limitation.

As said, the first experiment consists in tracking a trajectory that is optimized w.r.t. the norm of the output sensitivity. The final position is set to $(1,\, 1)$ [m] with a trajectory duration of $t_f = 5$ [s]. We then apply the method described before to generate an optimal trajectory vector $\mathbf{a}_\Pi$. Fig. 5.9 shows the corresponding trajectories. The tracking of these trajectories was realized five times in identical conditions, in order to average the behavior and thus try to limit any circumstancial bias. The corresponding non-optimized and optimized trajectories where tracked by the real robot, which led to the results of Table 5.1, first and second columns. In this table, the numbers are the averaged error values over the repeated runs, with the following definitions: $E_{\mathbf{y}}(\mathbf{p}) = \|\mathbf{y}(t_f) - \mathbf{y}^*\|$ is the final tracking error for the parameters $\mathbf{p}$, and $E_{\mathbf{u}}$ is the input error as defined in eq. (5.23). The units in Table 5.1 are meters for the output error, and rad^{2}m^{-2}s^{-1} for the input error.

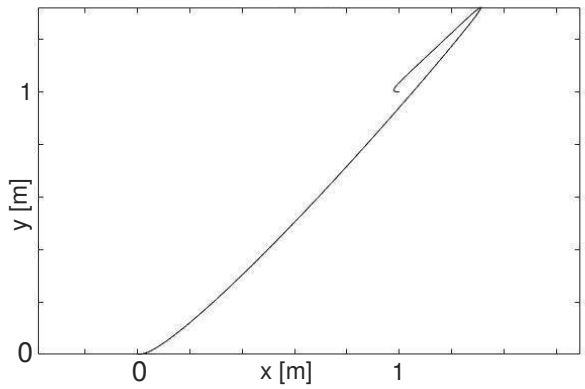

Figure 5.9 – Trajectory that was optimized for the output sensitivity. We see that the target is circumvented with a round maneuver, which allows the system to compensate the possible parameter errors by going back and forth.

	Initial guess	Output-optimized	Input-optimized	Weighted cost -optimized	Output-optimized with control error compensation
$E_{\mathbf{y}}(\mathbf{p}_c)$	0.0452	0.0335	0.0518	0.0647	0.0152
$E_{\mathbf{y}}(1.2\ \mathbf{p}_c)$	0.1016	0.0481	0.1143	0.0903	0.0396
$E_{\mathbf{u}}$	10.74	322.6	3.86	10.12	462.1

Table 5.1 – Average errors obtained by making the real robot track various non-optimal and optimal trajectories. Output errors are in m, and input errors in rad^{2}m^{-2}s^{-1}.

One can see that the error when not perturbing the parameters is not null, which as explained before comes from the fact that the real parameters $\mathbf{p}$ of the system remain

unknown to us. Moreover, the control lag was not modeled in these tests, and thus result in a tracking error that cannot be cancelled with this method. Obviously, note that other non-modeled perturbation may occur and participate in the tracking deterioration. Therefore, even with our best estimate for the control parameters $\mathbf{p}_c$, we get a positive final tracking error for any tested trajectory.

The interesting point to be raised, however, is that the final error when tracking the output-optimized trajectory is smaller in the perturbed case (second line) that the error we get when perturbing identically the parameters but on the tracking of the initial trajectory. That is, the output-optimized trajectory is such that the increase of the error when tracking it with wrong parameters is small, *i.e.*, it is less than the one obtained on the non-optimized trajectory in the same circumstances. This confirms the soundness of the output sensitivity minimization technique.

A second point that can be raised, is that the error in the nominal case (not perturbed), is also lower for the output-optimized trajectory than for the initial one. This tends to prove that the error may be partly explicable by an imperfection of the parameters: reducing the sensitivity of this error w.r.t. the parameters reduced the error because it partly comes from the parameters.

