



UNIVERSITÉ  
DE LORRAINE

École doctorale IAEM

---

# Apprentissage d'une loi de commande optimale d'un petit quadrotor pour le vol dans des tuyaux cylindriques

## THÈSE

présentée et soutenue publiquement le 27 juin 2022  
pour l'obtention du

**Doctorat de l'Université de Lorraine  
(mention informatique)**

par

Vladislav Tempez

### Composition du jury

<i>Président :</i>	Thibaut RAHARIJAONA	Professeur des Universités (HDR), Université de Lorraine
<i>Rapporteurs :</i>	Pascal MORIN Emmanuel Rachelson	Professeur des Universités (HDR), ISIR Professeur des Universités (HDR), ISAE-SUPAERO
<i>Examinateurs :</i>	Madiha NADRI-WOLF Thibaut RAHARIJAONA	Maitresse de conférences, Université Claude Bernard Professeur des Universités (HDR), Université de Lorraine
<i>Encadrants :</i>	Jean-Baptiste MOURET Franck RUFFIER	Directeur de Recherche (HDR), Inria Directeur de Recherche (HDR), CNRS

---

**Laboratoire Lorrain de Recherche en Informatique et ses Applications - UMR 7503**  
**Institut National de la Recherche en Informatique et en Automatique - INRIA**



# Remerciements

Je souhaite d'abord remercier Jean-Baptiste Mouret et Franck Ruffier qui ont dirigé cette thèse. Pendant ces 4 ans ils m'ont offert leurs conseils, encouragements, répondu à mes questions et c'est grâce à leur aide que j'ai pu réaliser cette thèse.

Je remercie Emmanuel Rachelson et Pascal Morin pour avoir pris le temps d'évaluer ce manuscrit et participé au jury avec Madiha Nadri-Wolf et Thibaut Raharijaona. Merci pour le temps accordé à l'évaluation de mon travail.

Merci à Alain Dutech pour son regard extérieur sur l'avancement de ma thèse, tu m'as donné un peu de recul bienvenu durant cette période pas toujours facile.

Je souhaite aussi remercier tout particulièrement Lucien Renaud pour toute son assistance dans le combat sans fin contre la fragilité des Crazyflies, sans toi j'aurais fait bien plus de simulations encore.

Merci à Nima Medhi pour ses lumières sur les algorithmes de filtre, et merci à Pierre Laclau et Corentin Bunel pour ces échanges autour des caprices du Crazyfly.

Merci aussi à tous les membres de l'équipe Larsen avec qui j'ai pu échanger lors de nos réunions d'équipe.

Merci à Nicolae Brinzei et Jean-François Pétin qui m'ont bien accompagné dans mes enseignements à l'ENSEM.

Merci à tous mes camarades de jeu-du-midi avec qui j'ai pu passer d'excellents moments de détente bienvenue, merci à Timothée, Yoann, Raphaël, Oriane, Yassine, Nima, Nicolas, Lucien et les autres pour ces discussions enrichissantes et les projets parfois loufoques.

Je ne me serais pas retrouvé avec l'opportunité de commencer cette thèse si ma curiosité n'avait pas été stimulée et soutenue au préalable. Je le dois à de nombreux professeurs durant ma scolarité, et à Monique, Georges, Karine et Valentin. Merci à Anna et Igor m'ont permis d'en arriver là où je suis.

Et merci à Aude, tu m'as bien rendu la pareille.



# Abstract

This work deals with the flight of small quadrotors (essentially the Crazyflie,  $\sim 10\text{cm}$ , 30g) for indoor environment such as pipes. Three main questions are dealt with in the thesis : the existence and nature of aerodynamic perturbations caused by the interaction of rotors' airflow with the environment, the design of a flight controller fit for environments like pipes and the adaptation of this controller to be embedded onboard a Crazyflie controller.

The static component of aerodynamic perturbations is measured and we detail a map of these for a given set of locations in the environment. These measures are then used to build a model that is able to predict these perturbations at any point of the environment.

We propose a MPC flight controller based on the resolution of the optimal control problem, taking the perturbation model into account. This controller is able to plan trajectories considering the perturbations and to reject these perturbations.

Solving the optimal control problem being too computationally expensive to be done in real time onboard the Crazyflie, we propose to learn a neural network approximating the MPC flight controller in a supervised way, by imitation. This neural network approximation is able to be executed in real time onboard a Crazyflie.



# Résumé

Cette thèse traite du vol de petits quadrotors (en particulier du modèle Crazyflie,  $\sim 10\text{cm}, 30\text{g}$ ) dans des environnements confinés comme les tuyaux. Trois problématiques sont principalement abordées dans cette thèse : la présence et la nature de perturbations dues à l'interaction entre les flux d'air déplacés par les rotors et les parois de l'environnement, la conception d'un contrôleur pour le vol dans un tel environnement malgré les perturbations et l'adaptation de ce contrôleur aux contraintes de fonctionnement à bord d'un petit quadrotor comme le Crazyflie.

Les perturbations aérodynamiques sont mesurées en régime statique et nous présentons une cartographie de celles-ci en un ensemble de points donné de l'environnement. Ces mesures sont utilisées pour dériver un modèle capable de réaliser des prédictions de ces perturbations en tout point de l'environnement.

Le contrôleur proposé pour voler dans un tel environnement est un contrôleur MPC basé sur la résolution de problèmes de contrôle optimal intégrant le modèle des perturbations, permettant ainsi à la fois la planification de la navigation en tenant compte des perturbations à venir et le rejet de celles qui seraient hors modèle.

La résolution de ces problèmes de contrôle optimal étant trop coûteuse en calcul pour être réalisée en temps réel à bord d'un petit quadrotor comme le Crazyflie, cette thèse aborde ensuite l'apprentissage par imitation du contrôleur MPC par un réseau de neurones qui permet d'obtenir une approximation de ce contrôleur dont le coût en calcul est compatible avec l'utilisation à bord d'un Crazyflie.



# Table des matières

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Les robots pour l'exploration . . . . .	1
1.2	Diversité des formes des robots . . . . .	2
1.3	Approches pour la commande des robots . . . . .	5
1.4	Objectifs . . . . .	8
1.5	Contribution . . . . .	11
<b>2</b>	<b>Modélisation et contrôle de multirotors</b>	<b>15</b>
2.1	Introduction . . . . .	15
2.2	Travaux connexes . . . . .	16
2.3	Dynamique et équations des multirotors . . . . .	20
2.3.1	Quelques repères et notations utiles . . . . .	20
2.3.2	Équations de la cinématique et représentation de l'orientation . . . . .	22
2.3.3	Modèles dynamiques des multirotors . . . . .	22
2.3.4	Le cas du quadrotor <i>Crazyflie</i> . . . . .	24
2.4	Contrôle optimal . . . . .	26
2.4.1	Formulation et théorie du contrôle optimal . . . . .	26
2.4.2	Résolution du contrôle optimal d'un multirotor . . . . .	27
2.4.3	Résultats en simulation . . . . .	31
2.5	Conclusion . . . . .	46
<b>3</b>	<b>Perturbations à l'intérieur d'un tuyau</b>	<b>51</b>
3.1	Introduction . . . . .	51
3.2	Travaux connexes . . . . .	54
3.3	Protocole de mesure et de traitement des perturbations dans un tuyau . . . . .	55
3.3.1	Collecte des données . . . . .	55
3.3.2	Traitement des données brutes collectées . . . . .	56
3.4	Analyses des perturbations observées dans le tuyau . . . . .	62
3.5	Apprentissage de modèles pour extrapoler les mesures . . . . .	65
3.5.1	Apprentissage avec des processus gaussiens . . . . .	65
3.5.2	Apprentissage avec des réseaux de neurones . . . . .	66
3.5.3	Transformation des mesures de perturbation pour faciliter l'apprentissage .	69
3.5.4	Performance des transformations des mesures . . . . .	70
3.5.5	Analyse des perturbations prédictes par le modèle appris . . . . .	76
3.6	Intégration du modèle des perturbations dans les contrôleurs basés contrôle optimal	79
3.7	Conclusion . . . . .	80

<b>4 Apprentissage par imitation du contrôle optimal</b>	<b>85</b>
4.1 Introduction . . . . .	85
4.1.1 Objectifs . . . . .	86
4.2 Travaux connexes . . . . .	86
4.3 Apprentissage par imitation . . . . .	88
4.4 Modèles et architecture des contrôleurs . . . . .	91
4.5 Résultats expérimentaux pour l'apprentissage d'un contrôleur de stabilisation . . . . .	94
4.6 Influence de la prise en compte de l'environnement . . . . .	99
4.7 Adaptation pour le suivi de trajectoire . . . . .	100
4.8 Conclusion . . . . .	110
<b>5 Estimation d'état d'un quadrotor volant à l'intérieur d'un tuyau</b>	<b>111</b>
5.1 Introduction . . . . .	111
5.2 Travaux connexes . . . . .	113
5.3 Capteurs . . . . .	114
5.4 Approche par régression linéaire . . . . .	115
5.5 Approche par modèle inverse via les mesures des télémètres . . . . .	122
5.6 Approche par filtre de Kalman . . . . .	126
5.6.1 Jacobienne de la dynamique $f$ . . . . .	128
5.6.2 Jacobienne du modèle de mesure $h$ . . . . .	129
5.7 Évaluation des estimations sur des données réelles . . . . .	132
5.8 Conclusion . . . . .	138
<b>6 Auto-organisation d'une flotte de drones pour explorer un tunnel</b>	<b>139</b>
6.1 Introduction . . . . .	139
6.2 Travaux connexes . . . . .	141
6.3 Formulation du problème de placement des robots . . . . .	142
6.3.1 Notations utiles . . . . .	143
6.3.2 Hypothèses . . . . .	143
6.3.3 Configurations optimales . . . . .	143
6.4 Algorithme de positionnement le long du tunnel . . . . .	147
6.4.1 Convergence vers la configuration optimale . . . . .	147
6.4.2 Estimation du RSSI par un filtre de Kalman . . . . .	149
6.4.3 Exploration avec une chaîne de drones . . . . .	151
6.5 Résultats . . . . .	152
6.5.1 En simulation . . . . .	152
6.5.2 Dans un environnement réel . . . . .	153
6.6 Discussion . . . . .	155
6.7 Conclusion . . . . .	157
<b>7 Discussion</b>	<b>159</b>
7.1 Mesure des perturbations dans un tuyau à l'aide d'un bras robotisé . . . . .	159
7.2 Choix du type de contrôleur . . . . .	161
7.3 Robustesse du contrôleur aux erreurs de modèle . . . . .	163
7.4 Approche par la modification de l'architecture du robot. . . . .	165
<b>Conclusion</b>	<b>167</b>

# Chapitre 1

## Introduction

### 1.1 Les robots pour l'exploration

Les robots, sous diverses formes, sont utilisés dans de nombreuses activités humaines. Construits initialement comme des curiosités ou des jouets, ils ont ensuite été introduits au sein des usines pour remplacer le travail humain dans les tâches simples et répétitives. C'est d'ailleurs de cet usage dont provient le mot robot.

Mais outre cet aspect d'automatisation du travail, l'utilisation de robots est une solution pour réaliser des tâches dans des environnements dans lesquels les humains ne peuvent pas, ou très difficilement, travailler parce que trop dangereux, trop coûteux, ou tout simplement inaccessible aux humains.

Des systèmes robotisés sont ainsi utilisés pour de nombreuses opérations de manipulation et de traitement des combustibles et déchets de l'industrie nucléaire afin de ne pas exposer leurs opérateurs humains à des niveaux de radioactivité trop élevés. Des robots sont également utilisés pour l'épandage de pesticides (FAİÇAL et al. 2014) ou pour réaliser le désamorçage de mines antipersonnelles. Leur usage pour la prise de vue ou même la conduite de frappes sur les champs de bataille s'est récemment illustré dans le conflit entre l'Azerbaïdjan et l'Arménie (2020) et à travers l'usage de missiles dit "intelligents".

Outre les environnements dangereux, certains robots sont utilisés en raison des coûts importants encourus par une présence humaine à bord d'un véhicule. Le cas le plus frappant est pour les véhicules volants en raison des coûts énergétiques importants de ce mode de transport et de l'impact de la masse d'un passager humain sur leur autonomie. Le coût bien plus faible de véhicules volant sans pilote a par exemple favorisé grandement l'usage de drones pour la prise de vue aérienne à la place d'avions. L'exploration des fonds marins (WHITCOMB et al. 2000) et de notre système solaire (HIRZINGER 1994 ; PEDERSEN et al. 2003) sont deux autres exemples de cas d'usage pour lesquels la présence d'un passager humain et du système de support vie associé font grandement augmenter les coûts et dimensions des véhicules en plus des risques pour les passagers. L'exploration spatiale, depuis la fin du programme Apollo, est le fait quasi exclusif de robots sous diverses formes.

Certains environnements sont tout simplement inaccessibles aux humains, en particulier à cause de leurs dimensions trop réduites. Les systèmes de distribution d'eau, de gaz ou d'air peuvent par exemple présenter des tuyaux dont les dimensions ne permettent pas à un humain de se mouvoir. L'exploration de certains sites archéologiques, la grande Pyramide de Kheops par exemple (GANTENBRINK 1999 ; RICHARDSON et al. 2013), peut aussi faire intervenir des environ-

nements nécessitant l'usage de robots.

C'est à de tels environnements que nous nous sommes intéressés durant cette thèse. L'exploration ou l'inspection de ces environnements étroits et confinés, dont l'accès direct est difficile pour les humains, pourrait être réalisée à l'aide de robots.

## 1.2 Diversité des formes des robots

Les usages des robots sont nombreux et divers et les formes et moyens de locomotion que prennent ces robots reflètent la diversité de ces usages. Les robots présents dans les usines sont en grande majorité dépourvus de capacités de locomotion et consistent en des outils divers montés au bout d'un bras composé de plusieurs segments articulés.

Comme pour nos véhicules, l'usage de roues et des systèmes assimilés (chenilles) est un choix qui présente beaucoup d'avantages pour les robots mobiles. Le contrôle et la direction de ces roues sont largement étudiés et développés comme conséquence de leur usage par de nombreux véhicules. De plus, les environnements dans lesquels nous évoluons sont souvent adaptés à l'usage des roues puisqu'ils ont été adaptés à l'usage de nos véhicules. Les robots se mouvant à l'aide de roues ont aussi été déployés sur des terrains difficiles. L'utilisation de nombreux robots à roues (rovers) pour l'exploration spatiale, Lunokhod et Yutu-2 sur la Lune, Sojourner, Opportunity, Curiosity, Perseverance et Zhurong sur Mars, témoigne de la capacité de la locomotion sur roues à s'accommoder de terrains très difficiles.

Une approche d'imitation des moyens de locomotions observés chez les animaux a poussé le développement de nombreux robots comportant des "pattes". L'usage de ces pattes, même s'il pourrait permettre une capacité à se mouvoir sur des terrains plus accidentés que ne le permettent les roues voire une capacité à grimper ou escalader certains obstacles, pose cependant des problèmes liés à l'équilibre du robot et à la coordination de l'action des différents membres. Certains de ces robots possèdent maintenant des capacités de mobilité importante comme l'illustrent les performances des robots bipèdes Atlas et du quadrupède Spot de Boston Dynamics, ou bien la capacité du robot quadrupède présenté dans les travaux de MIKI et al. 2022 (voir figure 1.1) à se mouvoir sur des sentiers de randonnée.

Sur la mer et sous la mer (STUTTERS et al. 2008), des robots exploitent les techniques de locomotion des véhicules marins, c'est-à-dire une coque flottante propulsée à l'aide d'hélices mues par des moteurs électriques ou thermiques. Les vastes distances des environnements maritimes et océaniques rendent cependant difficiles les usages autonomes de la locomotion par hélices et moteurs électriques.

Des robots utilisant la navigation à voile pour tirer parti de l'énergie éolienne parviennent à une meilleure autonomie dans ces environnements. SUN et al. 2022 présentent par exemple les principaux robots utilisant ce mode de locomotion et les différentes solutions mises en place pour les robots à voile.

Les robots se déplaçant dans les airs mettent en place de nombreuses techniques pour se mouvoir (MORIN et BIDAUD 2014). Les plus économies en énergie sont des aérostats, qui se maintiennent en l'air à l'aide de ballons plus légers que l'air et se déplacent à l'aide d'hélices. Ce mode de sustentation en fait des véhicules de taille importante (celle des ballons) et assez lents. Un avantage supplémentaire des ballons, outre la sustentation économique en énergie, est de fournir une bonne stabilité. Les travaux de GOMES et RAMOS 1998 présentent les équations de la dynamique d'un ballon.



FIGURE 1.1 – Le quadrupède ANYMal dans un escalier enneigé. Image tirée des travaux de MIKI et al. 2022. Les auteurs proposent une approche qui permet à ce robot de suivre des humains sur des sentiers de randonnée.

Les robots volants plus lourds que l'air dit “à voilure fixe”, c'est-à-dire des véhicules de la famille des avions, se maintiennent en l'air à l'aide d'effets aérodynamiques de portance provoqués par leurs ailes et sous conditions d'une vitesse horizontale importante. Cette contrainte sur la vitesse nécessite en particulier de grands espaces pour se mouvoir en particulier pour le décollage et l'atterrissage. Ce mode de déplacement est aussi plus coûteux en énergie que celui des ballons. On peut trouver dans l'ouvrage de VALAVANIS et VACHTSEVANOS 2014 une description du fonctionnement de ce type de robots.

Une autre catégorie de robots plus lourds que l'air existe. Ces robots possèdent une voilure dite tournante, ce qui correspond à des rotors dont la rotation permet la sustentation du robot. Ce mode de déplacement permet en particulier le vol stationnaire et de manœuvrer et de parcourir des distances bien plus courtes. La sous-catégorie la plus connue de ces robots à voilure tournante sont les hélicoptères qui possèdent en général un rotor principal pour produire la force de sustentation principale et un rotor de queue pour stabiliser l'appareil en lacet. ABBEEL, COATES et NG 2010 présentent par exemple un contrôleur pour la réalisation de trajectoires acrobatiques sur un hélicoptère. Cette architecture est très répandue dans les véhicules embarquant des pilotes humains, mais moins courante pour les robots. Ceci est dû à une complexité importante du système mécanique des rotors, difficilement miniaturisable. Un hélicoptère, Ingenuity WITHROW et al. 2020 a récemment été déployé sur mars conjointement au rover Perseverance. La figure 1.2 présente cet hélicoptère sur mars.

Les multirotors sont une sous-catégorie de robots volants à voilure tournante plus courante dont la mécanique et l'agencement des rotors est particulièrement simple. Ces robots possèdent en général quatre rotors ou plus. Contrairement aux hélicoptères pour lesquels l'angle d'attaque des hélices est également variable, les multirotors présentent habituellement davantage de rotors dont le seul degré de liberté est la vitesse de rotation de l'hélice. Cette simplicité mécanique et l'usage de moteurs électriques rend les architectures multirotors bien plus courantes pour les robots dont les dimensions sont réduites.

La plupart des multirotors placent leurs rotors dans un même plan, c'est-à-dire que les axes de rotation de tous ces rotors sont alignés. Une telle architecture ne possède que 4 degrés de



FIGURE 1.2 – L'hélicoptère Ingenuity sur Mars. Photo prise par le rover Perseverance le 6 avril 2021. Source NASA-JPL ([https://mars.nasa.gov/mars2020/multimedia/raw-images/SI1\\_0046\\_0671022109\\_238ECM\\_N0031416SRLC07021\\_000085J](https://mars.nasa.gov/mars2020/multimedia/raw-images/SI1_0046_0671022109_238ECM_N0031416SRLC07021_000085J)).

liberté : un degré de déplacement parallèlement à l'axe des rotors et 3 degrés de rotation du robot. À l'aide des rotations du robot, et donc de l'axe de ses rotors, il est bien entendu possible de faire se déplacer le robot dans toutes les directions, mais ces degrés de liberté sont dépendants de l'inclinaison du robot.

Par opposition à ces architectures à 4 degrés de liberté, des architectures dites “complètement actionnées” ont été développées et permettent un déplacement du robot selon les 3 axes indépendamment de la rotation. Ceci est rendu possible par un agencement des axes de rotation des rotors. Les travaux de RAJAPPA et al. 2015 ; BICEGO et al. 2020 ; FRANCHI et al. 2018 détaillent un modèle de la dynamique de ce type d’architecture. Cet agencement des rotors est moins efficace d'un point de vue énergétique que les architectures coplanaires, mais permet plus d'indépendance dans le contrôle de la position et de l'inclinaison du robot.