Finally, one can note that the input error $E_{\mathbf{u}}$ was highly increased by the trajectory optimization w.r.t. the output problem. As discussed before, this is not a big surprise because the input behavior is not mastered at all during this optimization process (except for the saturations). In this particular case, this resulted in a high sensitivity of the input w.r.t. the parameters, which means that the inputs are highly susceptible of changing from a parameter set to another even close.

The second experiment consists in the tracking of a trajectory optimized for problem (5.18). In solving this problem, the weighting matrix is chosen unitary as in the tests of Sec. 5.3, because the inputs are symmetric and thus do not necessitate any distinction. The resulting trajectory is showed in Fig. 5.10. We see that this trajectory do not feature the final circumvention of the target, which makes sense because this maneuver would probably make the input more shaky and thus less predictable. The quantitative results are reported in the third column of Table 5.1. We see that the output tracking error $E_{\mathbf{y}}$ were pretty bad in this experiment. As discussed, this is well expected since the objective of the optimization problem was different from the previous one, and thus there is no reason for the tracking performance to improve. On the other hand,

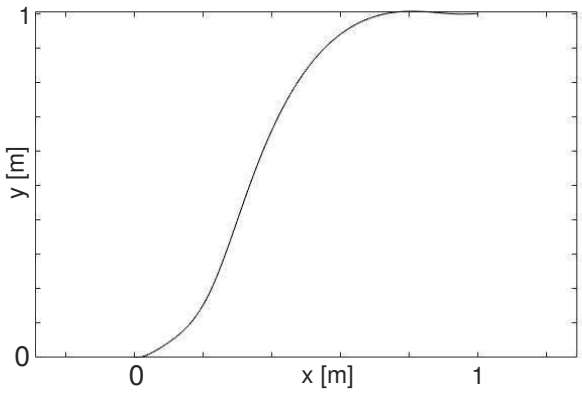

Figure 5.10 – Trajectory that was optimized for the input.

what we are interested in here is the input behavior: we clearly see that the tracking of this input-optimized trajectory led to smaller input error $E_{\mathbf{u}}$, which demonstrates that the behavior of the input is more predictable in this case, because it less affected by the parameters.

The third trajectory that we tested is the one generated with the weighted cost described before. As explained, we chose to set the two weights to the inverse of the obtained cost for the two previous optimization. The optimization led to the trajectory of Fig. 5.11, which again features an interesting rounded maneuver near the target. We

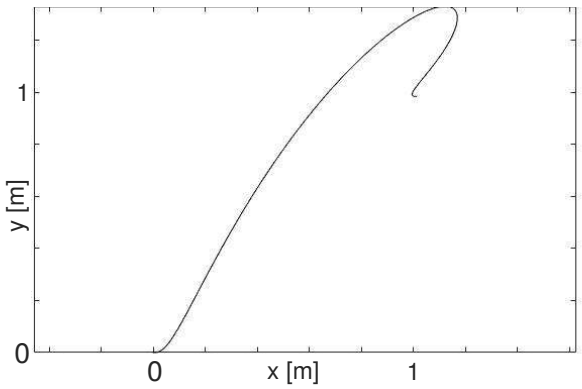

Figure 5.11 – Trajectory that was optimized for the weighted sum.

see that the resulting output error was reduced in the perturbed case compared to the initial guess. However, we see that the input error is almost unchanged compared to the initial guess. This is explicable by the fact that the weighted sum technique is not

able to correctly tune the importance of each cost, even with a normalization of the weights as we did. This also tends to prove that the link between the two objectives of input and output optimization (as treated in this Thesis) is complex and needs more investigation for us to be able to control them both.

Finally, we conducted a last experiment in which the robot tracked a trajectory that was optimized with the control error compensation that we described in Sec. 5.2. As said, we derived to this end the sensitivity equations for the case of the unicycle with limited input bandwidth. We chose not to implement the weighted sum cost for this test, in order to evaluate purely the ability of our framework to reduce the output error without caring for the input behavior. The results of this trial are reported in the last column of Table 5.1. One can see that in this case, the error is significantly reduced w.r.t. the others tested cases. The final control error compensation actually allowed the system to reach the target with ore precision, as wished. We also see that the increase of the error in the perturbed case is quite low, confirming that the variations in the parameters do not affect, or at least very slightly, the behavior of the system at the end of the trajectory.