Contrairement aux robots volants à voilure fixe, les robots à voilure tournante peuvent plus facilement naviguer dans des environnements étroits comme les tuyaux auxquels nous avons choisi de nous intéresser. Les dimensions des robots volants avec ballons peuvent poser problème dans des environnements dont les dimensions peuvent prévenir le passage de robots trop volumineux dans certains passages très étroits. Les robots à roues sont généralement capables de se déplacer dans les environnements de ce type qui présentent une surface au sol plate et dégagée. Dans le cas contraire, en présence d'échelles, d'escaliers ou d'une pente très forte voire de conduits verticaux, ceux-ci ont des difficultés importantes. En intérieur, le contournement de tels obstacles peut être bien plus difficile qu'en extérieur, voire impossible. En outre, les robots volants sont capables de se déplacer à des vitesses plus importantes que les robots à roues ce qui est un avantage pour l'exploration d'environnement potentiellement très étendus, comme peut l'être un réseau de tuyaux d'aération par exemple.

### 1.3 Approches pour la commande des robots

Pour réaliser les tâches qu'on attend des robots, il est nécessaire de leur appliquer des commandes appropriées sur leurs actionneurs. Ces commandes peuvent permettre le fonctionnement d'un robot pleinement autonome ou stabiliser un sous-système dont le temps caractéristique est trop réduit pour être directement contrôlé par un opérateur humain. De nombreuses approches existent pour décider automatiquement des commandes à appliquer à un robot.

Une première distinction importante entre ces approches concerne le retour d'état. Les commandes sans retour d'état, dites "en boucle ouverte", sont des commandes qui sont calculées à priori, ou hors ligne, et qui sont ensuite appliquées à un robot sans tenir compte de l'évolution de l'état de ce robot. Le fait que ces commandes puissent être calculées au préalable permet de réaliser des calculs coûteux qu'il serait impossible de réaliser en temps réel. L'absence de boucle de rétroaction empêche la correction d'un comportement qui n'a pas été prévu au moment du calcul de la commande. En conséquence, ce type de commande ne peut être appliqué qu'à des robots dont la dynamique est connue de manière extrêmement fine et dont l'état initial peut être mesuré de manière très précise. Un petit écart au comportement attendu peut effet s'amplifier de manière exponentielle en l'absence de correction. Ces hypothèses sont difficiles à assurer dans le cas de véhicules robotisés se déplaçant dans un environnement turbulent.

Contrairement à la commande en "boucle ouverte", la commande en "en boucle fermée" ou avec retour d'état est calculée en temps réel et prend en compte des mesures de capteurs ou des estimations de son état par le robot. Celui-ci peut en conséquence compenser ou corriger une inexacititude dans le modèle de la dynamique, ou une perturbation inattendue. Ce retour d'état introduit cependant une interaction entre la dynamique et la commande. Si la loi de commande est inadaptée à la dynamique commandée, cette interaction entre le contrôleur et la dynamique peut mener à une instabilité très importante du robot. « Introduction to Feedback Control of Underactuated VTOL vehicles » 2013 présentent une introduction à cette classe de commandes pour des robots à décollage vertical comme les multirotors.

La commande de robots dont la dynamique est linéaire peut s'appuyer sur une importante théorie mathématique qui permet de caractériser la stabilité et la contrôlabilité de ces robots et permet ainsi la conception de contrôleurs possédant des propriétés de stabilité. Grâce à une dynamique linéaire, les trajectoires et commandes peuvent en être transportées dans le domaine fréquentiel pour lequel l'analyse des propriétés de stabilité est plus aisée. Pour les systèmes possédant une dynamique linéaire, il est possible de déterminer automatiquement des gains pour des régulateurs PID via une analyse dans le domaine fréquentiel ou de déterminer une solution analytique au problème de contrôle optimal pour un coût quadratique comme l'illustre le contrôleur LQR (TEDRAKE 2021a). Ce contrôleur LQR est un contrôleur linéaire qui a la propriété de minimiser un coût quadratique et possède une expression analytique en fonction de la matrice de ce coût et de la matrice de la dynamique du système. L'hypothèse d'une dynamique linéaire offre de nombreuses solutions pour la conception de contrôleurs, mais les systèmes possédant une dynamique linéaire sont peu courants, que ce soit à cause de l'existence de contraintes sur certains états et de limites aux actionneurs ou bien à cause de phénomènes physiques non linéaires, comme les contacts, les chocs et les perturbations aérodynamiques.

De nombreux robots ne présentent pas une dynamique non linéaire. C'est en particulier le cas des quadrotor qui ne sont pas complètement actionnés et pour lesquels des changements d'inclinaison

sont nécessaires pour se déplacer latéralement. Ces changements d'inclinaison sont hautement non linéaires. De plus, les rotors utilisés présentent des limites hautes et basses aux forces qu'ils peuvent créer.

Dans certains cas les outils de contrôle des systèmes linéaires peuvent être adaptés. Une première approche consiste à approximer la dynamique du robot au premier ordre par une dynamique linéaire autour d'un ou plusieurs points de fonctionnement. BELKHEIRI et al. 2012 présentent un modèle de quadrotor linéarisé autour d'un état d'équilibre pour ensuite appliquer un contrôleur LQR à cette approximation linéaire de la dynamique non linéaire du robot. Un contrôleur LQR itéré (W. LI et TODOROV 2004) itère par exemple des linéarisations au premier ordre de la dynamique autour des états rencontrés durant les trajectoires planifiées.

Une autre approche consiste à introduire une transformation non linéaire de l'état et une variable de contrôle auxiliaire de manière à ce que la dynamique du robot exprimée à l'aide de ces nouvelles variables soit linéaire. Il s'agit d'une approche dite de "feedback linearization". Une fois la transformation réalisée, il est possible d'utiliser des outils du contrôle linéaire. RAJAPPA et al. 2015 utilisent par exemple une telle approche pour le contrôle d'un hexarotor complètement actionné en simulation. CHEMORI et MARCHAND 2008 l'appliquent à un avion à décollage vertical.

Une troisième approche utilisant les outils de la commande linéaire est appelée "sliding mode control" et consiste à dériver des équations de la dynamique dans un sous-espace d'états dans lequel la dynamique du robot est linéaire et converge (glisse) vers un état désiré, puis de concevoir un contrôle qui constraint le robot à rester dans ce sous-espace. WASLANDER et al. 2005 présentent un tel contrôleur pour un quadrotor et l'appliquer sur un quadrotor STARMAC. La commande appliquée pour maintenir le robot dans le sous-espace d'états où il possède une dynamique linéaire est habituellement discontinue ou avec un gain très important ce qui peut avoir comme conséquence des effets d'oscillation importants autour de ce sous-espace.

L'extension de l'usage des outils du contrôle linéaire n'est cependant pas utilisable systématiquement. La présence de contraintes dans la dynamique, la commande ou sur les états n'est par exemple traitée par les modèles qui utilisent les outils du contrôle linéaire qu'à travers la fonction de coût quadratique des contrôleurs LQR et ne fait que pénaliser le non-respect de ces contraintes plutôt que d'imposer que celles-ci soient respectées. Ces limites peuvent être traitées par des approches pour le contrôle qui n'utilisent pas les outils du contrôle linéaire.

Une approche n'utilisant pas directement les outils du contrôle linéaire, le "backstepping control" consiste à découper la dynamique du robot en une hiérarchie de sous-systèmes de manière à ce que l'état de chaque sous-système soit contrôlé par le système en dessous et contrôle le système au-dessus. Le contrôle de la vitesse d'un robot permet par exemple le contrôle de sa position. LEE, LEOK et MCCLAMROCH 2010 découpent par exemple le problème de contrôle de la position d'un quadrotor en un problème de contrôle de sa vitesse angulaire qui permet le contrôle de son orientation qui permet le contrôle de sa vitesse qui permet le contrôle de sa position. Une telle approche demande cependant que la vitesse de convergence d'un sous-système A soit grande devant la fréquence de commande du système B contrôlé par le sous-système A. Ceci peut limiter la réactivité de contrôleurs ayant une hiérarchie trop profonde.

Une approche plus générique pour le contrôle de robots consiste à formuler la commande comme la solution d'un problème d'optimisation d'une commande ou d'une trajectoire sous contrainte de respect de la dynamique du robot. Ce problème d'optimisation est appelé problème de contrôle optimal. Cette approche nécessite un modèle de la dynamique du robot qui puisse formuler celle-ci sous une forme compatible avec la méthode ou le solveur utilisé pour résoudre le problème de

contrôle optimal. Une telle approche génère une commande sans retour d'état.

Une approche dite de *commande prédictive* (Model Predictive Control - MPC) permet de refermer la boucle et d'inclure un retour d'état en n'exécutant que la première commande de la trajectoire solution et à résoudre à nouveau le problème d'optimisation à partir du nouvel état courant. CARLOS et al. 2020 présentent un exemple de ce type de contrôleur pour un quadrotor Crazyflie. La résolution du problème de contrôle optimal présente souvent un coût en calcul important. La résolution d'un tel problème à chaque pas de temps dans le cadre d'une commande prédictive nécessite donc de disposer de capacités de calcul en temps réel importantes et conduit parfois à réduire la fréquence à laquelle est réalisée la commande prédictive.

Les techniques d'apprentissage automatique permettent aussi la conception de contrôleurs pour les robots. Ces techniques réalisent l'exploration d'un espace de contrôleurs générique afin d'en trouver un qui soit satisfaisant. Contrairement aux approches présentées plus tôt, l'approche par apprentissage ne fait pas nécessairement d'hypothèses sur la forme et la manière de contrôler un système. Il est possible d'utiliser l'apprentissage pour concevoir des contrôleurs linéaires, des contrôleurs en boucle ouverte ou bien des contrôleurs non linéaires ou à retour d'état. Les réseaux de neurones sont par exemple une architecture souvent utilisée conjointement à l'apprentissage en raison de leur capacité à approcher n'importe quelle fonction. Dans le cas de contrôleurs linéaires, la recherche a lieu dans l'ensemble des matrices de dimensions pertinentes, pour les réseaux de neurones, paramétrés par le poids de connexions neuronales, elle a lieu dans l'espace de ces paramètres. Ces deux classes de contrôleurs sont des exemples de classes de contrôleurs génériques dans lesquelles les techniques d'apprentissage peuvent rechercher un contrôleur adapté à un robot et une tâche.

L'apprentissage supervisé permet en particulier de formuler des contrôleurs dont le comportement imite et généralise des exemples de commandes fournies par un expert, qu'il s'agisse d'un pilote ou d'un contrôleur d'un autre type. ABBEEL, COATES et NG 2010 présentent par exemple un algorithme pour l'apprentissage d'un contrôleur capable de reproduire des manœuvres acrobatiques sur un hélicoptère à partir d'exemples de ces manœuvres réalisées par un pilote humain. L'apprentissage par imitation nécessite cependant de savoir réaliser ou calculer des exemples de commandes pertinentes couvrant toutes les situations que devra rencontrer le robot. Dans de nombreux cas, il est plus intéressant et plus fiable d'utiliser directement la méthode qui permet d'obtenir des exemples de commandes pertinentes directement sur le robot plutôt que d'apprendre à imiter celle-ci.

L'apprentissage par renforcement entend traiter les cas où aucun exemple de commande pertinente n'existe ou aucune commande pertinente n'est facilement calculable. Cette approche cherche un contrôleur générique qui optimise un critère, coût ou récompense, spécifié par avance. L'ambition de telles approches est de pouvoir chercher ce contrôleur uniquement à l'aide des observations récoltées par le robot, c'est-à-dire les quadruplets (état, action, récompense, état suivant) observés durant le fonctionnement.

L'apprentissage par renforcement se distingue ici de la commande optimale en ce qu'il n'utilise pas directement de modèle de la dynamique connue au préalable, mais uniquement des observations de la manière dont fonctionne le robot. Certains algorithmes d'apprentissage par renforcement apprennent cependant en parallèle un modèle de la dynamique à partir des observations (CHUA et al. 2018; CHATZILYGEROUDIS et al. 2020). D'autres apprennent comme "modèle" non pas la dynamique du robot, mais de la récompense à venir, MOLCHANOV et al. 2019 utilisent par exemple l'algorithme d'apprentissage par renforcement PPO (SCHULMAN et al. 2017) pour apprendre un

contrôleur pour un quadrotor. Les auteurs testent leur approche sur le drone Crazyflie.

L'approche par renforcement demande d'importantes quantités d'observations et présente une variabilité importante : la qualité des contrôleurs appris peut varier beaucoup pour des conditions d'apprentissage identiques. Elle est difficile à appliquer directement sur des robots en raison du temps nécessaire pour collecter la quantité d'observations nécessaire en faisant fonctionner un véritable robot. Cette approche est plus souvent appliquée en simulation où il est possible de collecter en parallèle l'importante quantité d'observations nécessaires. Dans le cas d'un apprentissage à partir d'un simulateur, le transfert du contrôleur, appris en simulation, sur un robot, dont le fonctionnement ne correspond pas parfaitement à celui du simulateur, peut demander des efforts importants.

Dans cette thèse nous avons décidé de nous concentrer sur des contrôleurs obtenus par résolution de problèmes de contrôle optimal. Ceux-ci permettent d'intégrer, de manière générique, des modèles non linéaires de la dynamique des robots, mais aussi des modèles non linéaires d'interactions avec l'environnement et faire respecter des contraintes liées aussi bien à cet environnement qu'à la dynamique des robots.

Les robots à voilure tournante possèdent en effet une dynamique fortement non linéaire. Ceux qui ne présentent pas un actionnement complet doivent changer leur inclinaison pour se mouvoir horizontalement, et ce changement d'inclinaison est intrinsèquement non linéaire. De plus, les limites d'actionnement des rotors et les perturbations aérodynamiques limitent aussi l'usage d'outils du contrôle linéaire.

L'utilisation d'un quadrotor dans un environnement comme un tuyau motive la recherche d'une autonomie dans le domaine du contrôle : les communications avec des moyens de calcul externes sont difficiles dans un environnement intérieur. De plus, les faibles dimensions du Crazyflie ne permettent pas d'embarquer des capacités de calcul importantes, rendant difficile la résolution de problèmes de contrôle optimal directement à bord et en temps réel. Nous avons donc choisi d'approximer par imitation une commande basée sur la résolution de problèmes de contrôle optimal.

## 1.4 Objectifs

Les environnements intérieurs confinés ou étroits comme les conduits d'aération, les tunnels ou les tuyaux sont omniprésents dans les environnements urbains ou industriels. Ceux-ci sont très difficiles d'accès aux humains en raison de leurs dimensions réduites. Leur exploration ou leur inspection pourrait donc être réalisée à l'aide de robots autonomes ou télé opérés qui peuvent être conçus avec des dimensions adaptées à ces environnements étroits.

Les roues ou les chenilles constituent des modalités de déplacement très fiables pour des robots, mais c'est une approche qui s'accorde mal d'obstacles présents au sol, ou de fortes pentes. C'est aussi une modalité de déplacement peu rapide, ce qui est mal adapté au parcours de vastes réseaux de tunnels ou de tuyaux.

L'utilisation de robots volants permet de s'affranchir des difficultés liées à des obstacles au sol ou à de fortes pentes.

La figure 1.3 illustre un exemple d'usage applicatif pour l'exploration d'environnements internes avec un quadrotor de petite taille. Les robots à voilure fixe demandent de grands espaces pour manœuvrer, leur usage n'est donc pas bien adapté à des environnements étroits et confinés

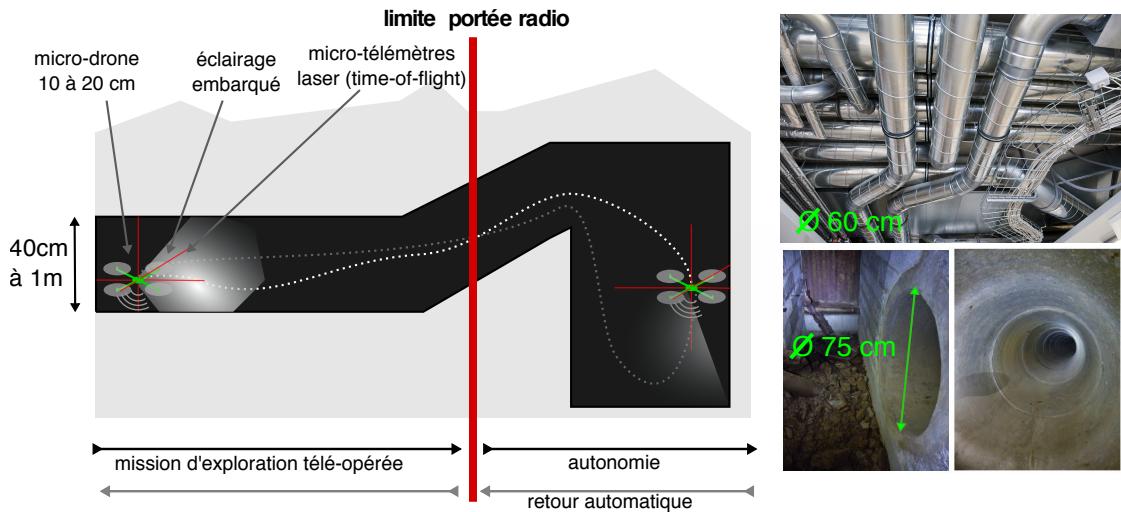


FIGURE 1.3 – (À gauche) Exemple d'un scénario d'utilisation d'un quadrotor pour l'exploration semi-autonomie d'un environnement souterrain. Les dimensions réduites du drone lui permettent le passage dans des environnements aux dimensions très réduites et le mode de déplacement permet de se déplacer dans des environnements dont le sol présente des obstacles ou des pentes importantes. Ce type de robot peut aussi se déplacer dans les trois dimensions de l'espace. (À droite) Exemple d'environnement pour lequel l'usage de drones pourrait être particulièrement adapté en raison de ses dimensions.

comme ceux auxquels nous nous intéressons. Les aérostats nécessitent des ballons dont les dimensions sont importantes et leur usage dans ces environnements est aussi limité. Dans la famille des robots à voilure tournante, les quadrotors présentent une architecture classique, très répandue et dont il est plus aisément de miniaturiser la mécanique que les hélicoptères.

Mais les quadrotors sont des robots instables et particulièrement sensibles aux collisions avec l'environnement qui provoquent aisément des crashes. De plus, leur modalité de déplacement à l'aide de rotors produit un important écoulement de l'air autour d'eux dans les environnements dans lesquels ils évoluent. Cet écoulement interagit de manière importante avec les parois d'environnements confinés et étroits comme les tuyaux. Ces turbulences dans l'air créées par le robot sont la cause de perturbations de la stabilité du robot. Ce phénomène de perturbations aérodynamiques causées par la présence du robot pourrait aussi être présent dans les environnements intérieurs étroits, comme des sites archéologiques, dont la géométrie n'est pas forcément aussi régulière et dans lesquels peuvent se trouver des obstacles au sol difficiles à surmonter pour des robots à roues.

Nous nous concentrerons dans cette thèse sur l'utilisation de quadrotor de petite taille et en particulier du quadrotor Crazyflie qui a été choisi pour son matériel et son logiciel ouverts et bien documentés ainsi que pour ses dimensions très réduites (environ 10cm de diamètre, 30g). Ces dimensions réduites permettent un vol dans des environnements très étroits. Elles permettent aussi de faire voler ce robot sans craindre des risques de dégâts sur l'environnement, le robot ou les expérimentateurs en cas de collision. Ceci rend plus aisée l'expérimentation en intérieur, dans des environnements étroits où les risques de collision sont très présents.

Les perturbations du robot par sa propre présence, son instabilité et sa sensibilité aux collisions rendent cruciale l'utilisation d'une commande adaptée qui permet de limiter, prévoir ou de rejeter ces turbulences et d'éviter les collisions. Les quadrotors sont des robots dont la dynamique est

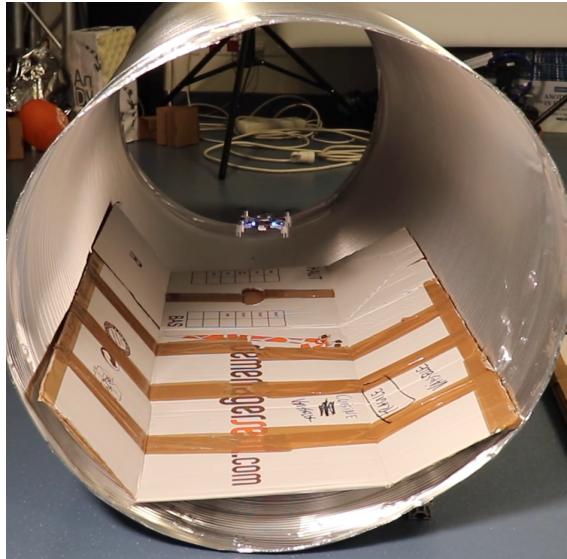


FIGURE 1.4 – Un quadrotor Crazyflie dans un tuyau de 60cm de diamètre.

hautement non linéaire dont les actionneurs présentent des saturations. En outre, l'évitement des collisions avec les parois de l'environnement introduit une contrainte qui nous fait écarter l'usage de contrôleurs linéaires ou utilisant les outils du contrôle linéaire de manière approchée. L'approche d'apprentissage par renforcement est sujette à des problèmes de convergence et de sensibilité élevées aux hyperparamètres et nous avons donc choisi de nous concentrer plutôt sur une commande prédictive basée sur la résolution de problèmes de contrôle optimal. Cette approche permet la prise en compte de contraintes de l'environnement, une dynamique non linéaire, une planification à court terme et le rejet de perturbations non prévues. C'est donc une approche adaptée aux environnements et robots auxquels nous nous intéressons.