5.6 Conclusion and perspectives

As a conclusion, in this chapter we have developed the sensitivity framework, yielding a few methods to improve the precision of trajectory tracking tasks. First, we have shown how the sensitivity metric can emerge from the wish of minimizing the output tracking error of a robot. This allowed us to set up a first optimization problem (5.8), that captures the idea of optimizing the runtime behavior at the prior planning stage by integrating some information about the controller into the trajectory generation. Moreover, this problem has been constructed such that it does not require to know the actual system parameters $\mathbf{p}$, but yet it ensures that the tracking errors are minimized in a certain way (see Sec. 5.2 for details of what it means).

Furthermore, a number of simulations have been conducted during and after this theoretical derivation, which we think justify the formulation and approach that we have chosen to pursue. This also led us to the construction of a second and third optimization formulation, derived in problem (5.18) and cost (5.24), that extend the application scope of our trajectory generation methods. The multiple analysis and tests conducted in

simulation made it possible for us to construct, refine and validate these methods in the most relevant way possible. In particular, we have conducted a large statistical analysis that aims at validating in depth the consistence of our 'sensitivity framework'. The results have shown that using these methods to improve the quality of the robotic behavior is sound.

Finally, we conducted a series of real experiment on a unicycle robot, that gave us the opportunity to test the methods for real. The results once again show that the methods make sense and improve the performances in terms of output tracking error and input predictability.

Based on this work, the perspectives for future studies are multiple:

1. apply the new closed-loop framework to the generation of trajectories for the MonkeyRotor (which should improve the tracking performance even more than the preliminary open-loop framework that was used in Chapter 2),
2. improve the computational speed,
3. apply the framework to other robotic case studies such as tele-operation with haptics,
4. evaluate the relevance of model simplification/reduction to make the analytical expressions more tractable.

CONCLUSION

In this Thesis we have presented our contributions to the fields of aerial robotics and trajectory planning under uncertainty. In particular, we have studied the concept of aerial locomotion, which consists in developing a special application of aerial navigation assisted by physical contacts with the environment. To do so, we have derived the case study of the contact-fly-contact problem for the MonkeyRotor, a quadrotor equipped with an actuated arm. This specific problem is the one of finding a suitable overall trajectory that brings the MonkeyRotor from an initial configuration with a physical contact, to a final configuration with another contact with the environment. After studying the switching dynamics and the features of this system (including the fact that the system is underactuated when flying but overactuated when hooked to the environment), we have proposed a planning algorithm that is able to provide feasible trajectories for the whole task of going from the first configuration with contact to the second one. This trajectory generation algorithm is able to propose near-optimal trajectories for an objective such as the time minimization, which we applied to numerous simulations. Thanks to this framework, we studied the effect of having a limited thrust to realize the locomotion task. Interesting trajectory features emerged from the trajectory generation that were not imposed by design, such as a back swing at the beginning for helping the MonkeyRotor to build enough kinetic energy before the flying phase. We also implemented a special cost that aims at maximizing the precision of the re-hooking event despite discrepancies in the parameter knowledge, by means of the so-called open-loop state sensitivity minimization. This cost helps in mitigating the negative effects of poorly known model parameters in the trajectory tracking performance. We verified in a number of simulations that this effectively improves the precision of the system when the controller works with perturbed parameters.