La résolution répétée des multiples problèmes de contrôle optimal est cependant trop coûteuse en calcul pour être exécutée en temps réel à bord des petits robots comme le Crazyflie. Pour rendre cette approche plus accessible sur l'aspect du coût en calcul, nous avons opté pour une approximation de cette approche. La commande optimale permet de générer de manière automatisée un grand nombre d'exemples de commandes pertinentes. Cette propriété permet, au prix de nombreux calcul réalisés hors ligne, de s'appuyer sur l'apprentissage supervisé pour apprendre, par imitation, une loi de commande qui est une approximation de cette commande optimale. Les algorithmes d'apprentissage supervisé possèdent une meilleure stabilité que l'apprentissage par renforcement, pour peu que de nombreux exemples d'apprentissage soient disponibles. La commande optimale permet la génération de ces exemples.

Cette thèse a donc pour objectif applicatif le vol d'un quadrotor Crazyflie dans des environnements de type tuyau d'aération comme illustré sur la figure 1.4. Cet objectif applicatif est motivé par les avantages que présentent les drones pour l'exploration d'environnements composés de tuyaux : leur vitesse de déplacement relativement importante par rapport aux robots à roues et leur capacité à passer outre des obstacles au sol ou des tuyaux orientés à la verticale, voire des escaliers ou des échelles.

À notre connaissance, ce cas d'usage n'est traité que par la compagnie Flyability (<https://www.flyability.com/>). Cette compagnie propose des services d'inspection d'environnements confinés (tuyaux, cuves, mines, glaciers, égouts, etc) à l'aide d'un drone. Leur approche utilise un quadrotor (Elios 2) aux dimensions plus importantes que le Crazyflie (40cm, 1.4kg) muni d'une cage de protection qui, plutôt que d'éviter les collisions, absorbe les chocs pour éviter les crashes. Cette approche est détaillée dans les travaux de BRIOD, KŁAPTOCZ et al. 2012 ; BRIOD, KORNATOWSKI et al. 2014. Les dimensions du matériel et des environnements dans ces travaux sont plus importantes que celles que nous abordons dans ce travail. Ce changement d'échelle signifie des contraintes sur le matériel plus importantes ainsi que des capacités d'actionnement moindres sur un quadrotor aux dimensions du Crazyflie en comparaison de l'Elios 2. Un Crazyflie serait par exemple incapable de supporter le poids d'une cage de protection (cette cage a une masse de près de 400g dans le travail de BRIOD, KORNATOWSKI et al. 2014, soit plus de 10 fois la masse du Crazyflie).

L'usage de drones dans de telles conditions présente des difficultés importantes que cette thèse aborde. La première concerne l'existence de perturbations aérodynamiques causées par le mode de déplacement des quadrotors, et la seconde la nécessité d'exercer un contrôle précis du robot pour éviter les collisions avec l'environnement quand celui-ci a des dimensions réduites.

Pour répondre à ces difficultés, cette thèse nous abordons 3 questions principales :

- Quelles sont les perturbations que rencontre un quadrotor dans un tuyau ?
- Comment contrôler précisément un quadrotor soumis à des perturbations dans un tuyau ?
- Comment embarquer cette approche de contrôle à bord de drones dont les capacités de calcul sont réduites comme le Crazyflie ?

Cette thèse a en partie été réalisée dans le cadre du projet ANR Proxilearn.

## 1.5 Contribution

Pour répondre aux questions abordées par cette thèse, nous présentons les contributions suivantes :

1. un dispositif et un protocole de mesures couplées d'un traitement des perturbations créées par un drone dans un tuyau ;
2. l'apprentissage à partir de données réelles d'un modèle de ces perturbations capable de réaliser des prédictions des perturbations observées en n'importe quel point du tunnel ;
3. la formulation d'une commande prédictive pour un multirotor à l'aide de la résolution de problèmes de contrôle optimal intégrant le modèle des perturbations ;
4. l'adaptation de l'algorithme DAGGER pour l'apprentissage par imitation de cette commande prédictive par un réseau de neurones capable de fonctionner à bord d'un Crazyflie ;
5. la formulation de trois approches pour estimer la position d'un drone via les spécificités de la géométrie d'environnement tuyau ;
6. un algorithme distribué pour positionner une flotte de drones dans un tunnel de manière à maintenir un réseau de communications dans un environnement où la propagation radio est limitée.

La figure 1.5 détaille l'articulation des principales contributions entre elles.

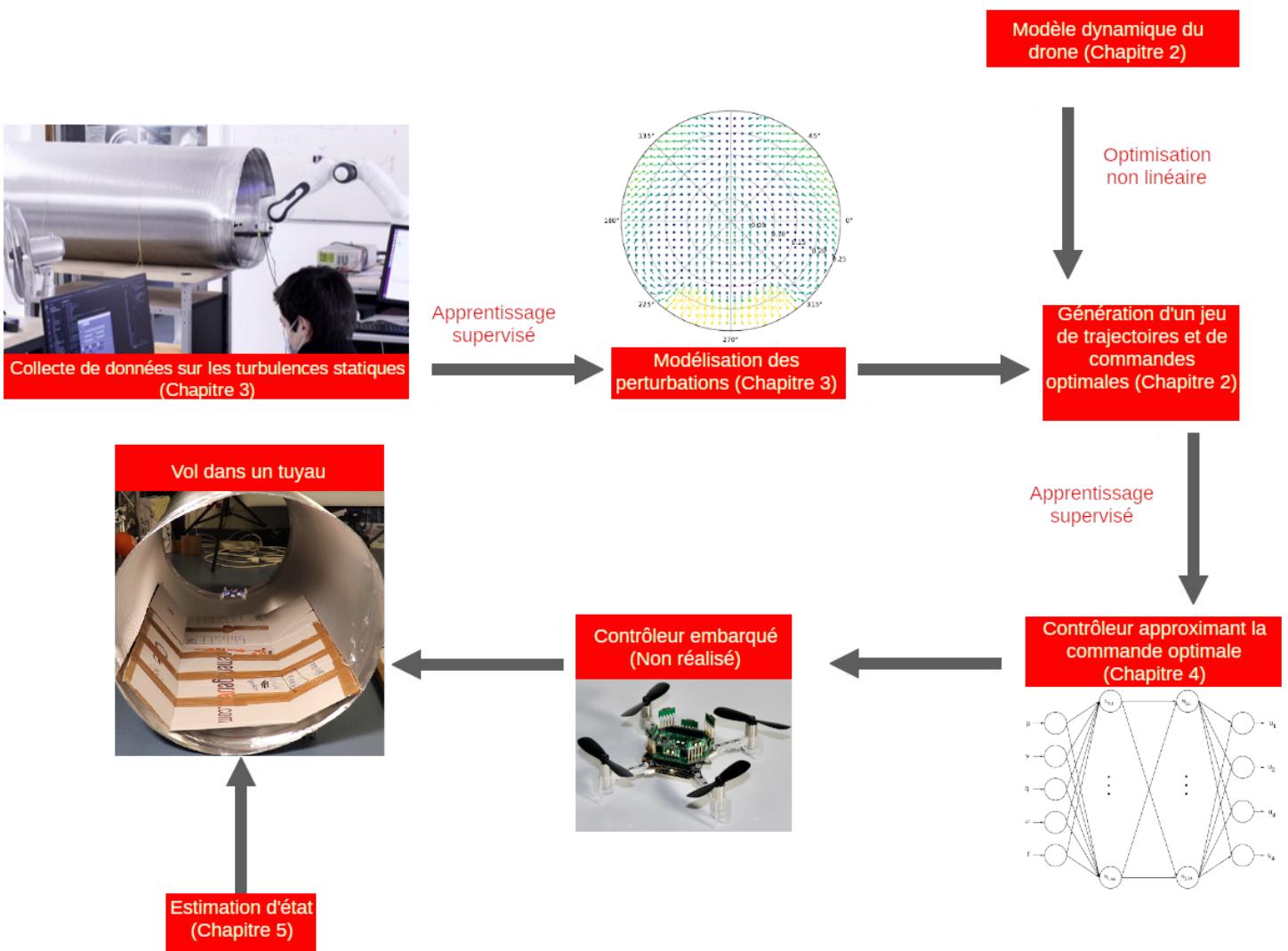


FIGURE 1.5 – Articulation des différentes contributions de cette thèse. Le contrôleur appris n'a pas pu être embarqué à bord du Crazyflie, mais des tests préliminaires ont confirmé des temps d'exécution suffisamment faibles. L'illustration du drone volant dans un tuyau a été faite avec un pilotage manuel.

## Plan

Le reste de cette thèse s'organise de la manière suivante :

- Le premier chapitre détaille les équations retenues pour modéliser la dynamique des multirotors et en particulier des quadrotors ainsi que les caractéristiques du quadrotor Crazyflie sur lequel cette thèse se concentre, les équations formulant le problème de contrôle optimal pour des multirotors et une approche pour sa résolution.
- Le second chapitre présente l'environnement du tuyau, ses caractéristiques, un dispositif pour collecter des données sur les perturbations et un protocole pour traiter ces données ainsi que des modèles pour réaliser des prédictions de ces perturbations, appris à partir des données collectées.
- Le troisième chapitre présente l'approche choisie pour réduire le coût en calcul d'un contrôle optimal, c'est-à-dire un algorithme d'apprentissage par imitation de ce contrôle optimal.
- Le quatrième chapitre présente trois approches pour estimer l'état du robot en tenant compte de la spécificité de l'environnement qu'est le tuyau et à l'aide des capteurs embarqués par un Crazyflie.
- Le cinquième chapitre présente un algorithme pour la coordination d'une flotte de quadrotors dans un tunnel pour en réaliser l'exploration tout en maintenant un contact radio avec l'entrée de ce tunnel. Cet algorithme exploite la qualité du lien radio entre les différents quadrotors afin d'ajuster de manière décentralisée la distance entre ceux-ci de manière à utiliser un nombre minimal de robots pour explorer jusqu'à une profondeur donnée.



## Chapitre 2

# Modélisation et contrôle de multirotors

### 2.1 Introduction

#### Contexte

Les multirotors, et en particulier les quadrotors sont des robots très répandus, et les modèles décrivant leur dynamique sont en conséquence très divers. De nombreuses conventions pour la représentation de l'architecture, de l'orientation ou des commandes existent.

Dans cette section je formalise les équations et notations d'un modèle qui sera utilisé pour le reste de cette thèse sous l'hypothèse d'un robot indéformable et dont le régime de fonctionnement (faible vitesse) permet de négliger les frottements.

Je veille en particulier à ce que la formulation choisie puisse modéliser correctement la dynamique du Crazyflie (GIERNACKI et al. 2017) et intégrer les estimations des paramètres réalisées par les travaux de FÖRSTER 2015 ; LANDRY et al. 2016 ; ANTONSSON 2015. La figure 2.1 présente un exemplaire d'un Crazyflie.

Nous avons choisi d'utiliser ce quadrotor en raison du caractère ouvert de son code et de son matériel. Cette ouverture et son utilisation dans de nombreux laboratoires permettent de faire émerger les informations sur les caractéristiques et le comportement de ce robot et permettent le développement et le support de nombreux outils compatibles.

Les multirotors sont des robots dont la dynamique est non linéaire et instable. En plus d'une connaissance de la dynamique de ces robots, le choix d'un contrôleur adapté est crucial pour réaliser des vols satisfaisants.

Les approches utilisées pour le contrôle des multirotors sont nombreuses et sont souvent déterminées en lien avec la manière dont est modélisée la dynamique. Nous avons choisi de nous intéresser à une approche de type commande prédictive (Model Predictive Control, MPC) calculée à partir de la résolution d'un problème de contrôle optimal. Cette approche permet en effet d'allier un contrôle en boucle fermée, une capacité de planification, et l'usage explicite d'un modèle de la dynamique et de l'environnement via la formulation d'un problème d'optimisation intégrant cette dynamique. Ceci permet une formulation qui découpe la spécification du contrôle du modèle dynamique utilisé pour la planification et permet ainsi d'appliquer un même contrôleur à de nombreux robots en changeant simplement le modèle dynamique. Cette approche a pour inconvénient



FIGURE 2.1 – Le petit quadrotor Crazyflie de Bitcraze

un coût important en calcul.

## Objectifs

Dans ce chapitre l'objectif est de formuler les notations et équations décrivant la dynamique des multirotors, de définir le problème de contrôle optimal permettant la formulation d'une commande prédictive puis d'évaluer la pertinence de cette commande en simulation sur des tâches de stabilisation et de suivi de trajectoire.

## Plan

Dans le reste de ce chapitre je présente donc pour commencer plusieurs travaux traitants des architectures et de la modélisation des multirotors, du quadrotor Crazyflie et de quelques approches pour le contrôle des multirotors.

Je détaille ensuite les équations et notations choisies pour la dynamique des multirotors et les caractéristiques du Crazyflie.

Je termine par une présentation du problème du contrôle optimal et de la manière dont on peut se servir de la résolution de ce problème pour réaliser une commande prédictive. Je présente une implémentation de cette commande prédictive et ses performances en simulation. Pour mettre en perspective cette commande, je présente une comparaison avec un contrôle géométrique adapté de travaux existants.

## 2.2 Travaux connexes

### Modèles de quadrotors

Les multirotors sont des robots régulièrement étudiés dans la littérature scientifique et de nombreuses modélisations sont proposées, à différents niveaux de fidélité. RAJAPPA et al. 2015

décrivent un modèle de multirotor générique qui permet de décrire aussi bien les classiques quadrotors que d'autres architectures moins répandues. FAESSLER, FRANCHI et SCARAMUZZA 2018 décrivent un modèle pour les quadrotors, et y intègrent un effet de traînée lié à la vitesse afin de réaliser un meilleur contrôle pour des trajectoires à vitesse élevée. MANECY 2015 présente un modèle plus complet encore incluant une description de l'effet de flapping et son impact sur les forces et moments qui s'exercent sur un quadrotor.

## Multirotors complètement actionnés

Il existe maintenant des architectures de multirotors permettant un actionnement complet des robots et des translations sur les trois axes indépendamment des rotations. RAJAPPA et al. 2015 présentent une de ces architectures et abordent la manière d'optimiser l'orientation des moteurs ainsi qu'un contrôleur adapté à ces architectures. L'intérêt d'une telle architecture pour sa capacité à rejeter plus efficacement, plus rapidement et de manière plus fiable des perturbations externes est abordé par JIANG et VOYLES 2014. Cette capacité à rejeter des perturbations externes permet en particulier aux multirotors de conduire des interactions physiques avec leur environnement. RYLL et al. 2017 présentent un cas d'usage d'usage d'interactions avec un humain via une corde. JIANG et VOYLES 2013 présentent un second cas d'usage pour l'inspection de bâtiments nécessitant un contact physique avec ceux-ci. RASHAD et al. 2020 réalisent une revue de la littérature sur les architectures, contrôles et usages de ces multirotors complètement actionnés.

## Le quadrotor Crazyflie

Dans cette thèse nous travaillons avec le robot Crazyflie, un quadrotor entièrement open source produit par Bitcraze. En raison de son caractère ouvert, de nombreux travaux s'appuient sur ce robot. FÖRSTER 2015 décrit des protocoles de mesure pour l'identification des paramètres du Crazyflie comme la matrice d'inertie, les coefficients de traînée ou le comportement des moteurs (dynamique d'évolution, lien entre poussée et vitesse de rotation, et commande appliquée). GREIFF 2017 décrit aussi le comportement des moteurs du Crazyflie. BUDACIU et al. 2019 décrivent en détail l'architecture du contrôleur embarqué à bord du Crazyflie par Bitcraze. Pour rendre l'expérimentation avec ce robot plus accessible, SILANO et IANNELLI 2020 ont réalisé un simulateur de Crazyflie dans le moteur de simulation Gazebo permettant une communication via ROS. La petite taille de ce quadrotor le rend particulièrement adapté à une utilisation en intérieur. LANDRY et al. 2016 réalisent un algorithme pour la planification de trajectoire dans un environnement intérieur encombré qu'ils embarquent à bord d'un Crazyflie. CARLOS et al. 2020 conçoivent un contrôleur à commande prédictive non linéaire (non linear model predictive control) déporté sur une machine externe qui transmet des consignes d'orientation par radio au contrôleur embarqué d'un Crazyflie. BANSAL et al. 2016 apprennent un réseau de neurones modélisant la dynamique du Crazyflie à l'aide de données collectées avec un système de capture de mouvement (VICON). Ce modèle est ensuite utilisé par un contrôleur LQR pour commander le robot. LAMBERT et al. 2019 apprennent eux aussi un réseau de neurones modélisant la dynamique du Crazyflie, mais cette fois à partir de l'estimation d'état embarquée et des capteurs du robot, mais sans système de capteur externe. Ce modèle est ensuite utilisé avec un contrôleur type MPC. DUISTERHOF et al. 2019 apprennent directement un réseau de neurones réalisant la commande du robot. Ce contrôleur appris par renforcement (PPO) donne des consignes de haut niveau (gauche, droite, tout droit) est embarqué à bord du robot dans une tâche de suivi de source lumineuse. MOLCHANOV et al. 2019 apprennent par renforcement (PPO) un réseau de neurones réalisant une commande bas niveau, contrôlant

directement la poussée moteur, qu'ils embarquent à bord du Crazyflie pour une tâche de stabilisation du robot. PALOSSI et al. 2019 tirent parti d'un composant modulaire du Crazyflie, le deck IA, qui permet la réalisation des calculs parallèles à bas coût à bord du Crazyflie, pour déployer une commande basée sur la vision. Ce contrôleur est un réseau convolutionnel appris de manière supervisée et fournissant une commande haut niveau pour la navigation en intérieur.

## Approches classiques pour le contrôle des quadrotors

La commande des quadrotors est un sujet abordé dans de nombreux travaux (GUERRERO-CASTELLANOS et al. 2011 ; MINH-DUC HUA et al. 2009) Les quadrotors ne sont pas complètement actionnés, ils doivent modifier leur inclinaison pour se déplacer latéralement. Ceci rend l'utilisation d'une commande récursive (backstepping control) particulièrement adaptée à leur contrôle. LEE, LEOK et MCCORMICK 2010 décrivent un tel contrôleur pour un Crazyflie : à partir d'une vitesse de référence, les auteurs dérivent une orientation permettant d'atteindre cette vitesse de référence de manière géométrique puis appliquent cette orientation à l'aide d'un contrôleur proportionnel sur la vitesse angulaire du robot. MELLINGER et KUMAR 2011 complètent un contrôleur similaire par un algorithme de génération temps réel de trajectoires admissibles pour le robot dont le contrôleur peut ensuite assurer le suivi.

## Commande Optimale et Prédictive

Pour aller plus loin et concevoir des commandes adaptées à des perturbations d'environnement ou de robots spécifiques, la théorie de la commande optimale, dont le principe fondateur est introduit par PONTRYAGIN 1962, permet de formaliser la conception de contrôleur par le biais de la résolution des problèmes d'optimisations faisant intervenir les équations de la dynamique du robot. Les approches pour formuler et résoudre de manière exacte par le biais d'approximations numériques sont nombreuses. RAO 2012 présente les deux approches principales pour la résolution de ces problèmes d'optimisation : les méthodes directes qui se concentrent sur l'aspect optimisation et les méthodes indirectes qui s'intéressent à la résolution des problèmes d'équations différentielles inhérents à la formulation de ces problèmes. Il souligne néanmoins les nombreux ponts qui existent entre ces deux approches. KELLY 2017 présente une revue introductory orientée sur les approches basées collocation, c'est-à-dire utilisant de polynômes comme approximation des trajectoires continues dans le temps des états et commandes par opposition aux approches qui discrétisent le temps. Il y aborde des exemples en lien avec la robotique. Ce contrôle optimal est particulièrement utilisé par les approches de commande prédictive qui recalculent régulièrement des signaux de commandes optimaux à partir de l'état estimé du robot afin d'ajouter un retour d'état et de fermer la boucle de contrôle en conservant l'optimalité de la commande. On pourra trouver le détail des différentes approches utilisées en commande prédictive dans les ouvrages de RAWLINGS, MAYNE et DIEHL 2017 et RAKOVIĆ et W. S. LEVINE 2019. Le cours de TEDRAKE 2021b présente aussi des usages de la commande optimale en robotique.