In a second part, we have studied more in details the concept of the sensitivity minimization that we think is interesting in general for any robotic trajectory planning under parametric uncertainty. Based on previous works that were conducted in this field, we proposed a new control-aware scheme that is able to integrate some knowledge concerning the controller at the planning stage (prior to tracking execution). We thus

derived a more general theory that aims at constructing a more solid framework for trajectory optimization, and validated it through statistical analysis based on simulations of two robots (a unicycle and a quadrotor). After checking the validity of the theory when tested on these systems with perfect controllers, *i.e.*, able to track a trajectory with zero error when the parameters are exactly known, we extended the theoretical derivation to the cases were the controller has some limitations, *i.e.*, may not be able to track a trajectory because of other issues (e.g., limited control authority or unmodeled second order effects), even with correctly known parameters. We also extended the concept to other sensitivity metrics such as the input sensitivity, which is an interesting metric to quantify the ability of a trajectory in minimizing the deviations of the control efforts also in presence of parameter discrepancies. After setting up implementations of the required algorithms, we then tested the framework in large-scale statistical analysis, and in a real experiment with a unicycle robot. The results confirmed the benefits of using our framework for trajectory generation, with improved precision in the trajectory tracking as wished.

Future work on these two fields should include:

1. validation of the developed locomotion concept through experiment with the real MonkeyRotor prototype,
2. extension of the aerial locomotion trajectory generation framework to more general contexts such as multiple branches and 3D space,
3. application of the sensitivity framework to other robots with uncertain parameters,
4. improvements in the solving method for the sensitivity optimization, e.g., based on model simplification/reduction to speed up the computations.

BIBLIOGRAPHY

[1] D. Abeywardena et al., « Improved State Estimation in Quadrotor MAVs: A Novel Drift-Free Velocity Estimator », *in*: *IEEE Robotics & Automation Magazine* 20.*4* (2013), pp. 32–39.

[2] AeRoArms, *EU Collab. Project ICT-644271*, www.aeroarms-project.eu, 2015-2019.

[3] AEROworks, *EU Collab. Project ICT-644128*, www.aeroworks2020.eu.

[4] AIRobots, *EU Collab. Project ICT-248669*, www.airobots.eu, 2010-2013.

[5] Alex Ansari and Todd Murphey, « Minimum Sensitivity Control for Planning with Parametric and Hybrid Uncertainty », *in*: *Int. J. Rob. Res.* 35.*7* (June 2016), pp. 823–839, ISSN: 0278-3649, DOI: 10.1177/0278364915600536, URL: http://dx.doi.org/10.1177/0278364915600536.

[6] G. Antonelli et al., « Adaptive Trajectory Tracking for Quadrotor MAVs in Presence of Parameter Uncertainties and External Disturbances », *in*: *IEEE Transactions on Control Systems Technology* (2017).

[7] Tomoki Anzai et al., « Aerial Grasping Based on Shape Adaptive Transformation by HALO: Horizontal Plane Transformable Aerial Robot with Closed-Loop Multilinks Structure », *in*: May 2018, pp. 6990–6996, DOI: 10.1109/ICRA.2018.8460928.

[8] ARCAS, *EU Collab. Project ICT-287617*, www.arcas-project.eu, 2011-2015.

[9] Uri M. Ascher and Linda R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, 1st, Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1998, ISBN: 0898714125.

[10] A. Bahloul, S. Tliba, and Y. Chitour, « Dynamic Parameters Identification of an Industrial Robot: A Constrained Nonlinear WLS Approach », *in*: *2018 26th Mediterranean Conference on Control and Automation (MED)*, 2018, pp. 1–6, DOI: 10.1109/MED.2018.8442630.

[11] Moses Bangura and Robert Mahony, « Real-time Model Predictive Control for Quadrotors », *in*: *IFAC Proceedings Volumes* 47.*3* (2014), 19th IFAC World Congress, pp. 11773 –11780, ISSN: 1474-6670, DOI: https://doi.org/10.3182/20140824-6-ZA-1003.00203, URL: http://www.sciencedirect.com/science/article/pii/S1474667016434890.

[12] A. Becker and T. Bretl, « Approximate Steering of a Unicycle Under Bounded Model Perturbation Using Ensemble Control », *in*: *TRo* 28.*13* (2012), pp. 580–591.

[13] Richard Ernest Bellman, *Dynamic Programming*, New York, NY, USA: Dover Publications, Inc., 2003, ISBN: 0486428095.