## Outils pour la commande optimale

En parallèle des méthodes et algorithmes, de nombreuses bibliothèques et logiciels ont été développés pour la formulation et la résolution de ces problèmes de commande optimale. OZANA et al. 2021 présente une revue des outils existants, leurs caractéristiques et langages d'usage. RAO 2012 présente lui aussi différents outils utilisés pour la commande optimale. Un outil en particulier

est utilisé dans cette thèse, il s'agit de la bibliothèque CasADi (ANDERSSON et al. 2019). Cette bibliothèque permet la formulation et la résolution de problèmes de contrôle optimal à partir des environnements python et matlab. La résolution de ces problèmes se fait via l'appel à des solveurs externes à travers une interface qui permet d'utiliser de manière transparente plusieurs solveurs différents pour une même formulation. La formulation de ces problèmes passe par le calcul symbolique et il est possible de générer du code C pour embarquer directement celui-ci sur un robot. Le solveur utilisé dans cette thèse est IPOPT, un algorithme d'optimisation non linéaire sous contraintes introduit par WÄCHTER et BIEGLER 2006. Une introduction à l'utilisation de ce solveur est réalisée par WÄCHTER 2009. AMOS et al. 2018 présentent par ailleurs une bibliothèque permettant de calculer la dérivée du coût de trajectoires calculées par commande prédictive par rapport aux variables intervenant dans le calcul des trajectoires, comme par exemple certains paramètres de l'environnement. Ceci permet de faire de l'identification des paramètres de la dynamique.

## Utilisation de commandes optimales ou prédictives sur des quadrotors

La commande prédictive est utilisée dans de nombreux travaux pour le contrôle de drones, malgré un coût en calcul souvent important. CARLOS et al. 2020 implémente une commande prédictive haut niveau (consigne en orientation) basée commande optimale à l'aide de la bibliothèque CasADi et fait tourner ce contrôleur sur une machine externe pour contrôler un quadrotor Crazyflie en temps réel. BANSAL et al. 2016 utilisent la commande prédictive pour générer une trajectoire admissible sur un modèle appris hors ligne puis utilise un contrôleur LQR déporté pour le suivi de la trajectoire. LAMBERT et al. 2019 mettent en œuvre une commande prédictive type "random shooter" basée sur l'échantillonnage en parallèle et la sélection des commandes les meilleures sur une machine déportée afin de réduire les coûts en calcul de la commande prédictive. Sur un robot ayant une capacité de calcul plus importante, MANSOURI, KANELAKIS et al. 2020 embarquent une commande prédictive basée commande optimale résolue en temps réel pour la navigation à l'intérieur d'une mine. SONG et SCARAMUZZA 2022 apprennent un réseau de neurones qui restreint l'espace de recherche d'une commande optimale, réduisant ainsi le coût en calcul de celle-ci pour l'embarquer sur un quadrotor dans le but de franchir des portes mobiles.

## Conclusion

Les modèles de la dynamique des multirotors et les formalismes et outils des commandes prédictives et optimale permettent ainsi de contrôler des multirotors et en particulier des quadrotors comme le Crazyflie, mais le coût en calcul de ces approches ne permet un déploiement que pour des drones dont les capacités de calcul embarqué sont importantes.

## 2.3 Dynamique et équations des multirotors

### 2.3.1 Quelques repères et notations utiles

Afin de décrire la cinématique et la dynamique d'un multirotor sans ambiguïtés, il est nécessaire de préciser le contexte dans lequel sont faites les observations de ce mouvement et de ses causes. Ainsi, les grandeurs qui décrivent ce mouvement sont relatives à ce contexte d'observation, un *référentiel*, qu'il convient de préciser dans les notations de ces grandeurs. Dans le cadre de ce travail, pour définir un référentiel il sera nécessaire de préciser une origine et une base orthonormée. Pour

éviter toute ambiguïté, la mention du référentiel sera indiquée en indice des grandeurs utilisées, par exemple  $\mathbf{v}_X(t)$  indiquera la vitesse d'un point dans le référentiel  $X$  à l'instant  $t$ . Les grandeurs vectorielles seront distinguées des grandeurs scalaires par la mise en gras :  $\mathbf{f}$  représente une grandeur vectorielle et  $g$  une grandeur scalaire. La notation  $\overrightarrow{AB}$  sera réservée pour les vecteurs associés à deux points nommés, ici  $A$  et  $B$  pour distinguer la notation de ce vecteur de la notation  $AB$  qui fait souvent référence à une droite ou un segment.

La convention utilisée pour représenter par des nombres ces grandeurs géométriques caractérisant le mouvement dans un référentiel donné, appelée *système de coordonnées*, sera précisée d'une autre manière afin que la distinction entre référentiel et système de coordonnées soit claire. Pour un système de coordonnées on notera la représentation d'une grandeur dans ce système de coordonnées entre crochets, indicés par le nom du système de coordonnées, e.g.  $[p]_S$  pour les coordonnées du vecteur  $p$  exprimées le système de coordonnées  $S$ .

Chaque référentiel, parce qu'il possède une origine et une base orthonormée, définit implicitement un système de coordonnées. On notera le système de coordonnées défini par un référentiel  $X$ ,  $C_X$ . Il existe bien entendu d'autres systèmes de coordonnées que  $C_X$  permettant la représentation par des nombres des grandeurs du mouvement dans le référentiel  $X$ .

On commence par définir un référentiel terrestre dont la verticale dans la direction de la gravité, orientée vers le haut. Pour la durée des mouvements dont il sera question, ce référentiel sera supposé galiléen. On note  $W = (O_W, \mathbf{x}_W, \mathbf{y}_W, \mathbf{z}_W)$  ce référentiel, son origine et sa base orthonormée. Ce référentiel fera office de référentiel absolu, par opposition au référentiel  $B = (O_B, \mathbf{x}_B, \mathbf{y}_B, \mathbf{z}_B)$  centré sur le centre de masse du robot et dont l'orientation est fixée sur celle du robot.

Pour décrire le mouvement du robot, considéré comme un solide indéformable, on caractérise la position de l'ensemble des points du robot par rapport à un référentiel  $X$  à l'aide de deux transformations géométriques (FEATHERSTONE 1987) :

- une translation du centre de masse ;
- une rotation de l'ensemble des points du robot autour d'un axe passant par le centre de masse.

La translation est définie par le vecteur entre l'origine du référentiel et le centre de masse qu'on nommera *position*. La rotation autour du centre de masse est caractérisée par l'image de la base du repère du référentiel par rapport auquel on étudie le mouvement. On appelle cette rotation l'*orientation* du robot.

On notera le vecteur position du centre de masse du robot dans le référentiel inertiel à un instant  $t$ ,  $\mathbf{p}_W(t) = \overrightarrow{O_W O_B}(t)$  et la vitesse, dérivée de cette position,  $\mathbf{v}_W(t) = \lim_{dt \rightarrow 0} \frac{\mathbf{p}_W(t+dt) - \mathbf{p}_W(t)}{dt}$ .

Il existe plusieurs manières de représenter une rotation et une des plus courantes est l'utilisation de l'image des trois vecteurs de la base du référentiel inertiel qui caractérise cette rotation, rassemblés sous forme de matrice :  $M = (R(x_W), R(y_W), R(z_W))$  où  $R(x_W)$  est l'image du vecteur  $x_W$  par la rotation  $R$ .

Cette rotation transforme la base du repère du référentiel inertiel  $W$  en la base du repère du référentiel du robot  $B$  :

$$\begin{aligned} R(\mathbf{x}_W) &= \mathbf{x}_B \\ R(\mathbf{y}_W) &= \mathbf{y}_B \\ R(\mathbf{z}_W) &= \mathbf{z}_B \end{aligned}$$

Cette rotation permet aussi de convertir les coordonnées d'un vecteur  $\mathbf{u}$  exprimées dans le système de coordonnées  $C_B$  associé au référentiel  $B$  en les coordonnées du même vecteur exprimé dans  $C_W$  :

$$R([\mathbf{u}]_{C_B}) = [\mathbf{u}]_{C_W}$$

Il est important de prendre garde au fait que  $R$  transforme la base du référentiel  $W$  en la base du référentiel  $B$  mais que  $R$  transforme les coordonnées dans le repère de  $B$  en les coordonnées dans le repère de  $W$ .

C'est en accord la deuxième approche qu'on conviendra de noter la matrice indiquant l'orientation relative du repère d'un référentiel  $Y$  par rapport au repère d'un référentiel  $X$ ,  $R_Y^X$ . On aura ainsi la notation :

$$R_Y^X[\mathbf{u}]_{C_Y} = [\mathbf{u}]_{C_X}$$

qui permet de lire par associativité le système de coordonnées dans lequel est exprimée l'image d'un vecteur par une matrice de rotation. Par ailleurs on aura

$$\begin{aligned} R_Y^X &= (R_X^Y)^{-1} \\ &= (R_X^Y)^T \end{aligned}$$

L'orientation du robot sera donc caractérisée par la rotation  $R_B^W$  et on pourra utiliser aussi bien la matrice  $[R_B^W]_{C_W} = ([x_B]_{C_W}, [y_B]_{C_W}, [z_B]_{C_W})$  pour représenter cette rotation qu'une autre représentation comme un quaternion unitaire. La représentation par des quaternions unitaires est abordée par SOLÀ, DERAY et ATCHUTHAN 2021.

La dérivée de l'orientation d'un repère  $W$  par rapport à un repère  $B$  s'exprime en fonction de l'orientation par la relation suivante :

$$\dot{R}_B^W = M R_B^W$$

où  $M$  est une application linéaire antisymétrique donc assimilable à un produit vectoriel par un vecteur  $\boldsymbol{\omega}$  (SOLÀ, DERAY et ATCHUTHAN 2021). De cette manière on a pour tout vecteur  $\mathbf{u}$  :

$$M(\mathbf{u}) = \boldsymbol{\omega} \wedge \mathbf{u}$$

On note ainsi  $[\boldsymbol{\omega}]_\wedge = M$  l'application linéaire associée au produit vectoriel à gauche par  $\boldsymbol{\omega}$  et on a ainsi

$$\dot{R}_B^W = [\boldsymbol{\omega}]_\wedge R_B^W$$

On appelle *vitesse angulaire* de  $B$  par rapport à  $W$  le vecteur  $\boldsymbol{\omega}_{WB}$  tel que

$$\dot{R}_B^W = [\boldsymbol{\omega}_{WB}]_\wedge R_B^W$$

### 2.3.2 Équations de la cinématique et représentation de l'orientation

On a les équations suivantes :

$$\begin{aligned} \dot{\mathbf{p}}_W &= \mathbf{v}_W \\ \dot{\mathbf{v}}_W &= \frac{\mathbf{f}}{m} \\ \dot{R}_B^W &= \boldsymbol{\omega}_{WB} R_B^W \\ \dot{\boldsymbol{\omega}}_{BW} &= J^{-1}(\boldsymbol{\tau} - \boldsymbol{\omega}_{WB} \wedge (J\boldsymbol{\omega}_{WB})) \end{aligned}$$

Où  $\mathbf{f}$  est la résultante des forces appliquées au robot,  $\boldsymbol{\tau}$  est la résultante des moments appliqués au robot par rapport au centre de masse  $O_B$ ,  $J$  est l'inertie du robot par rapport au centre de masse et  $m$  la masse du robot.

Pour des raisons de simplicité des expressions, on choisit de travailler avec les coordonnées de ces grandeurs décrivant le mouvement exprimées par rapport à différents systèmes de coordonnées pour décrire le mouvement du robot.

Pour représenter l'état du robot, on utilisera par la suite :

- $[\mathbf{p}_W]_{C_W}$  le vecteur coordonné dans  $C_W$  de la position ;
- $[\mathbf{v}_W]_{C_W}$  le vecteur coordonné dans  $C_W$  de la vitesse ;
- $[R_B^W]_{C_W}$  la matrice de rotation exprimée dans  $C_W$  ;
- $[\boldsymbol{\omega}_{WB}]_{C_B}$  le vecteur coordonné dans  $C_B$  de la vitesse angulaire.

Et on aura alors la dynamique suivante :

$$\begin{aligned} [\dot{\mathbf{p}}_W]_{C_W} &= [\mathbf{v}_W]_{C_W} \\ [\dot{\mathbf{v}}_W]_{C_W} &= \frac{[\mathbf{f}]_{C_W}}{m} \\ [\dot{R}_B^W]_{C_W} &= [R_B^W]_{C_W} [[\boldsymbol{\omega}_{WB}]_{C_B}]_\wedge \\ [\dot{\boldsymbol{\omega}}_{BW}]_{C_B} &= [J]_{C_B}^{-1} ([\boldsymbol{\tau}]_{C_B} - [[\boldsymbol{\omega}_{WB}]_{C_B}]_\wedge [J]_{C_B} [\boldsymbol{\omega}_{WB}]_{C_B}) \end{aligned} \quad (2.1)$$

### 2.3.3 Modèles dynamiques des multirotors

Il existe une grande variété de multirotors tant par leur forme ou leurs capacités de mouvement. Je vais décrire ici un modèle générique permettant de caractériser une vaste gamme de robots via les paramètres suivants :

- le nombre  $n_p$  de moteurs ;
- pour chaque moteur indicé  $i \in [1, n_p]$  sa position  $\mathbf{p}_{a,i}$  par rapport au centre de masse du multirotor et l'orientation  $\mathbf{t}_i$  du vecteur unitaire d'actuation, c'est-à-dire le vecteur le long duquel une force est exercée par le moteur ;
- le domaine des forces  $[f_{\min}, f_{\max}]$  que peuvent exercer les moteurs et la dynamique de la force que peut exercer chaque moteur :  $\delta_f : (f, u) \mapsto \dot{f}$  ;
- pour chaque moteur  $i$ , un coefficient couple de traînée/poussée  $c_i$  ;
- la masse totale du multirotor  $m$  ;
- la matrice d'inertie du robot  $J$  exprimée dans le système de coordonnées du repère du robot.

Ces paramètres permettent en particulier de décrire les différentes architectures des classiques quadrotors, et de leurs variations à plus de 4 moteurs, mais aussi de nombreuses architectures permettant une actuation complexe (RASHAD et al. 2020).

Dans ce modèle, chaque moteur produit, le long de son axe  $t_i$ , une part de la poussée dont le drone se sert pour se déplacer. Cette poussée est appliquée au drone à la position du moteur  $i : p_{a,i}$ . Par ailleurs, ce modèle prend en compte la dynamique des moteurs, on inclura donc dans la description de l'état du robot à un instant  $t$  la poussée que chaque moteur exerce en plus des grandeurs décrivant son mouvement et la dérivée de ce mouvement. On supposera que les moteurs sont fixes par rapport au référentiel du robot et donc que les poussées exercées par ceux-ci sont selon un axe constant dans le système référentiel du robot. On suppose que les moteurs font tourner des hélices qui génèrent une poussée (par opposition à d'autres moyens de propulsion comme par exemple des moteurs de type "fusée"). Ces hélices génèrent aussi un couple de traînées, c'est la raison pour laquelle on précise une valeur  $c_i$  pour ces moteurs. Dans le cas de moteurs à hélice, la

valeur de  $c_i$  dépend du sens de rotation de l'hélice, et pour les autres moteurs elle peut être nulle si les moteurs ne génèrent pas de couple de traînées.

Une fois ces paramètres décrits, on peut exprimer la résultante des résultantes des forces liées à l'actionnement du drone :

$$\mathbf{f}_{act} = \sum_{i=1}^{n_p} f_i \mathbf{t}_i \quad (2.2)$$

où  $f_i \in [f_{\min}, f_{\max}]$  est l'intensité de la poussée exercée par le moteur  $i$ .

La résultante des couples par rapport au centre de masse est :

$$\boldsymbol{\tau}_{act} = \sum_{i=1}^{n_p} f_i (\mathbf{p}_{a_i} \wedge \mathbf{t}_i + c_i \mathbf{t}_i) \quad (2.3)$$

Si l'expression de la masse est indépendante du système de coordonnées, celle de l'inertie  $J$  ne l'est pas. Celle-ci est cependant fixe dans le système de coordonnées du robot et c'est pour cette raison qu'on préférera exprimer la variation de la vitesse angulaire dans ce système de coordonnées comme on l'a fait pour les équations en (2.1). Il s'ensuit que l'on souhaite exprimer la résultante des moments par rapport au centre de masse dans le système de coordonnées du robot :

$$[\boldsymbol{\tau}]_{C_B} = \sum_{i=1}^{n_p} f_i ([\mathbf{p}_{a_i}]_{C_B} \wedge [\mathbf{t}_i]_{C_B} + c_i [\mathbf{t}_i]_{C_B}) \quad (2.4)$$

Dans le système de coordonnées du robot, les coordonnées de la position  $[p_{a,i}]_{C_B}$  du moteur et de son orientation  $[\mathbf{t}_i]_{C_B}$  sont fixes, on peut donc exprimer la résultante des forces en fonction de son orientation par rapport au référentiel inertiel :

$$\begin{aligned} [\mathbf{t}_i]_{C_W} &= [R_B^W]_{C_W} [\mathbf{t}_i]_{C_B} \\ [\mathbf{f}_{act}]_{C_W} &= \sum_{i=1}^{n_p} f_i [R_B^W]_{C_W} [\mathbf{t}_i]_{C_B} \end{aligned} \quad (2.5)$$

En combinant (2.1), (2.5) et (2.4) on obtient l'ensemble d'équations suivant pour décrire la dynamique de la représentation d'état du multirotor :

$$\begin{aligned} [\dot{\mathbf{p}}_W]_{C_W} &= [\mathbf{v}_W]_{C_W} \\ [\dot{\mathbf{v}}_W]_{C_W} &= \frac{1}{m} \left( \sum_{i=1}^{n_p} f_i [R_B^W]_{C_W} [\mathbf{t}_i]_{C_B} + [f_{ext}]_{C_W} \right) - g [\mathbf{z}_w]_{C_W} \\ [\dot{R}_B^W]_{C_W} &= [R_B^W]_{C_W} [[\boldsymbol{\omega}_{WB}]_{C_B}]_{\wedge} \\ [\dot{\boldsymbol{\omega}}_{WB}]_{C_B} &= [J]_{C_B}^{-1} \left( \left( \sum_{i=1}^{n_p} f_i ([\mathbf{p}_{a_i}]_{C_B} \wedge [\mathbf{t}_i]_{C_B} + c_i [\mathbf{t}_i]_{C_B}) \right) + [\tau_{ext}]_{C_B} - [[\boldsymbol{\omega}_{WB}]_{C_B}]_{\wedge} [J]_{C_B} [\boldsymbol{\omega}_{WB}]_{C_B} \right) \\ \dot{f}_i &= \delta_f(u, f_i) \quad \forall i \in [1, n_p] \end{aligned} \quad (2.6)$$

Avec  $m, [\mathbf{t}_i]_{C_B}, [\mathbf{p}_{a_i}]_{C_B}, [J]_{C_B}$  et  $\delta_f$  des paramètres du modèle,  $g = 9.81 N.m^{-1}$  la constante de gravité locale,  $f_{ext}$  et  $\tau_{ext}$  d'éventuelles forces externes et les couples associés.

Ce modèle ne couvre bien entendu pas toutes les architectures de multirotors possibles, et en particulier il ne tient pas compte de toutes les architectures dont l'approche consiste à avoir une orientation variable pour les moteurs (RASHAD et al. 2020 ; HAMANDI et al. 2021). Ce modèle pourrait être étendu à ces architectures en ajoutant à la représentation de l'état du drone  $[\mathbf{t}_i]_{C_B}$  et  $[\mathbf{p}_{a_i}]_{C_B}$  plutôt que de supposer qu'il s'agit de paramètres du modèle fixés dans le temps. Cette dynamique est résumée dans la figure 2.2

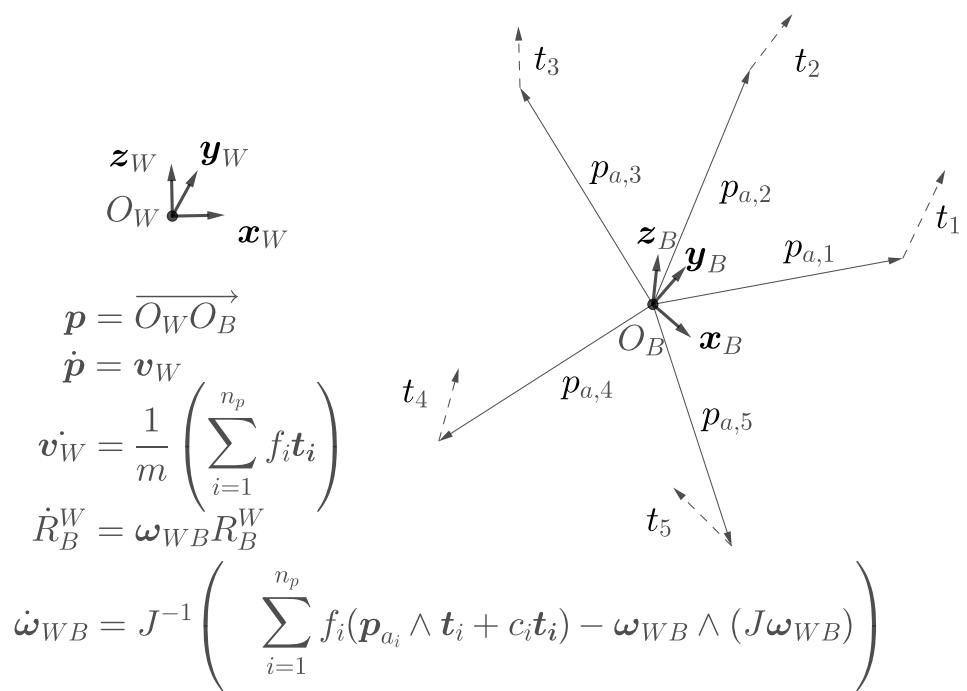


FIGURE 2.2 – Dynamique d'un multirotor

### 2.3.4 Le cas du quadrotor *Crazyflie*

Quand on instancie le modèle décrit précédemment 2.3.3 pour le Crazyflie, on trouve les valeurs suivantes(FÖRSTER 2015 ; LANDRY et al. 2016 ; ANTONSSON 2015) :

$$\begin{aligned}
n_p &= 4 \\
m &= 28 \cdot 10^{-3} \text{kg} \\
[J]_{C_B} &= \begin{pmatrix} 16.6 & 0. & 0. \\ 0. & 16.6 & 0. \\ 0. & 0. & 29.3 \end{pmatrix} \cdot 10^{-6} \cdot \text{kg} \cdot \text{m}^2 \\
c_i &= (-1)^{i+1} \cdot 5.9 \cdot 10^{-3} \text{m} \\
f_{\min} &= 0.02 \text{N} \\
f_{\max} &= 0.16 \text{N} \\
\delta_f(f, u) &= \begin{cases} \frac{f_{\max} - f_{\min}}{0.18} = 0.89 \text{Ns}^{-1} & \text{si } f \in ]f_{\min}, f_{\max}[ \\ 0 & \text{sinon} \end{cases} \\
t_i &= \mathbf{z}_B \\
[p_{a,1}]_{C_B} &= (-0.028, 0.028, 0) \\
[p_{a,2}]_{C_B} &= (0.028, 0.028, 0) \\
[p_{a,3}]_{C_B} &= (0.028, -0.028, 0) \\
[p_{a,4}]_{C_B} &= (-0.028, -0.028, 0)
\end{aligned} \tag{2.7}$$

Les valeurs de  $f_{\min}$  et  $f_{\max}$  sont obtenues à partir de l'expression de la poussée en fonction de la commande inférée dans FÖRSTER 2015 où est aussi inférée la valeur de  $c_i$ .

Ici le modèle de dynamique des moteurs est un modèle linéaire. On trouve sur le site du fabricant du Crazyflie une courbe de la réponse d'un moteur à un échelon de commande (ANTONSSON 2015). Le temps de réponse total est de 180ms, ce modèle linéaire de la dynamique fait donc l'hypothèse que durant une période de temps  $\delta_t$  un moteur peut changer l'intensité de la poussée qu'il émet de  $\frac{f_{\max} - f_{\min}}{0.180s} = 0.89 \text{Ns}^{-1}$ .

La courbe expérimentale n'est manifestement pas une droite, mais j'utiliserais néanmoins un modèle simplifié, linéaire, dans la suite de ce travail. On voit par ailleurs que pour les régimes de basse poussée, le temps de réponse pour une réduction de poussée est bien plus important que pour les régimes à poussée importante. Pour cette raison, on limitera les moteurs à une poussée  $f_i \leq 0.02 \text{N}$  afin de rester dans un domaine où le modèle linéaire n'est pas trop éloigné des observations expérimentales.

La matrice d'inertie  $J$  mesurée dans FÖRSTER 2015 n'est pas diagonale, mais les valeurs non diagonales sont faibles ( $\sim 10\%$  des valeurs diagonales) et on modélisera donc  $J$  par une matrice diagonale.

Les différents moteurs sont répartis à respectivement  $\theta_i = -45^\circ, 45^\circ, 135^\circ$  et  $215^\circ$  au bout de bras de longueur  $l = 4 \text{cm}$ . Leurs coordonnées sont donc  $[p_{a_i}]_{C_B} = (l \cdot \cos(\theta_i), l \cdot \sin(\theta_i), 0)$ . Les 4 hélices sont orientées verticalement et fournissent une poussée vers le haut en alternant les sens de rotation des hélices afin d'équilibrer le couple de traînées des hélices et éviter de produire du lacet.

## 2.4 Contrôle optimal

### 2.4.1 Formulation et théorie du contrôle optimal

Un problème de contrôle optimal est un problème dans lequel on cherche un contrôle (ou commande)  $u$  dont la mise en œuvre fait se comporter un système dynamique comme souhaité (RAO 2012). Plus précisément, soit un système  $S$  dont l'état est représenté par un vecteur  $\mathbf{x}_S \in \mathcal{X} \subset \mathbb{R}^n$ .  $S$  est commandé grâce à un signal  $u \in \mathcal{U} \subset \mathbb{R}^m$ , c'est à dire que  $u$  est le moyen par lequel un opérateur de  $S$  peut influencer l'évolution de  $\mathbf{x}$ . La dynamique de  $S$  est décrite par la fonction  $f : \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}^n$  c'est à dire :

$$\dot{\mathbf{x}}_S = f(\mathbf{x}_S, u)$$

On se donne un critère  $J : \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}$  qui caractérise la désirabilité d'un état de  $S$  et de la commande mise en œuvre.

Avec ces notations on peut formaliser le problème de contrôle optimal entre deux instants  $t_0$  et  $t_f$  et à partir d'un état initial du système  $\mathbf{x}_S(t_0) = \mathbf{x}_0$  de la manière suivante (RAO 2012) :

$$\begin{aligned} u^* = & \arg \max_{u \in \mathcal{F}(\mathcal{T}, \mathcal{U})} \int_{t_0}^{t_f} C(x_S(t), u(t)) dt \\ & x_S(t_0) = x_0 \\ & \dot{x}_S(t) = f(x_S(t), u(t)) \end{aligned} \quad (2.8)$$

Dans cette formulation classique, la *variable de décision*  $u$  est une fonction quelconque du temps et  $\mathcal{F}((T), (U))$  est l'ensemble de ces fonctions. Rechercher  $u^*$  dans un espace si grand est difficile, et c'est ce qui distingue les problèmes de contrôle optimal des problèmes d'optimisation non linéaire en général.

Pour calculer  $u^*$  il y a plusieurs approches (RAO 2012), une première approche consiste à calculer analytiquement une trajectoire solution de l'équation différentielle  $\dot{x}_S(t) = f(x_S(t), u(t))$  qui s'exprime en fonction de  $u$  puis de résoudre un problème d'optimisation sur cette solution. Cette solution demande une expertise dans la résolution d'équations différentielles et est en général difficile à mettre en œuvre si l'équation différentielle est suffisamment complexe. Deux approches types, numériques, sont souvent préférées : les approches directes et indirectes. Dans les approches indirectes, on se sert du principe de maximum de Pontryagin (PONTRYAGIN 1962 ; RAO 2012) pour exprimer des conditions nécessaires sur la trajectoire optimale, puis on cherche numériquement des solutions qui vérifient ces conditions. Ces approches indirectes sont aussi qualifiées de “optimize then discretize”.

Dans les approches directes, la solution cherchée et la trajectoire sont d'abord exprimées sous la forme d'une fonction paramétrique (e.g. des polynômes) qui permet de transformer le problème de recherche dans un espace de dimension infinie en un problème d'optimisation de paramètres non linéaires plus classique. Ces approches directes sont aussi qualifiées de “discretize then optimize”. Pour plus de détails sur les différentes approches et techniques mises en œuvre pour la résolution de problèmes de contrôle optimal, on peut se référer à RAO 2012 ; KELLY 2017 ; BETTS 2010.

### 2.4.2 Résolution du contrôle optimal d'un multirotor

Ici, la résolution du problème de contrôle optimal sera réalisée avec une approche directe. La première étape de discréétisation sera d'introduire  $k - 1$  instants intermédiaires et un pas de temps  $h$  :

$$t_0 < t_1 = t_0 + h < \dots < t_f - h = t_{k-1} + h = t_f$$

Entre ces instants,  $\mathbf{u}$  est supposé constant. Cette hypothèse discréétisation du temps pour les commandes est appuyée par le fait que, dans le contexte du contrôle d'un robot, l'ordinateur de bord qui calcule et envoie les commandes aux moteurs fonctionne avec un temps discret. De manière similaire, l'évolution de l'état au cours de la trajectoire est faite avec un temps discret, c'est-à-dire que l'état de  $S$  est constant entre  $t_i$  et  $t_{i+1}$ . Cette discréétisation du temps pour les états n'est pas justifiée par un argument similaire à celui de la discréétisation du temps pour la commande, mais par un argument de simplicité de la formulation. Cependant, l'évolution de cet état, c'est-à-dire l'intégration de sa dérivée est réalisée à grâce à un schéma d'intégration Runge-Kutta d'ordre 4 (RAO 2012)

Ce schéma d'intégration permet de calculer des états  $\hat{\mathbf{x}}$  à des instants intermédiaires entre  $t_i$  et  $t_{i+1}$  de la manière suivante :

$$\mathbf{k}_1 = f(\mathbf{x}_S(t_i), \mathbf{u}(t_i)) \quad (2.9)$$

$$\hat{\mathbf{x}}_1 = \mathbf{x}_S(t_i) + \frac{h}{2} \mathbf{k}_1 \quad (2.10)$$

$$\mathbf{k}_2 = f(\hat{\mathbf{x}}_1, \mathbf{u}(t_i)) \quad (2.11)$$

$$\hat{\mathbf{x}}_2 = \mathbf{x}_S(t_i) + \frac{h}{2} \mathbf{k}_2 \quad (2.12)$$

$$\mathbf{k}_3 = f(\hat{\mathbf{x}}_2, \mathbf{u}(t_i)) \quad (2.13)$$

$$\hat{\mathbf{x}}_3 = \mathbf{x}_S(t_i) + h \mathbf{k}_3 \quad (2.14)$$

$$\mathbf{k}_4 = f(\hat{\mathbf{x}}_3, \mathbf{u}(t_i)) \quad (2.15)$$

$$\mathbf{x}_S(t_{i+1}) = \mathbf{x}_S(t_i) + h \frac{(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)}{6} \quad (2.16)$$

$$(2.17)$$

La figure 2.3 illustre ce schéma d'intégration graphiquement. On note cette intégration entre  $t_i$  et  $t_{i+1}$  à partir de la dynamique  $f$  et de l'état initial  $\mathbf{x}_i$  avec une commande  $\mathbf{u}_i$  et un pas de temps  $h$  :

$$I_{RK_4}(f, t_i, t_{i+1}, h, \mathbf{x}_i, \mathbf{u}_i)$$

Avec cette discréétisation, le problème de contrôle optimal devient :

$$\mathbf{u}^* = \arg \max_{\mathbf{u} \in \mathcal{U}^k} \sum_{i=0}^{k+1} C(\mathbf{x}_i, \mathbf{u}_i) dt \quad (2.18)$$

$$\mathbf{x}_0 = \mathbf{x}_S(t_0)$$

$$\forall i \in [0, k], \mathbf{x}_{i+1} = I_{RK_4}(f, t_i, t_{i+1}, h, \mathbf{x}_i, \mathbf{u}_i)$$

où on a remplacé une recherche dans un espace de dimension infinie  $\mathbf{u} \in \mathcal{F}(\mathcal{T}, \mathcal{U})$  en une recherche dans un espace de dimension  $k : \mathcal{U}^k$ .

Dans une telle formulation, l'expression de l'état à un instant  $t_i$ , c'est-à-dire la valeur de  $\mathbf{x}_S(t_i)$ , dépend indirectement de tous les signaux de commandes antérieurs. Ces dépendances en cascade

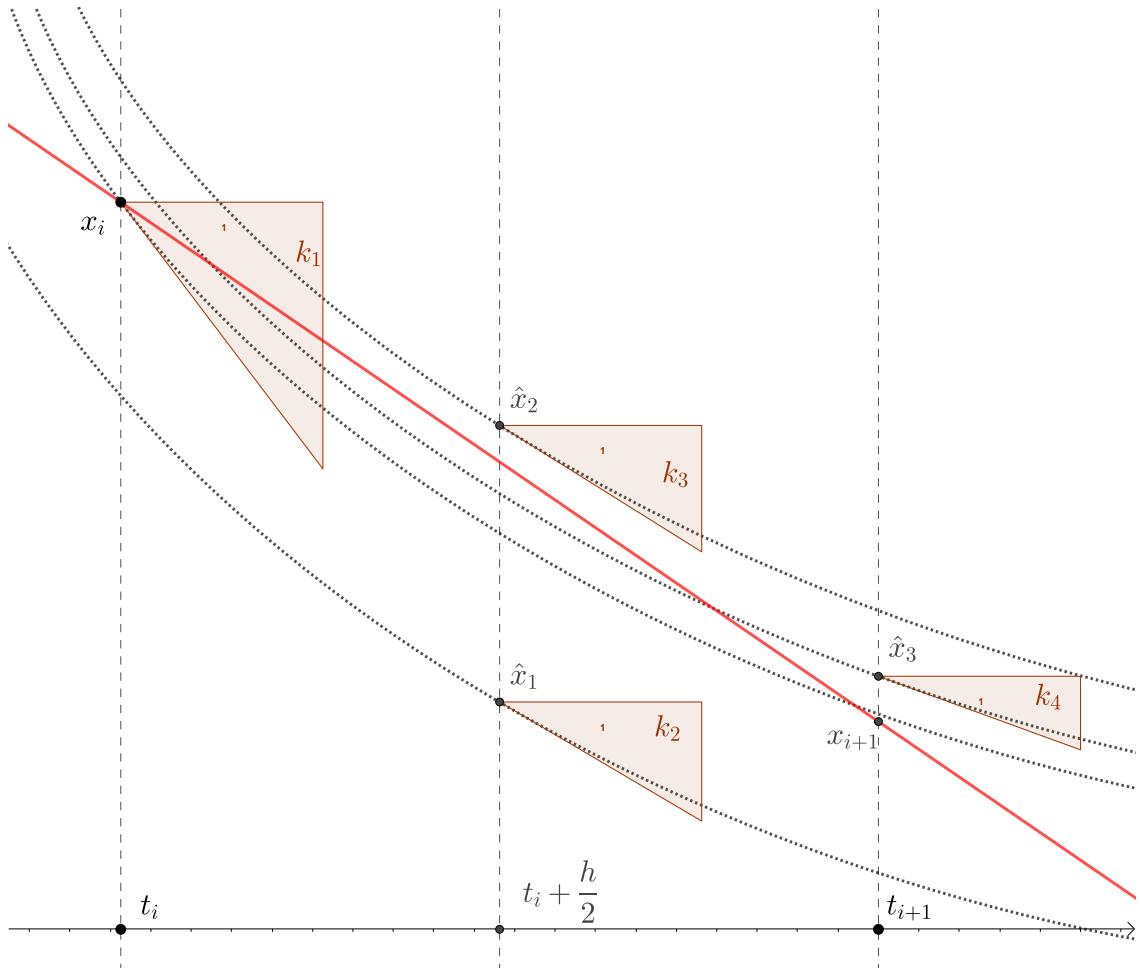


FIGURE 2.3 – Un schéma Runge Kutta d’ordre 4. Les valeurs de  $x$  sont sur l’axe des ordonnées et le temps sur l’axe des abscisses. La droite en rouge est le pas final d’intégration, moyenne pondérée des pentes  $k_1, k_2, k_3, k_4$ . Les arcs en pointillés sont les dérivées de la dynamique, mais ces arcs représentent une fonction de  $\mathcal{X}$  dans  $\mathcal{X}$  et non de  $\mathcal{T}$  dans  $\mathcal{X}$ .

ont pour conséquence une sensibilité plus importante de la fonction optimisée ( $J$ ) aux variables de décision et des instabilités numériques dues à l'effet potentiellement important de la commande initiale sur les états proches de la fin (TEDRAKE 2021b ; RAO 2012 ; BETTS 2010). Pour réduire ces dépendances et faciliter la résolution du problème d'optimisation, on peut introduire l'état aux instants intermédiaires comme variables de décision :

$$\begin{aligned}
 (\boldsymbol{u}^*, \boldsymbol{x}^*) = & \arg \max_{\boldsymbol{u} \in \mathcal{U}^k, \boldsymbol{x} \in \mathcal{X}^k} \sum_{i=0}^{k+1} C(\boldsymbol{x}_i, \boldsymbol{u}_i) dt \\
 & \boldsymbol{x}_0 = \boldsymbol{x}_S(t_0) \\
 & \forall i \in [0, k], \quad \boldsymbol{x}_{i+1} = I_{RK_4}(f, t_i, t_{i+1}, h, \boldsymbol{x}_i, \boldsymbol{u}_i)
 \end{aligned} \tag{2.19}$$

Cette formulation ajoute des variables de décision (ici  $\boldsymbol{x}_i$  de dimension  $(k + 1) \times n$ , ce qui fait plus que doubler le nombre de variables de décision), mais permet de rendre la structure du problème plus creuse (TEDRAKE 2021b ; BETTS 2010) et donc de réduire les dépendances à long terme entre les variables de décision. Bien entendu ces dépendances ne disparaissent pas complètement puisque les équations des contraintes représentant la dynamique du système font toujours partie du problème d'optimisation. Ceci permet cependant de décorreler en partie l'influence de la commande sur l'ensemble des pas de temps futur et permet ainsi une forme de parallélisme dans la recherche de la solution optimale.

Une fois ce problème d'optimisation non linéaire formulé, la résolution est confiée à un solveur générique : IPOPT (Interior Point OPTimizer WÄCHTER et BIEGLER 2006). IPOPT résout des problèmes d'optimisation selon les étapes décrites par l'algorithme 1. IPOPT est un solveur qui ne donne pas des solutions globales, mais locales au problème d'optimisation passé en entrée.

---

#### **Algorithme 1** Algorithme IPOPT

---

- 1: remplacement des inégalités dans les contraintes par une pénalisation de la proximité à ces contraintes
  - 2: expression des conditions d'optimalité au premier ordre du problème
  - 3: résolution des conditions d'optimalité par une méthode de Newton
  - 4: filtrage des candidats solutions pour ne garder que les maximums
  - 5: réduction du poids de la pénalisation des inégalités
  - 6: retour en 2 ou sortie de l'algorithme
- 

La méthode des points intérieurs dont il tire son nom cherche les solutions au problème à l'intérieur de l'ensemble admissible (l'ensemble délimité par les contraintes d'inégalités), contrairement à l'algorithme du simplexe DANTZIG 1951 ; GILBERT 2021 qui cherche ses solutions sur le bord de cet ensemble admissible. C'est l'étape 1 qui permet cette recherche en ajoutant à la fonction à maximiser une fonction barrière qui pénalise la proximité au bord de l'ensemble admissible avec une pénalité infinie quand les valeurs sont sur le bord, via une fonction logarithme ou inverse. Quand on fait tendre le poids de cette pénalité vers 0, les solutions pénalisées tendent vers les solutions du problème contraint par inégalités. Dans les formulations précédentes, aucune inégalité n'est directement apparente, mais l'appartenance des variables de décisions aux ensembles  $\mathcal{X}$  et  $\mathcal{U}$  peut faire intervenir des inégalités. On peut par exemple penser à des contraintes sur le régime de vitesse d'un robot, ou bien un ensemble de positions autorisées, etc.

Les conditions d'optimalité au premier ordre de l'étape 2 sont des conditions nécessaires que vérifient les valeurs optimales variables de décision du problème. Mais ce ne sont pas des conditions suffisantes, les valeurs qui vérifient ces conditions peuvent aussi être des minimums ou des points selle. C'est pour cette raison qu'une phase de filtration pour ne garder des valeurs vérifiant ces conditions que les maximums.

Afin de pouvoir exprimer les conditions d'optimalité au premier ordre, il faut avoir accès à la dérivée première de la fonction à maximiser, aux égalités des contraintes. Pour résoudre ces conditions par une méthode de Newton il est nécessaire d'avoir accès à la dérivée première de ces conditions, c'est-à-dire la dérivée seconde de la dynamique et de la fonction objectif.

Pour obtenir un problème de contrôle optimal dont les équations soient deux fois dérivable, j'ai choisi de formuler les équations de la dynamique  $f$  et la fonction  $J$  à l'aide de CasADi (ANDERSSON et al. 2019). Cette bibliothèque dédiée à l'optimisation non linéaire qui permet de dériver de manière automatique des expressions formelles par l'intermédiaire d'une interface de calcul symbolique. Elle est conçue pour pouvoir fournir à des solveurs comme IPOPT des représentations de problèmes que ces solveurs peuvent résoudre.

#### 2.4.3 Résultats en simulation

La formulation dans un langage exploitable par le solveur IPOPT du problème d'optimisation de la trajectoire d'un robot permet donc de simuler des trajectoires grâce aux commandes optimales  $\mathbf{u}^*$ , solutions du problème. Ce simulateur diffère de la formulation générique décrite précédemment.