[14] S. Bouabdallah and R. Siegwart, « Backstepping and sliding-mode techniques applied to an indoor micro quadrotor », *in*: *2005 ICRA*, 2005, pp. 2247–2252.

[15] O. Bourquardez et al., « Image-Based Visual Servo Control of the Translation Kinematics of a Quadrotor Aerial Vehicle », *in*: *IEEE Transactions on Robotics* 25.*3* (2009), pp. 1552–3098.

[16] P. Byrne and M. Burke, « Optimization with trajectory sensitivity considerations », *in*: *IEEE Transactions on Automatic Control* 21.*2* (1976), pp. 282–283, ISSN: 0018-9286, DOI: 10.1109/TAC.1976.1101182.

[17] S. Candido and S. Hutchinson, « Minimum uncertainty robot navigation using information-guided POMDP planning », *in*: *IEEE Int. Conf. on Robotics and Automation*, 2011, 6102–6108.

[18] A. Censi, L. Marchionni, and G. Oriolo, « Simultaneous maximum-likelihood calibration of robot and sensor parameters », *in*: *2008 ICRA*, Pasadena, CA, 2008, pp. 2098–2103.

[19] A. Censi et al., « Simultaneous Maximum-likelihood Calibration of Odometry and Sensor Parameters », *in*: *IEEE Transactions on Robotics* 29.*2* (2013), pp. 475–492.

[20] Piwai N. Chikasha and Chioniso Dube, « Adaptive Model Predictive Control of a Quadrotor », *in*: *IFAC-PapersOnLine* 50.*2* (2017), Control Conference Africa CCA 2017, pp. 157 –162, ISSN: 2405-8963, DOI: https://doi.org/10.1016/j.ifacol.2017.12.029, URL: http://www.sciencedirect.com/science/article/pii/S2405896317335656.

[21] Marco Cognetti, Paolo Salaris, and Paolo Robuffo Giordano, « Optimal Active Sensing with Process and Measurement Noise », *in*: *ICRA 2018 - IEEE International Conference on Robotics and Automation*, Brisbane, Australia: IEEE, May 2018, pp. 2118–2125, DOI: 10.1109/ICRA.2018.8460476, URL: https://hal.inria.fr/hal-01717180.

[22] I. D. Cowling, J. F. Whidborne, and A. K. Cooke, « Optimal Trajectory Planning and LQR Control for a Quadrotor UAV », *in*: *ICC*, Glasgow, Scotland, 2006.

[23] Davide Falanga et al., « The Foldable Drone: A Morphing Quadrotor That Can Squeeze and Fly », *in*: *IEEE Robotics and Automation Letters* PP (Dec. 2018), pp. 1–1, DOI: 10.1109/LRA.2018.2885575.

[24] M. Fliess et al., « Flatness and defect of nonlinear systems: Introductory theory and examples », *in*: *IJC* 61.*6* (1995), pp. 1327–1361.

[25] J. Ghommam and M. Saad, « Autonomous Landing of a Quadrotor on a Moving Platform », *in*: *IEEE Transactions on Aerospace and Electronic Systems* 53.*3* (2017), pp. 1504–1519, DOI: 10.1109/TAES.2017.2671698.

[26] G. Gioioso et al., « A Force-based Bilateral Teleoperation Framework for Aerial Robots in Contact with the Environment », *in*: *2015 ICRA*, Seattle, WA, 2015, pp. 318–324.

[27] V. Grabe, H. H. Bülthoff, and P. Robuffo Giordano, « On-board velocity estimation and closed-loop control of a quadrotor UAV based on optical flow », *in*: *2012 ICRA*, St. Paul, MN, 2012, pp. 491–497.

[28] K. Hausmana et al., « Observability- aware trajectory optimization for self-calibration with application to uavs », *in*: *IEEE Robotics and Automation Letters* (2017).

[29] Markus Hehn, Robin Ritz, and Raffaello DÕAndrea, « Performance Benchmarking of Quadrotor Systems Using Time-Optimal Control », *in*: *Autonomous Robots* 33.*1–2* (2012), pp. 69–88.