La première différence concerne la représentation de l'orientation qui est faite à l'aide de quaternions unitaires (FRESK et NIKOLAKOPOULOS 2013) plutôt que de matrices de rotations. Ce choix résulte du fait de la volonté d'utiliser, par la suite, ce simulateur et les trajectoires optimales calculées avec un algorithme d'apprentissage et donc de limiter la dimension de la représentation de cet état en utilisant 4 dimensions du quaternion plutôt que les 9 de la matrice de rotation. Les différentes manières de représenter l'orientation et leurs mérites respectifs sont détaillés par SOLÀ, DERAY et ATCHUTHAN 2021.

Les quaternions unitaires, comme leur nom l'indique, sont de norme 1. Le schéma d'intégration présenté plus tôt (2.9) traite les objets intégrés comme s'ils étaient des grandeurs euclidiennes, ce que ne sont pas les quaternions unitaires. En effet, une intégration de leur dérivée au premier ordre ne garantit pas de conserver leur caractère unitaire. Ainsi, afin de prévenir une accumulation des erreurs sur la valeur de la norme de cette représentation, une renormalisation de la représentation de l'orientation est effectuée à chaque étape de simulation. Une représentation de l'orientation par une matrice de rotation aurait nécessité une opération analogue. Cette normalisation n'est pas intégrée directement au problème d'optimisation pour des raisons de stabilité numérique.

Dans ce simulateur, on introduit des perturbations par rapport au modèle nominal décrit par (2.6). Ces perturbations interviennent directement comme des forces et couples non décrits dans le modèle nominal, c'est-à-dire l'ajout d'un terme à  $f_{ext}$  et  $\tau_{ext}$  dans (2.6).

Afin de représenter différents profils de perturbations, on génère les séquences de perturbations par interpolation à partir d'un échantillonnage gaussien à une fréquence donnée. Ceci permet de donner une certaine régularité au profil des perturbations, régularité contrôlée par la fréquence à laquelle sont échantillonnes les points support de l'interpolation. Si on souhaite simuler une

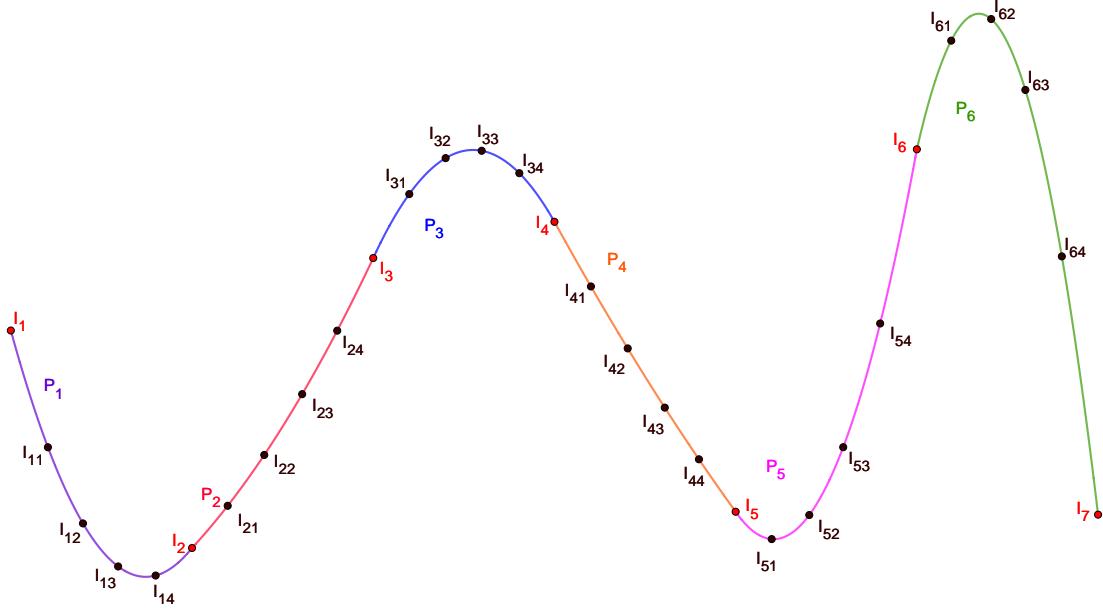


FIGURE 2.4 – Illustration de la génération de perturbations à fréquence donnée par interpolation. Les points en rouge ( $I_1, I_2, I_3, I_4, I_5, I_6, I_7$ ) sont échantillonnés à partir d'une distribution unique et répartis dans le temps à fréquence  $F_2 = 5F_1$ . Entre deux de ces points, on intercale 4 points dont l'ordonnée est calculée par interpolation. Les différentes couleurs de courbe présentent les différents polynômes d'ordre 3 utilisés pour l'interpolation. Chacun de ces polynômes passe par deux points rouges consécutifs et la pente de ces polynômes est ajustée à celle du polynôme suivant aux points de jonction.

trajectoire à une fréquence  $F_1$ , sur une durée  $T$ , on peut échantillonner de manière indépendante à une fréquence  $F_2 < F_1$  des perturbations sur toute la durée  $T$ . Ensuite, on calcule la valeur des forces et couples de perturbations aux instants intermédiaires par interpolation entre les valeurs échantillonnées iid. La figure 2.4 illustre un échantillonnage à 5Hz interpolé à 1Hz par des polynômes de degré 3, c'est-à-dire que pour chaque couple de points consécutifs  $I_i = (x_i, y_i)$ ,  $I_{i+1} = (x_{i+1}, y_{i+1})$ , on ajuste un polynôme de degré 3,  $P_i$  qui passe par ces deux points et dont la dérivée aux extrémités est égale aux dérivées au même point des polynômes des couples adjacents : Les perturbations utilisées par ce simulateur utilisent aussi une interpolation par des polynômes de degré 3.

$$P_i(x_i) = y_i \quad (2.20)$$

$$P_i(x_{i+1}) = y_{i+1} \quad (2.21)$$

$$P'_i(x_{i+1}) = P'_{i+1}(x_{i+1}) \quad (2.22)$$

#### 2.4.3.1 Contrôleurs basés sur le calcul du contrôle optimal

La résolution du problème de contrôle optimal telle que présentée dans la section 2.4.2 calcule une séquence d'états et de commandes localement optimaux sachant un modèle nominal de multirotor.

La première approche pour utiliser cette séquence de commandes optimales sur un multirotor est de résoudre un problème de contrôle optimal à partir de l'état initial du robot, puis d'appliquer successivement les commandes de la séquence. Ce fonctionnement en boucle ouverte ignore totalement les différences de dynamique entre les équations qui ont servi à formuler le problème de contrôle optimal et la dynamique réelle robot, ou même celle d'un simulateur faisant intervenir une dynamique un peu différente ou des perturbations. Ces différences peuvent faire diverger très rapidement l'état du robot de la séquence optimale d'état calculée par le contrôle optimal.

Pour cette raison, afin de contrôler un quadrotor par l'utilisation de techniques de contrôle optimal tout en tenant compte d'un retour d'état via des capteurs, j'utiliserais une approche dite *Model Predictive Control* (MPC) (RAWLINGS, MAYNE et DIEHL 2017). Le contrôle en boucle ouverte demande de résoudre un problème de contrôle optimal sur toute la durée de la trajectoire en une fois, et à la fréquence à laquelle seront appliquées les commandes. L'approche MPC consiste à résoudre le problème de contrôle optimal à partir d'un état initial, d'exécuter la première commande optimale de la séquence, puis de résoudre un nouveau problème de contrôle optimal à partir du nouvel état mesuré du robot.

Le modèle ne prédit toujours pas les phénomènes non pris en compte dans les équations ni les perturbations, mais replanifie à chaque étape, permettant ainsi de corriger les divergences entre l'état désiré et l'état actuel.

Il est courant qu'un contrôleur MPC ne résolve le problème de contrôle optimal que sur une plage de temps assez courte démarrant à l'instant courant plutôt que jusqu'à la fin de la plage de temps attendue pour la trajectoire complète. Le fait de résoudre ainsi le problème de contrôle optimal sur un horizon mouvant renforce le fait que le résultat du problème de contrôle optimal n'est qu'un optimum local. Ce caractère local de l'optimum était déjà conséquence de la résolution du problème par IPOPT 1.

Ceci permet aussi de formuler un problème de contrôle optimal pour lequel les commandes sont appliquées à une fréquence différente de celle à laquelle sont réellement appliquées les commandes sur le robot. Il est par exemple possible d'utiliser un contrôleur MPC avec un problème de contrôle optimal pour lequel le pas de temps sur la trajectoire optimisée est supérieure à la fréquence à laquelle le contrôleur MPC fonctionne. Cette approche qui consiste à utiliser des fréquences différente dans le MPC de celles utilisées dans la formulation du problème de contrôle optimal n'est pas standard et a été décidée à l'aide de constatations expérimentales (en simulation) : si la fréquence du problème de contrôle optimal est plus faible, le nombre de variables est plus faible et le temps nécessaire pour la résolution plus court alors que les commandes obtenues continuent à donner expérimentalement des résultats satisfaisants.

L'approche MPC permet éventuellement de résoudre le problème de contrôle optimal sur un horizon plus long en temps, mais avec un nombre de variables de décision, et donc un coût computationnel de résolution de même ordre en échange d'une optimalité plus locale.

#### **2.4.3.2 Contrôleur géométrique proportionnel pour quadrotor**

Pour mettre en perspective les deux contrôleurs basés sur le problème de contrôle optimal décrit dans la section 2.4.3.1, j'utilise un contrôleur géométrique proportionnel pour quadrotor. Dans la suite de ce document, il sera parfois fait référence à un contrôleur abusivement appelé "PID". Ces références se rapportent au contrôleur décrit ici qui ne fait intervenir aucun terme intégral. Ce contrôleur, adapté et simplifié des travaux de LEE, LEOK et McCCLAMROCH 2010 et FRESK et

NIKOLAKOPOULOS 2013, calcule la direction que devrait prendre la verticale  $[\mathbf{z}]_{C_W}$  dans le repère du quadrotor de manière à aligner cette verticale avec l'accélération nécessaire pour corriger les erreurs en position et en vitesse du robot. Comme la seule accélération que peut produire un quadrotor se fait selon sa verticale (ses moteurs sont orientés selon celle-ci) il faut ensuite calculer la norme que cette accélération doit prendre. On déduit de la différence entre l'orientation actuelle et la rotation désirée la vitesse angulaire nécessaire pour réaliser cette orientation. On calcule ensuite les commandes nécessaires pour que le drone produise les couples correspondant à la vitesse angulaire désirée et l'accélération désirée inversant la matrice d'allocation des commandes.

Plus précisément, on définit une erreur en position et une erreur en vitesse par rapport à une trajectoire de référence  $\mathbf{e}_p = \mathbf{p}_r - \mathbf{p}$ ,  $\mathbf{e}_v = \mathbf{v}_r - \mathbf{v}$ . À partir de ces erreurs, on détermine l'accélération désirée qui est proportionnelle à ces erreurs :  $\mathbf{a}_d = k_p \mathbf{e}_p + k_v \mathbf{e}_v + g \mathbf{z}_W$ , on y ajoute un terme pour compenser l'accélération de gravité. Une fois cette accélération désirée calculée, on détermine les 3 vecteurs de base d'une orientation du robot permettant de réaliser cette accélération. Le premier de ces vecteurs est celui de la verticale, déterminé par l'accélération désirée :  $\mathbf{z}_d = \frac{\mathbf{a}_d}{\|\mathbf{a}_d\|}$ , ensuite on projette  $\mathbf{x}_W$  sur le plan normal à  $\mathbf{z}_d$  pour obtenir un deuxième vecteur  $\mathbf{x}_d = \left( \frac{\mathbf{z}_d \wedge \mathbf{x}_w}{\|\mathbf{z}_d \wedge \mathbf{x}_w\|} \right) \wedge \mathbf{z}_d$ . Le dernier vecteur de cette base est obtenu en complétant celle-ci de manière à ce qu'elle soit orthonormale :  $\mathbf{y}_s = \mathbf{z}_d \wedge \mathbf{x}_d$ . Ces 3 vecteurs de base, exprimés dans le système de coordonnées du référentiel inertiel  $C_W$  donnent une rotation  $R_d$  décrivant l'orientation qui aligne la base du système de coordonnées du robot avec celle désirée.

On calcule une erreur d'orientation par la conversion de la rotation entre l'orientation actuelle  $R$  et l'orientation désirée  $R_d$ ,  $R_r = R_d R^{-1}$ , sous forme d'un axe et d'un angle}  $\mathbf{e}_o = \text{Angle}(R_r) \text{Axe}(R_r)$ . On en déduit ensuite les couples désirés  $\boldsymbol{\tau}_d = J(k_o \mathbf{e}_o - k_\omega \boldsymbol{\omega}) + \boldsymbol{\omega} \wedge J \boldsymbol{\omega}$ . On déduit ensuite l'intensité des forces produites par chaque moteur  $(f_i)^T = M^{-1}(\|\mathbf{a}_d\|, \boldsymbol{\tau}_d)^T$  avec  $M$  la matrice d'allocation telle que  $M(f_i)^T = (\mathbf{f}_z, \boldsymbol{\tau})^T$ .

Dans la suite, et en particulier dans le contexte des deux sous-sections suivantes, le coût de trajectoire optimisé dans le problème de contrôle optimal est le suivant :

$$C(\mathbf{x}, \mathbf{u}, t) = \|10^1(\mathbf{p} - \mathbf{p}_r(t))\|^2 + \|10^{-3}(\mathbf{v} - \mathbf{v}_r(t))\|^2 + \|5.10^{-3}(R - R_r(t))\|^2 + \|10^{-3}(\boldsymbol{\omega} - \boldsymbol{\omega}_r(t))\|^2 \quad (2.23)$$

Il s'agit donc d'un contrôle essentiellement en position vu la répartition des poids dans le coût. Comme le changement d'inclinaison est indispensable au changement de vitesse et donc de position, le poids de l'erreur en orientation est gardé faible afin de ne pas mener permettre que l'immobilisme soit une trajectoire optimale localement.

#### 2.4.3.3 Stabilisation

Dans un premier temps, pour évaluer la capacité d'un contrôleur basé sur la résolution d'un problème de contrôle optimal à contrôler un quadrotor, je présente des résultats en simulation dans un contexte de stabilisation du quadrotor à partir d'un état initial perturbé. La trajectoire de référence est alors constante dans le temps :

$$[\mathbf{p}_r]_{C_W} = (0, 0, 0) \quad [\mathbf{v}_r]_{C_W} = (0, 0, 0) \quad R_B^W r = I_3 \quad [\boldsymbol{\omega}_r]_{C_B} = (0, 0, 0) \quad (2.24)$$

On regarde la capacité des contrôleurs présentés plus tôt à stabiliser le robot à partir d'un état initial différent de celui de référence et en présence de perturbations. L'état initial est généré aléatoirement. Pour cela je perturbe l'état stabilisé :

- on ajoute à la position un vecteur aléatoire de norme  $\|\Delta_p\| \leq 0.5m$  ;
- on ajoute à la vitesse un vecteur aléatoire de norme  $\|\Delta_v\| \leq 0.3m.s^{-1}$  ;
- on change l'orientation d'un angle  $|\theta| < 45^\circ$  par rapport à un axe échantillonné uniformément ;
- on ajoute à la vitesse angulaire un vecteur aléatoire de norme  $\|\Delta_\omega\| \leq 0.05 rad.s^{-1}$ .

Les perturbations durant la simulation sont générées par le mécanisme décrit précédemment (figure 2.4) avec des échantillons réalisés à une fréquence 10 fois supérieure à celle du contrôle (20Hz vs 200Hz). Les forces et couples supplémentaires s'exerçant sur le système ont des normes  $\|f_{ext}\| \leq 2.10^{-3}N$  et  $\|\tau_{ext}\| \leq 2.10^{-5}Nm$ .

Ces perturbations s'exerçant au cours de la trajectoire ne sont pas connues à l'avance pour le problème de contrôle optimal, elles représentent les phénomènes non inclus dans le modèle de la dynamique du robot et sont ajoutées à la trajectoire pour estimer la robustesse des contrôleurs vis-à-vis du défaut de perfection du modèle utilisé dans la formulation du problème de contrôle optimal.

À l'inverse, l'état initial, différent de celui de référence, est une donnée connue lors de la résolution du problème de contrôle optimal. Cette différence entre les deux permet d'illustrer la capacité des contrôleurs à fonctionner en dehors de l'état de référence et à rejoindre celui-ci.

Les contrôleurs et la simulation fonctionnent ici à 200Hz. Le contrôleur en boucle ouverte résout donc le problème de contrôle optimal avec des variables de décision séparées de  $5.10^{-3}s$ .

Le contrôleur MPC lui fonctionne avec un horizon de 1s, et une fréquence de 20Hz. Cela signifie que toutes les  $5.10^{-3}s$  (200Hz) le contrôleur MPC résout un problème de contrôle optimal à partir de son état actuel pour lequel les variables de décisions sont espacées de  $5.10^{-2}s$  (20Hz) et couvrent une trajectoire durant 1s.

La figure 2.5 présente le résultat de l'application en boucle ouverte du contrôle obtenu par résolution du problème de contrôle optimal pour la stabilisation, en l'absence des perturbations décrites plus haut, ceci pour bien illustrer l'importance de celles-ci pour un contrôle en boucle ouverte.

L'application du contrôle en boucle ouverte, en l'absence de perturbations, permet une stabilisation rapide du robot à partir d'un état initial éloigné de l'état dans lequel est stabilisé. Ce contrôle fait initialement augmenter le coût en éloignant l'orientation de la référence. Ceci montre la capacité de ce contrôleur à prévoir un peu au-delà d'un futur immédiat et à accepter de passer par un état moins désirable pour atteindre un état plus désirable à terme.

Cependant, la figure 2.6 montre la grande sensibilité du contrôleur aux perturbations. Sous l'influence de celles-ci et sans retour d'état, le robot diverge et la commande ne parvient pas à le stabiliser. Sur cette figure la simulation est interrompue avant 6s, la durée de la trajectoire sur la figure 2.5, suite à l'écart important entre l'état du robot et celui désiré pour la stabilisation.

La figure 2.7 présente une trajectoire dans les mêmes conditions contrôlée par un *MPC*. Le retour d'état de ce contrôleur et le calcul d'une nouvelle commande optimale à chaque pas de temps permet au robot de se stabiliser malgré ces perturbations.

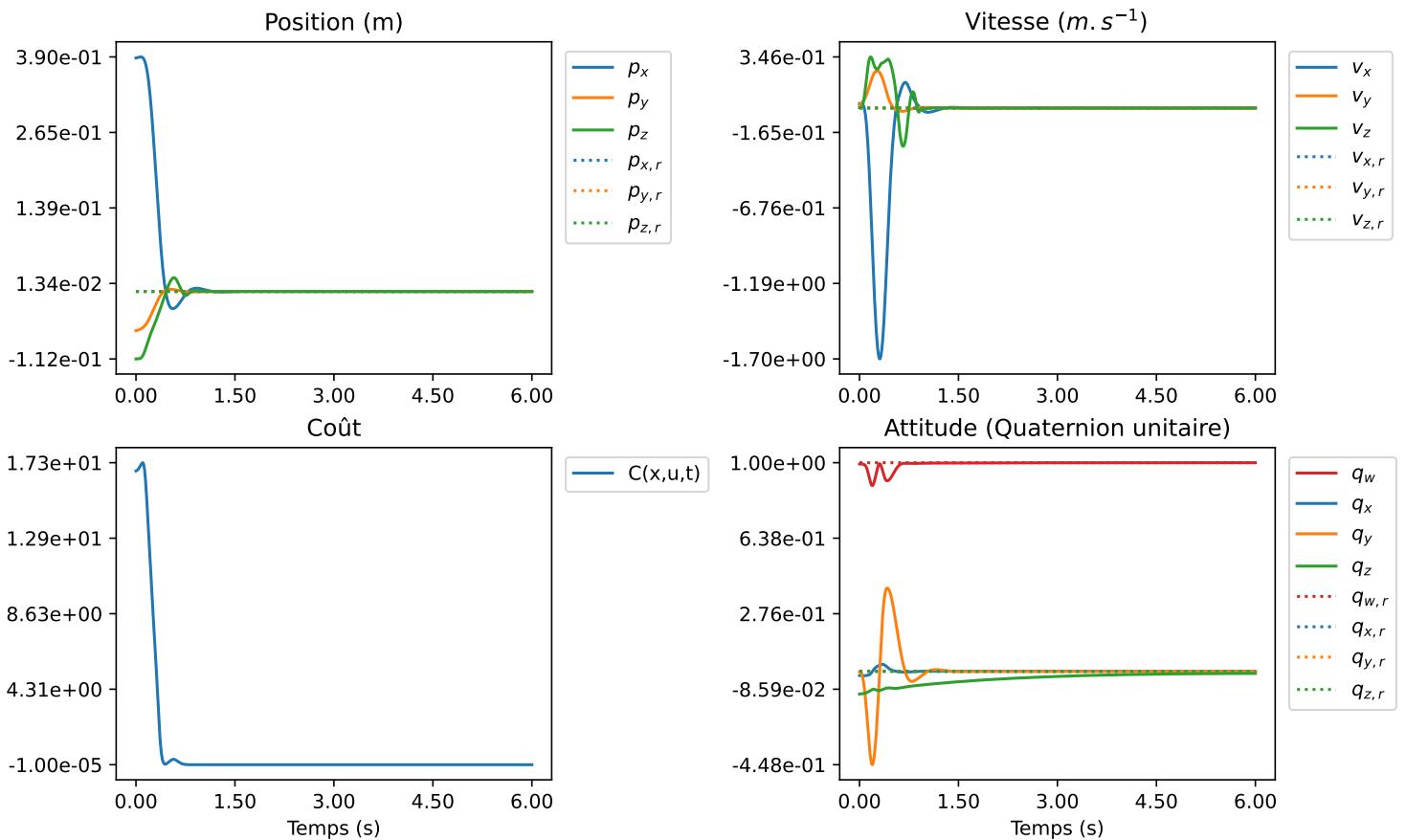


FIGURE 2.5 – Trajectoire de stabilisation avec un contrôle en boucle ouverte en l'absence de perturbations. L'état de référence est indiqué par un trait pointillé et l'indice  $r$ . Quand cet état de référence n'est pas visible, c'est qu'il est couvert par la courbe de la trajectoire et que ceux-ci coïncident. À partir d'un état initial éloigné de la référence, le contrôleur en boucle ouverte permet un retour rapide à l'origine et une diminution rapide du coût optimisé.

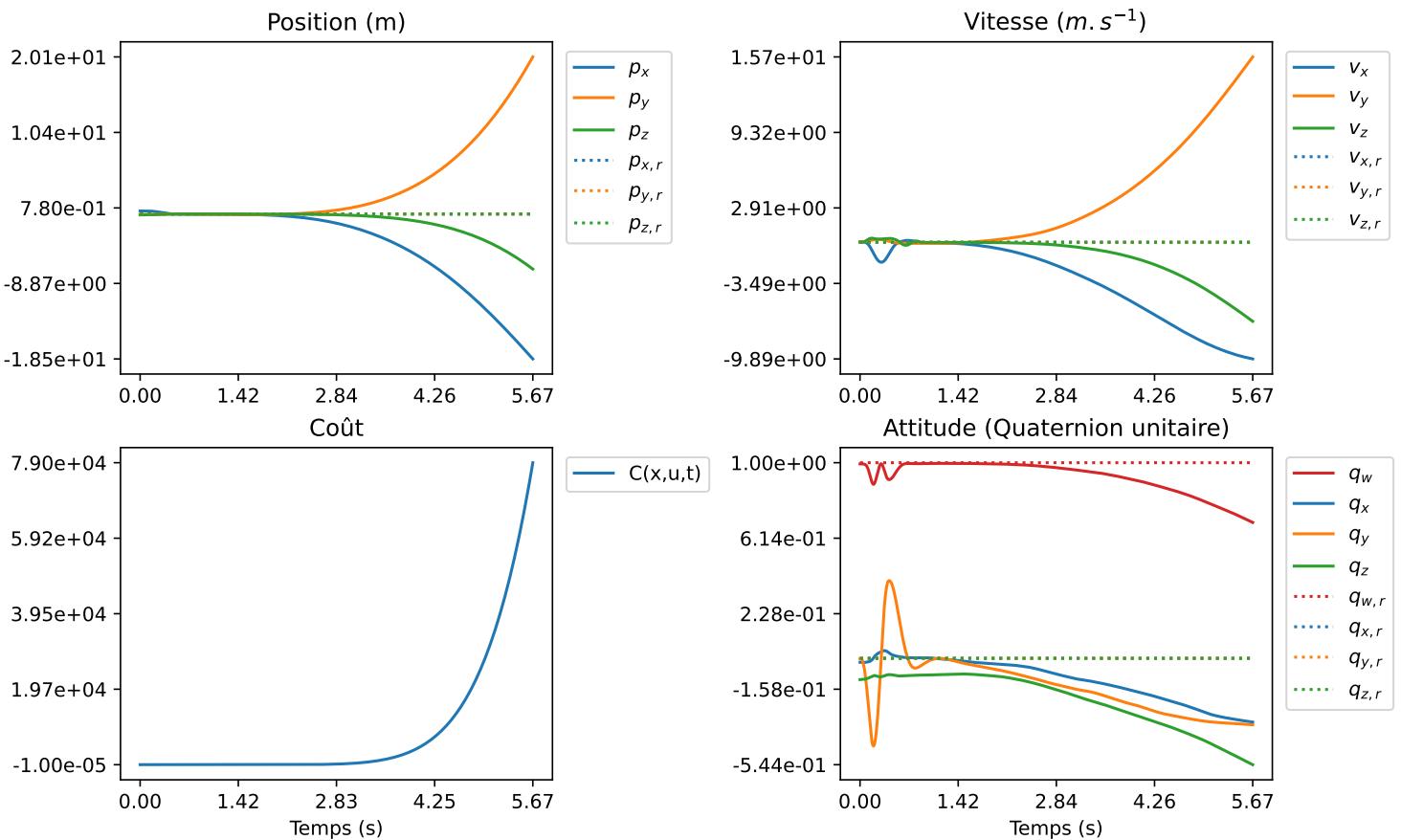


FIGURE 2.6 – Trajectoire de stabilisation avec un contrôle en boucle ouverte en présence de perturbations. L'état de référence est indiqué par un trait pointillé et l'indice  $_r$ . Quand cet état de référence n'est pas visible, c'est qu'il est couvert par la courbe de la trajectoire et que ceux-ci coïncident. La présence de ces perturbations fait rapidement diverger le robot de la trajectoire prévue. Le caractère instable de sa dynamique fait exploser le coût et provoque une interruption précoce de la simulation.

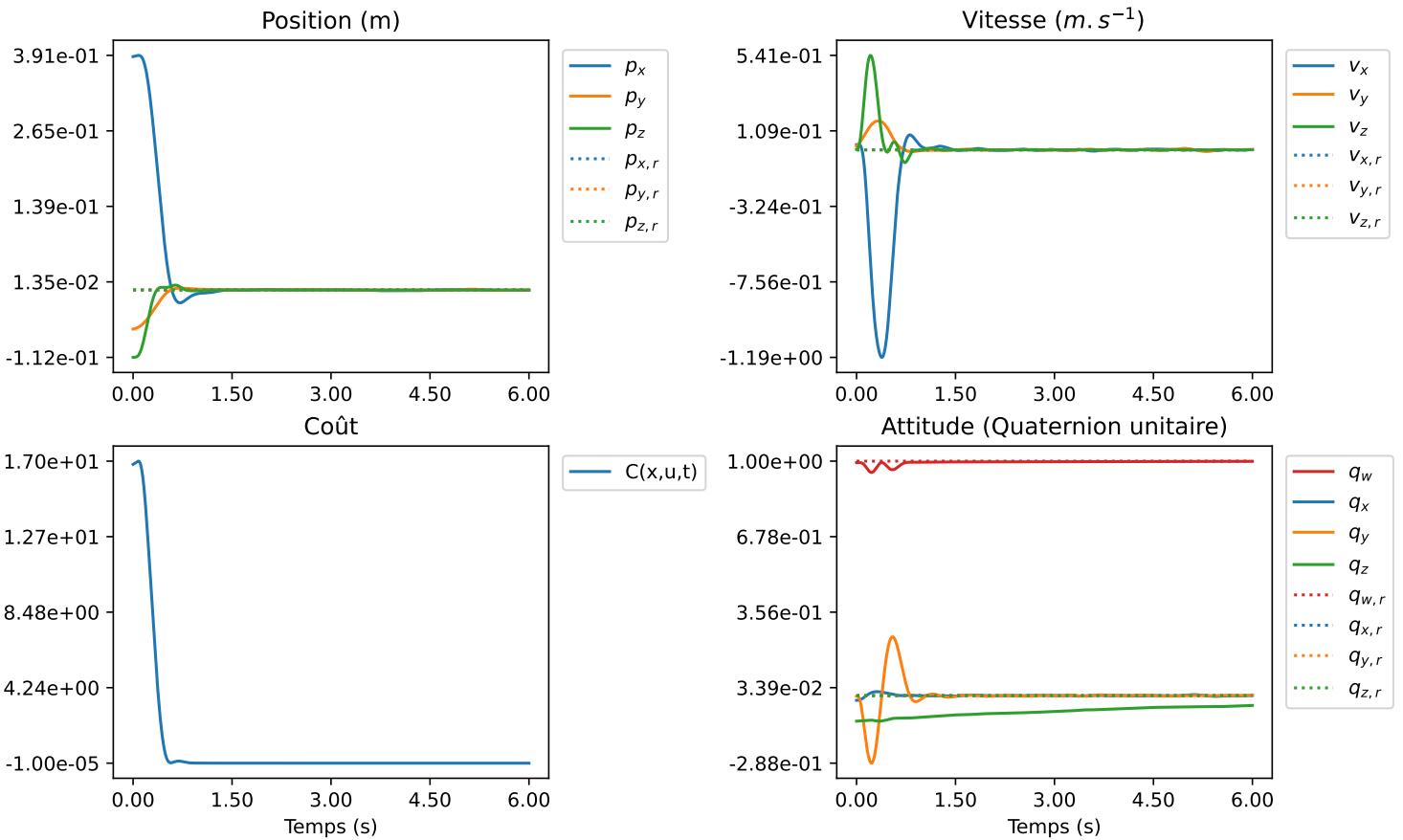


FIGURE 2.7 – Trajectoire de stabilisation avec un contrôle *MPC* en présence de perturbations. L'état de référence est indiqué par un trait pointillé et l'indice  $r$ . Quand cet état de référence n'est pas visible, c'est qu'il est couvert par la courbe de la trajectoire et que ceux-ci coïncident. À partir d'un état initial éloigné de l'état de référence, le contrôleur *MPC* permet un retour rapide à l'origine et une diminution rapide du coût optimisé, cela malgré la présence de perturbations dont on voit l'effet sur la vitesse et la position une fois le robot stabilisé.

La figure 2.8 présente la comparaison du contrôleur *MPC* avec le contrôleur proportionnel décrit en 2.4.3.2. Le contrôleur proportionnel possède lui aussi un retour d'état et est ainsi capable de compenser les perturbations et de ramener le robot à l'origine et d'atteindre un coût assez bas. Ce contrôleur proportionnel n'est cependant pas capable d'atteindre l'état de référence aussi tôt dans la trajectoire que le contrôleur *MPC*. La structure du contrôleur proportionnel où l'erreur en position permet de définir une orientation de référence qui permet de corriger cette erreur de position permet, comme le contrôleur *MPC* d'atteindre un état avec un coût faible nécessitant un passage par un état avec un coût plus élevé que l'état initial : on voit dans la figure 2.8 un phénomène similaire pour les coûts, ceux-ci augmentent un peu à partir de l'état initial avant de redescendre vers 0. Pour le contrôleur proportionnel, il ne s'agit pas de planification de la trajectoire, mais pour le contrôleur proportionnel, il ne s'agit pas de planification, mais de l'interaction entre les différents niveaux de consignes pour l'état (position > orientation > force des moteurs).

La figure 2.9 permet de vérifier que les analyses faites sur des trajectoires uniques restent valides sur un nombre plus important d'échantillons (20 répliques) d'états initiaux et les perturbations durant le vol. Les augmentations importantes de coûts, qui apparaissent pour le contrôleur proportionnel, correspondent aux situations pour lesquelles le robot a une vitesse initiale importante, ce qui a pour conséquence d'éloigner le robot de la position de référence et donc de faire augmenter le coût, que le contrôleur proportionnel ne réduit que lentement.

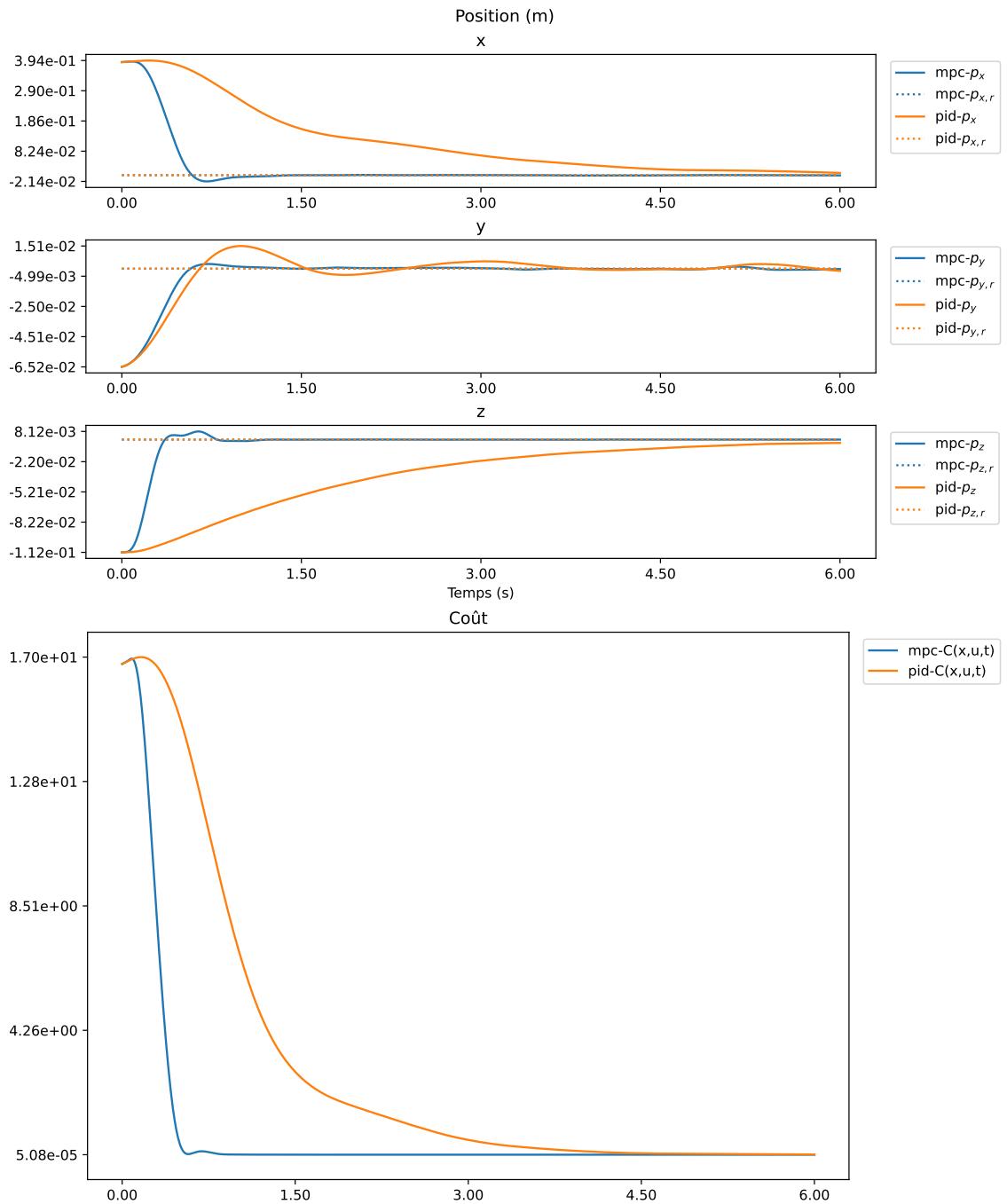


FIGURE 2.8 – Comparaison des positions et du coût pour des trajectoires d'un contrôleur proportionnel et un contrôleur *MPC* en présence de perturbations. L'état de référence est indiqué par un trait pointillé et l'indice *r*. Quand cet état de référence n'est pas visible, c'est qu'il est couvert par la courbe de la trajectoire et que ceux-ci coïncident. À partir d'un état initial perturbé, l'application du contrôle permet un retour à l'origine. Le retour est nettement plus rapide pour le contrôleur *MPC* que pour le contrôleur proportionnel.

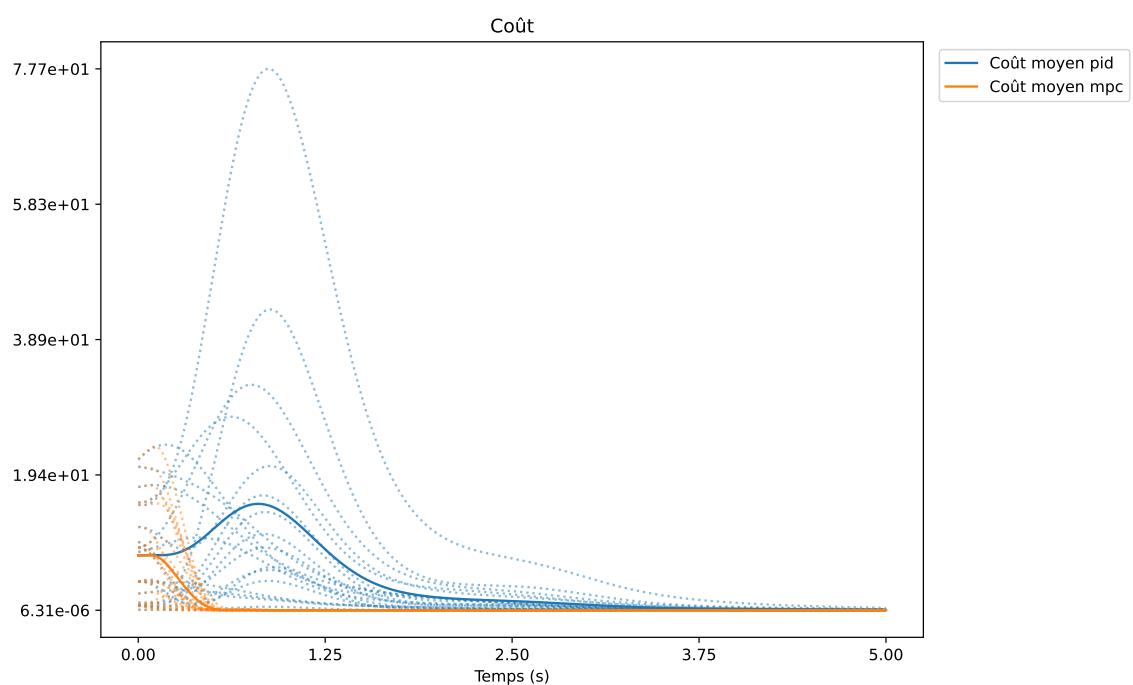


FIGURE 2.9 – Comparaison des coûts pour 20 trajectoires d'un contrôleur proportionnel et un contrôleur *MPC* en présence de perturbations. Les trajectoires individuelles sont représentées en pointillés pâles, et la valeur moyenne de ces trajectoires est en trait plein avec une couleur plus saturée. Les analyses faites sur des trajectoires uniques restent valides sur cet ensemble : le contrôleur *MPC* fait rapidement décroître le coût alors que le contrôleur proportionnel y parvient au bout d'un temps plus important.

#### 2.4.3.4 Suivi de trajectoire

Les contrôleurs basés contrôle optimal 2.4.3.1 et le contrôleur proportionnel 2.4.3.2 permettent de suivre des trajectoires qui ne sont pas des trajectoires de stabilisation, mais des trajectoires variables dans le temps. Ici je présenterais le suivi de trajectoires pour lesquelles le robot se déplace vers l'avant tout en décrivant des oscillations sur un ou deux des axes perpendiculaires au déplacement vers l'avant (e.g. haut/bas et droite/gauche). Cette classe de trajectoire particulière est choisie avant tout pour illustrer les capacités de contrôle, mais des oscillations similaires peuvent aider à l'estimation de la vitesse du robot par des capteurs de flux optique (RUFFIER et al. 2003). Plus précisément cette classe de trajectoire correspond aux références suivantes :

$$\begin{aligned} [\mathbf{p}_r]_{C_W}(t) &= \begin{pmatrix} a_x \sin(2\pi\omega_x t) \\ vt \\ a_z \sin(2\pi\omega_z t) \end{pmatrix} \\ [\mathbf{v}_r]_{C_W} &= \begin{pmatrix} a_x \omega_x \cos(2\pi\omega_x t) \\ v \\ a_z \omega_z \cos(2\pi\omega_z t) \end{pmatrix} \\ R_B^W{}_r &= I_3 \\ [\boldsymbol{\omega}_r]_{C_B} &= \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \end{aligned} \quad (2.25)$$

Dans ces équations, l'orientation et la vitesse angulaire spécifiées restent celles de la trajectoire de stabilisation. Il serait possible de calculer l'orientation que doit avoir le robot pour avoir la vitesse attendue à un instant  $t$ . J'ai préféré laisser cet aspect, outre le choix du lacet à la discréption des différents contrôleurs, au moins dans le cas d'un quadrotor. Le lacet (rotation autour de l'axe vertical) est en effet un paramètre libre dans le contrôle des quadrotor : il est possible de suivre une trajectoire pour laquelle la vitesse et la position sont fixées avec différentes valeurs de lacet.