[30] B. Houska and M. E. Villanueva, « Robust Optimization for MPC », *in*: *Handbook of Model Predictive Control*, Springer, 2018, pp. 415–447.

[31] D. C. Jiles and D. L. Atherton, « Theory of ferromagnetic hysteresis (invited) », *in*: *Journal of Applied Physics* 55.*6* (1984), pp. 2115–2120, DOI: 10.1063/1.333582, eprint: https://doi.org/10.1063/1.333582, URL: https://doi.org/10.1063/1.333582.

[32] M. Jufer, *Electromecanique*, Traite d'electricite de l'Ecole polytechnique federale de Lausanne, Presses polytechniques et universitaires romandes, 1995, ISBN: 9782880742850.

[33] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, « Optimization by Simulated Annealing », *in*: *Science* 220.*4598* (1983), pp. 671–680, ISSN: 0036-8075, DOI: 10.1126/science.220.4598.671, eprint: https://science.sciencemag.org/content/220/4598/671.full.pdf, URL: https://science.sciencemag.org/content/220/4598/671.

[34] E. Kreindler, « Formulation of the minimum trajectory sensitivity problem », *in*: *IEEE Transactions on Automatic Control* 14.*2* (1969), pp. 206–207, ISSN: 0018-9286, DOI: 10.1109/TAC.1969.1099130.

[35] D. Kubus, T. Kroger, and F. M. Wahl, « On-line estimation of inertial parameters using a recursive total least-squares approach », *in*: *2008 IROS*, Nice, France, 2008, pp. 3845–3852.

[36] T. Lee, M. Leoky, and N. H. McClamroch, « Geometric tracking control of a quadrotor UAV on SE(3) », *in*: *49th CDC*, Atlanta, GA, 2010, pp. 5420–5425.

[37] R. Mahony, V. Kumar, and P. Corke, « Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor », *in*: *IEEE Robotics & Automation Magazine* 19.*3* (2012), pp. 20–32.

[38] R. Mahony, V. Kumar, and P. Corke, « Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor », *in*: *IEEE Robotics & Automation Magazine* 19.*3* (2012), pp. 20–32.

[39] A. Majumdar and R. Tedrake, « Funnel libraries for real-time robust feedback motion planning », *in*: *IJRR* 36.*18* (2017), pp. 947–982.

[40] P. Martin, R. M. Murray, and P. Rouchon, « Flat systems, equivalence and trajectory generation », *in*: *2003 CDS Technical Report*, 2003.

[41] C. Masone et al., « Interactive Planning of Persistent Trajectories for Human-Assisted Navigation of Mobile Robots », *in*: *2012 IROS*, Vilamoura, Portugal, 2012, pp. 2641–2648.

[42] C. Masone et al., « Semi-autonomous Trajectory Generation for Mobile Robots with Integral Haptic Shared Control », *in*: *2014 ICRA*, Hong Kong, China, 2014, pp. 6468–6475.

[43] C. Masone et al., « Shared Trajectory Planning for Human-in-the-loop Navigation of Mobile Robots in Cluttered Environments », *in*: *5th Int. Work. on Human-Friendly Robotics*, Bruxelles, Belgium, 2012.

[44] D. Mellinger and V. Kumar, « Minimum Snap Trajectory Generation and Control for Quadrotors », *in*: *2011 ICRA*, Shanghai, China, 2011, pp. 2520–2525.

[45] D. Mellinger et al., « Design, modeling, estimation and control for aerial grasping and manipulation », *in*: *2011 IROS*, San Francisco, CA, 2011, pp. 2668–2673.

[46] Daniel Mellinger, Michael Shomin, and Raghvendra Kumar, « Control of Quadrotors for Robust Perching and Landing », *in*: 2010.

[47] Brian Vincent Mirtich, « Impulse-based Dynamic Simulation of Rigid Body Systems », AAI9723116, PhD thesis, 1996, ISBN: 0-591-32089-4.

[48] V. Mistler, A. Benallegue, and N.K. M'Sirdi, « Exact linearization and noninteracting control of a 4 rotors helicopter via dynamic feedback », *in*: *Proc. of the 2001 IEEE Int. Worksop on Robot and Human Interactive Communication*, 2001, pp. 586–593.

[49] Marco M. Nicotra, Roberto Naldi, and Emanuele Garone, « Nonlinear control of a tethered UAV: The taut cable case », *in*: *Automatica* 78 (2017), pp. 174–184.

[50] M. J. Van Nieuwstadt and R. M. Murray, « Real-time trajectory generation for differentially flat systems », *in*: *IJRNC* 8 (1998), pp. 995–1020.

[51] G. Oriolo, A. De Luca, and M. Vendittelli, « WMR control via dynamic feedback linearization: Design, implementation and experimental validation », *in*: *IEEE Transactions on Control Systems Technology* 10.*6* (2002), pp. 835–852.

[52] Bryan Penin, Paolo Giordano, and François Chaumette, « Vision-Based Reactive Planning for Aggressive Target Tracking while Avoiding Collisions and Occlusions », *in*: *IEEE Robotics and Automation Letters* PP (July 2018), DOI: 10.1109/LRA.2018.2856526.

[53] G. V. Raffo, M. G. Ortega, and F. R. Rubio, « An integral predictive/nonlinear H-infinity control structure for a quadrotor helicopter », *in*: *Automatica* 46.*1* (2010), pp. 29–39.

[54] M. Ryll et al., « 6D Physical Interaction with a Fully Actuated Aerial Robot », *in*: *2017 ICRA*, Singapore, 2017.

[55] Markus Ryll, Davide Bicego, and Antonio Franchi, « A Truly Redundant Aerial Manipulator exploiting a Multi-directional Thrust Base », *in*: *12th IFAC Symposium on Robot Control*, Budapest, Hungary, 2018.

[56] I. Sa and P. Corke, « System identification, estimation and control for a cost effective open-source quadcopter », *in*: *2012 ICRA*, St. Paul, MN, 2012, pp. 2035–2041.

[57] D. E. Soltero, S. L. Smith, and D. Rus, « Collision Avoidance for Persistent Monitoring in Multi-Robot Systems with Intersecting Trajectories », *in*: *2011 IROS*, San Francisco, CA, 2011, pp. 3645–3652.

[58] S.Ponda, R.Kolacinski, and E.Frazzoli, « Trajectory Optimization for Target Localization Using Small Unmanned Aerial Vehicles », *in*: *AIAA Guidance, Navigation, and Control Conference*, 2012.

[59] William Squire and George Trapp, « Using Complex Variables to Estimate Derivatives of Real Functions », *in*: *SIAM Review* 40 (1998), pp. 110–112.

[60] Nicolas Staub et al., « The Tele-MAGMaS: an Aerial-Ground Co-manipulator System », *in*: *IEEE Robotics and Automation Magazine* 25 (2018), pp. 66–75.

[61] Nicolas Staub et al., « Towards a Flying Assistant Paradigm: the OTHex », *in*: *2018 IEEE Int. Conf. on Robotics and Automation*, Brisbane, Australia, 2018, pp. 6997–7002.

[62] J. Swevers et al., « EXPERIMENTAL ROBOT IDENTIFICATION USING OPTIMISED PERIODIC TRAJECTORIES », *in*: *Mechanical Systems and Signal Processing* 10.*5* (1996), pp. 561 –577, ISSN: 0888-3270, DOI: https://doi.org/10.1006/mssp.1996.0039, URL: http://www.sciencedirect.com/science/article/pii/S0888327096900394.

[63] Russ Tedrake and the Drake Development Team, *Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems*, 2016, URL: http://drake.mit.edu.

[64] J. Thomas et al., « Toward autonomous avian-inspired grasping for micro aerial vehicles », *in*: *Bioinspir. Biomim.* 9.*2* (2014).

[65] M. Tognon and A. Franchi, « Dynamics, Control, and Estimation for Aerial Robots Tethered by Cables or Bars », *in*: *IEEE Transactions on Robotics* 33.*4* (2017), pp. 834–845, ISSN: 1552-3098, DOI: 10.1109/TRO.2017.2677915.

[66] M. Tognon et al., « Dynamic Decentralized Control for Protocentric Aerial Manipulators », *in*: *2017 ICRA*, Singapore, 2017.

[67] A. D. Wilson, J. A. Schultz, and T. D. Murphey, « Trajectory Optimization for Well-Conditioned Parameter Estimation », *in*: *IEEE Transactions on Automation Science and Engineering* 11.*1* (2015), pp. 28–36.

[68] Y. Yong, T. Arima, and S. Tsujio, « Inertia parameter estimation of planar object in pushing operation », *in*: *2005 ICIA*, Hong Kong and Macau, China, 2005, pp. 356–361.

[69] B. Yüksel, N. Staub, and A. Franchi, « Aerial Robots with Rigid/Elastic-joint Arms: Single-joint Controllability Study and Preliminary Experiments », *in*: *2016 IROS*, Daejeon, South Korea, 2016, pp. 1667–1672.

[70] B. Yüksel et al., « A Nonlinear Force Observer for Quadrotors and Application to Physical Interactive Tasks », *in*: *2014 AIM*, Besançon, France, 2014, pp. 433–440.

[71] K. Zhang et al., « SpiderMAV: Perching and stabilizing micro aerial vehicles with bio-inspired tensile anchoring systems », *in*: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 6849–6854, DOI: 10.1109/IROS.2017.8206606.

Titre : Algorithmes d'estimation et de commande pour des quadrirotors en interaction physique avec l'environnement

Mot clés : robotique aérienne, locomotion, planification de trajectoires

Résumé : Le champ de la robotique aérienne pour l'interaction physique permet aujourd'hui à un robot aérien d'appliquer un effort maîtrisé sur un objet ou sur l'environnement alors qu'il vole. En s'inspirant de l'utilisation des contacts faite en robotique humanoïde, nous proposons dans cette thèse de s'appuyer sur ces approches pour dépasser l'idée que l'environnement est une contrainte, en exploitant le contact physique avec celui-ci dans le but de réaliser de la locomotion aérienne. Cette idée est étudiée et démontrée au travers de simulations et expérimentations d'une nouvelle plateforme robotique aérienne consistant en un quadrirotor équipé d'un bras robotique a 1 degré de liberté. D'autre part, nous avons aussi étudié le problème de la génération de trajectoires dont la sensibilité aux paramètres du modèle est minimale. Ce problème est généralisé à n'importe quel robot, et se révèle particulièrement approprié dans le cas du quadrirotor du fait de l'incertitude importante concernant ses paramètres inertiels et d'actionnement. Pour traiter ce problème, nous définissons et utilisons la "sensibilité de l'état aux paramètres" afin de générer des trajectoires dont la sensibilité aux paramètres est minimale, garantissant une forte robustesse.

Title: Algorithms for estimation and control of quadrotors with physical interaction with their environnement

Keywords: aerial robotics, locomotion, trajectory planning

Abstract: In recent years, the field of aerial robotics has been improved, allowing the UAVs to apply a controlled wrench on their environment or on an object while flying.Inspired by the use of contacts in legged robots, in this Thesis, we propose the idea of exploiting physical contact with the environment for the purpose of 'locomotion' during flight, with the goal of going beyond the common thought that the surrounding environment is a constraint to avoid. These ideas are studied and demonstrated in simulations and experiments on a novel aerial platform consisting of a quadrotor with a 1-dof arm that realizes maneuvers by leveraging contacts with pivot points. Additionally, we also study the problem of generating trajectories that are most insensitive to variations in the model parameters.This problem has a general validity for any robot, and it is particularly relevant for UAVs because of the high uncertainty in their inertial parameters and actuation. In order to address these issues, we define and leverage the novel notion of "closed-loop state sensitivity" for generating trajectories that are minimally-sensitive to parameters with high robustness guarantees.