Vu la trajectoire décrite par (2.25), les valeurs initiales de la vitesse peuvent être non nulles, ce qui pourrait déstabiliser certains contrôleurs. Pour éviter ces discontinuités, j'introduis une sorte d'amortissement sous la forme d'un temps ralenti en début et en fin de trajectoire. Je note  $\tau(t)$  une fonction représentant ce temps ralenti telle que

$$\tau(t) = \int_0^t \kappa(t) dt \quad (2.26)$$

Avec  $\kappa(t)$  la fonction qui détermine à quel point le temps est ralenti. Pour une trajectoire de durée  $T$ , j'ai choisi :

$$\kappa(t) = \begin{cases} \left(\frac{10t}{T}\right)^2 & \text{si } t < \frac{1}{10}T \\ \left(\frac{T-10t}{T}\right)^2 & \text{si } t > \frac{9}{10}T \\ 1 & \text{si } t \in [\frac{1}{10}, \frac{9}{10}] \end{cases} \quad (2.27)$$

$\kappa$  représente la vitesse d'écoulement de  $\tau$ , pendant le premier dixième de la trajectoire, celui-ci d'abord très lent puis de plus en plus rapide jusqu'à atteindre une vitesse d'écoulement "normale" de 1 puis ensuite ralentir de plus en plus vite une fois dans le dernier dixième de trajectoire. Avec

ce temps modifié, on peut introduire une nouvelle référence pour la position :

$$[\mathbf{p}_r]_{C_W}(t) = \begin{pmatrix} a_x \sin(2\pi\omega_x\tau(t)) \\ v\tau(t) \\ a_z \sin(2\pi\omega_z\tau(t)) \end{pmatrix} \quad (2.28)$$

$$[\mathbf{v}_r]_{C_W}(t) = \begin{pmatrix} a_x \omega_x \frac{d\tau}{dt} \sin(2\pi\omega_x\tau(t)) \\ v \frac{d\tau}{dt} \\ a_z \omega_z \frac{d\tau}{dt} \sin(2\pi\omega_z\tau(t)) \end{pmatrix} = \begin{pmatrix} a_x \omega_x \kappa(t) \sin(2\pi\omega_x\tau(t)) \\ v\kappa(t) \\ a_z \omega_z \kappa(t) \sin(2\pi\omega_z\tau(t)) \end{pmatrix} \quad (2.29)$$

en particulier

$$[\mathbf{v}_r]_{C_W}(0) = [\mathbf{v}_r]_{C_W}(T) = 0$$

et entre le premier dixième de trajectoire et le dernier, la vitesse et celle décrite par (2.25), avec un léger retard de  $\tau$  par rapport à  $t$ .

Vu la définition de  $\tau$  par une intégrale, en pratique, une trajectoire de référence est représentée par une séquence de couples  $(t_i, x_i)_i$  avec  $t_i$  le temps et  $x_i$  l'état de référence initialisé de la manière suivante :

$$\begin{aligned} t_0 &= 0 \\ \tau_0 &= 0 \\ [\mathbf{p}_r]_{C_W 0} &= 0 \\ [\mathbf{v}_r]_{C_W 0} &= 0 \end{aligned} \quad (2.30)$$

et mise à jour itérativement

$$\begin{aligned} t_i &= t_{i-1} + dt \\ \tau_i &= \tau_{i-1} + \kappa(t_i)dt \\ [\mathbf{p}_r]_{C_W i} &= [\mathbf{p}_r]_{C_W}(\tau_i) \\ [\mathbf{v}_r]_{C_W i} &= \frac{[\mathbf{p}_r]_{C_W}(\tau_{i+1}) - [\mathbf{p}_r]_{C_W}(\tau_{i+1})}{dt} \end{aligned} \quad (2.31)$$

Les figures suivantes présentent le résultat en simulation de l'utilisation des contrôleurs basés contrôle optimal 2.4.3.1 et du contrôleur proportionnel 2.4.3.2 pour le suivi de la trajectoire décrite plus tôt. Les valeurs d'amplitude, de fréquence et de vitesse sont les suivantes :

$$\begin{aligned} \omega_x &= 0.5s^{-1} \\ a_x &= 0.05m \\ \omega_z &= 0.2s^{-1} \\ a_z &= 0.05m \\ v &= 0.05m.s^{-1} \end{aligned} \quad (2.32)$$

Les perturbations durant la trajectoire et sur l'état initial sont identiques à celles utilisées pour la stabilisation.

La figure 2.10 montre que le contrôleur en boucle ouverte parvient à rejoindre la trajectoire de référence à partir d'un état initial différent de l'état initial de référence, en l'absence de perturbations. Après avoir rejoint la trajectoire de référence, il suit celle-ci de manière précise. Soumis à des perturbations, le contrôleur en boucle ouverte diverge comme il le fait pour la stabilisation 2.6. De manière similaire à la stabilisation, on voit via la figure 2.11 que le retour d'état permet au contrôleur MPC de suivre la trajectoire en présence de perturbations. La figure 2.12 permet de voir un avantage supplémentaire du contrôleur MPC et des contrôleurs basés contrôle optimal en général. On y voit en effet que le contrôleur proportionnel présente un retard net dans le suivi

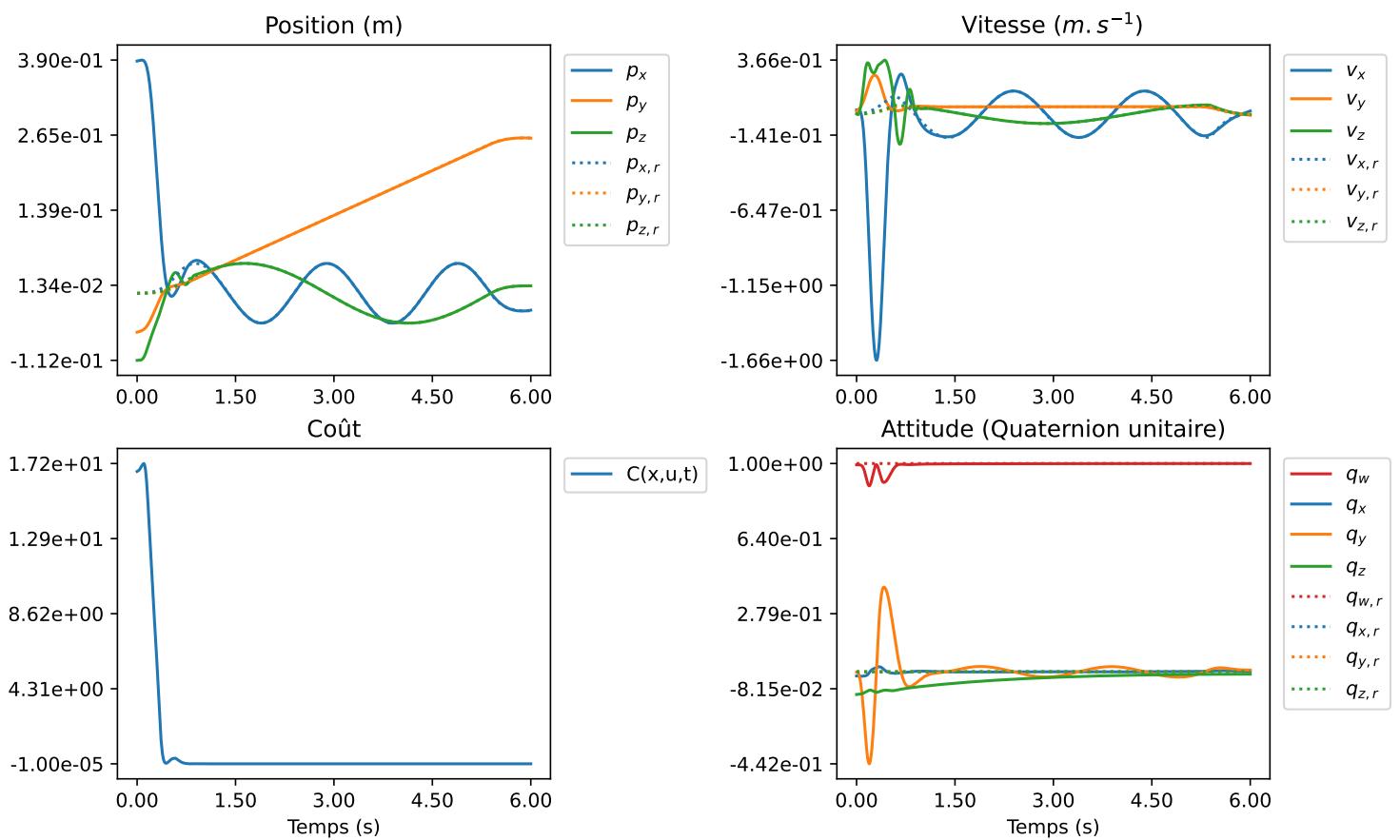


FIGURE 2.10 – Suivi d'une trajectoire avec un contrôle en boucle ouverte en l'absence de perturbations. L'état de référence est indiqué par un trait pointillé et l'indice  $r$ . Quand cet état de référence n'est pas visible, c'est qu'il est couvert par la courbe de la trajectoire et que ceux-ci coïncident. À partir d'un état initial éloigné de la trajectoire de référence, le robot rejoint rapidement la trajectoire de référence.

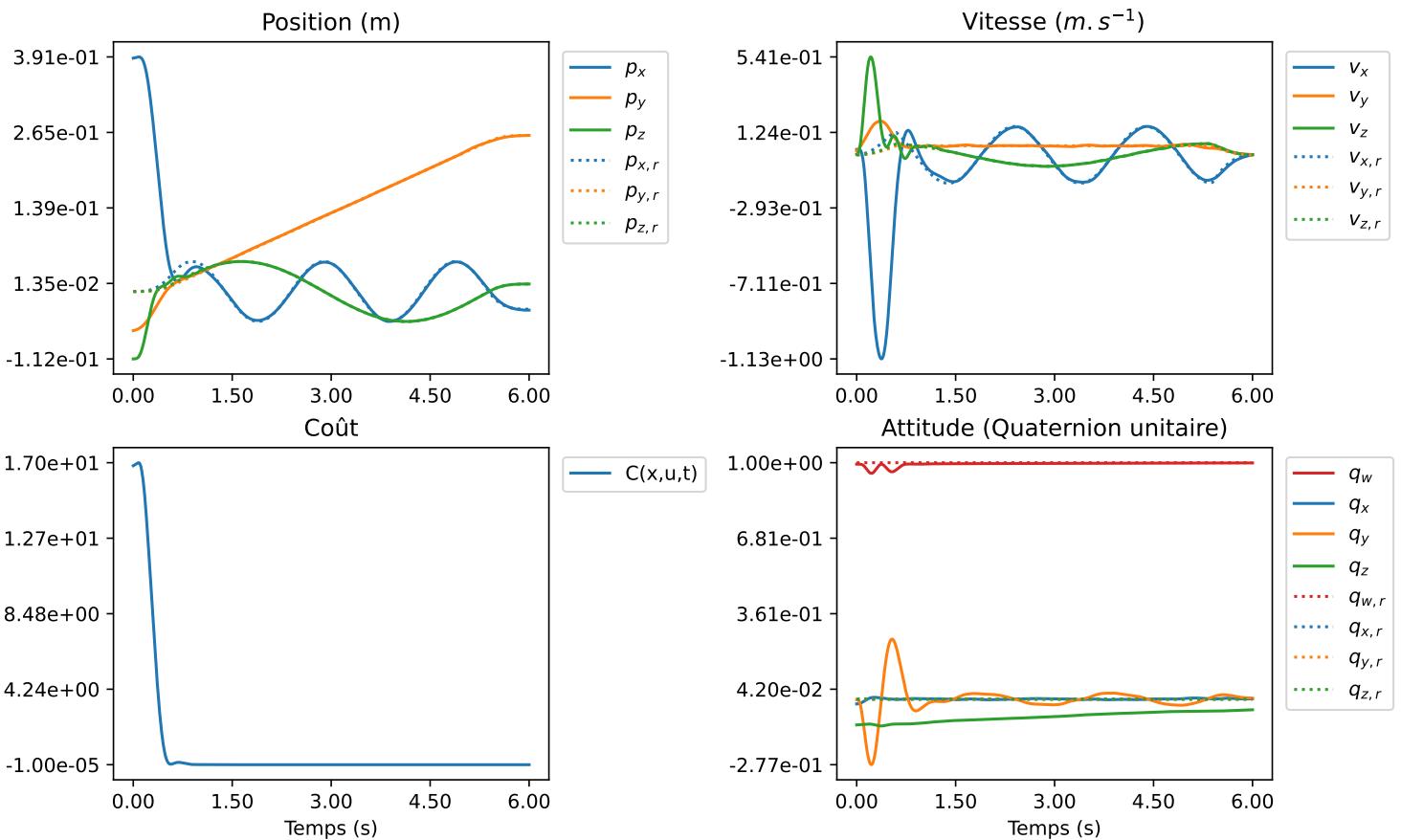


FIGURE 2.11 – Suivi d’une trajectoire avec un contrôle *MPC* en présence de perturbations. L’état de référence est indiqué par un trait pointillé et l’indice  $r$ . Quand cet état de référence n’est pas visible, c’est qu’il est couvert par la courbe de la trajectoire et que ceux-ci coïncident. À partir d’un état initial éloigné de la trajectoire de référence, le robot rejoint rapidement la trajectoire de référence, cela malgré la présence de perturbations.

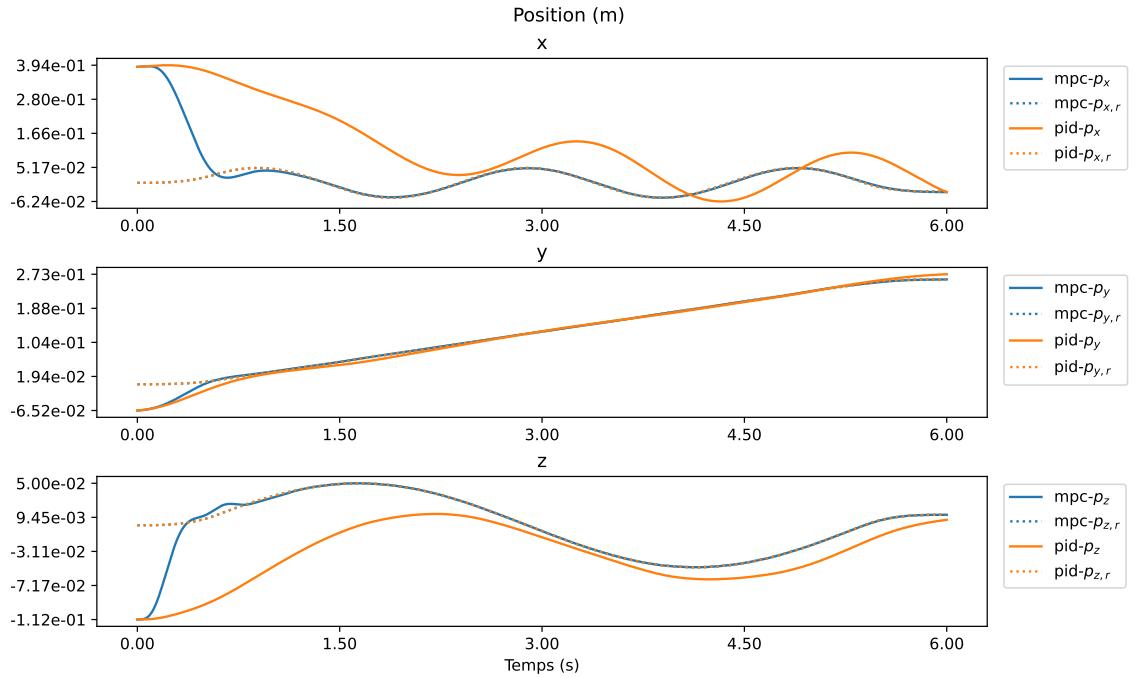


FIGURE 2.12 – Comparaison des positions pour le suivi d’une trajectoire par un contrôleur proportionnel et un contrôleur *MPC* en présence de perturbations. L’état de référence est indiqué par un trait pointillé et l’indice  $r$ . Quand cet état de référence n’est pas visible, c’est qu’il est couvert par la courbe de la trajectoire et que ceux-ci coïncident. À partir d’un état initial perturbé, le contrôleur *MPC* rejoint rapidement la position de référence. Le contrôleur proportionnel montre un retard et dépasse les positions de référence sur les axes  $x$  et  $z$ .

des oscillations sur l’axe  $x$  et un retard plus faible pour l’axe  $z$  alors que ce n’est pas le cas du contrôleur MPC. Ceci est permis par le fait que le contrôleur MPC planifie le futur proche de sa trajectoire et peut donc anticiper les changements de la trajectoire de référence. Le contrôleur proportionnel accès uniquement à l’état de référence pour l’instant suivant. Celui-ci contient bien la vitesse de référence que doit atteindre le robot pour pouvoir rester sur la trajectoire de référence, mais le contrôleur proportionnel ne parvient pour autant pas à éliminer ce retard complètement. La figure 2.13 montre cependant que le contrôleur proportionnel parvient à réduire de manière importante l’erreur, même si ce retard a pour conséquence des sursauts réguliers dans le coût. Le contrôleur MPC parvient à réduire le coût bien plus rapidement que le contrôleur proportionnel. La figure 2.14 confirme l’analyse sur une trajectoire unique. Le contrôleur proportionnel présente des augmentations régulières des coûts dues à un retard. On voit cependant que l’amplitude de ces augmentations semble être la même pour toutes les trajectoires une fois passé un certain temps. Quand le contrôleur proportionnel est parvenu à rejoindre la trajectoire de référence, son retard est similaire sur toutes les trajectoires, indépendamment de l’état initial.

## 2.5 Conclusion

En formulant le problème de détermination des commandes à donner au robot par un problème d’optimisation, sous contraintes de respect de la dynamique du robot, d’un coût dont les commandes sont les variables de décision (le problème de contrôle optimal 2.4), on peut obtenir une

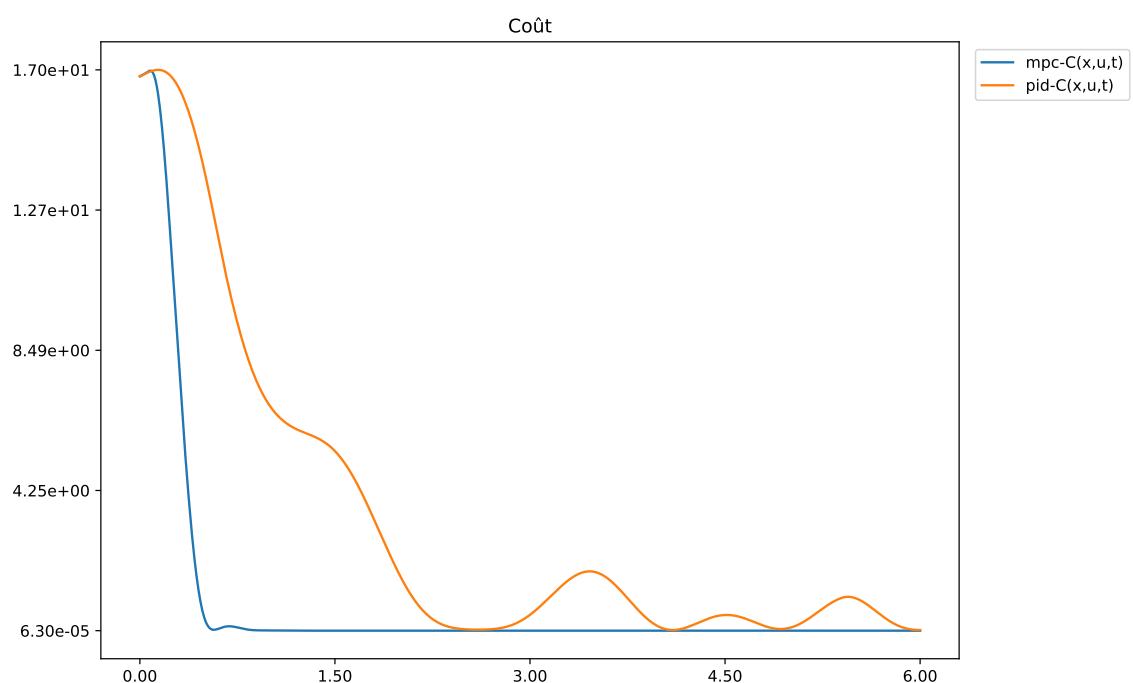


FIGURE 2.13 – Comparaison des coûts pour le suivi d'une trajectoire par un contrôleur proportionnel et un contrôleur *MPC* en présence de perturbations. À partir d'un état initial perturbé, le contrôleur *MPC* réduit rapidement le coût et le maintient à une valeur quasi nulle. Le contrôleur proportionnel parvient à maintenir borné le coût de la trajectoire, mais ne parvient pas à réduire celui-ci à un niveau similaire au contrôleur *MPC*

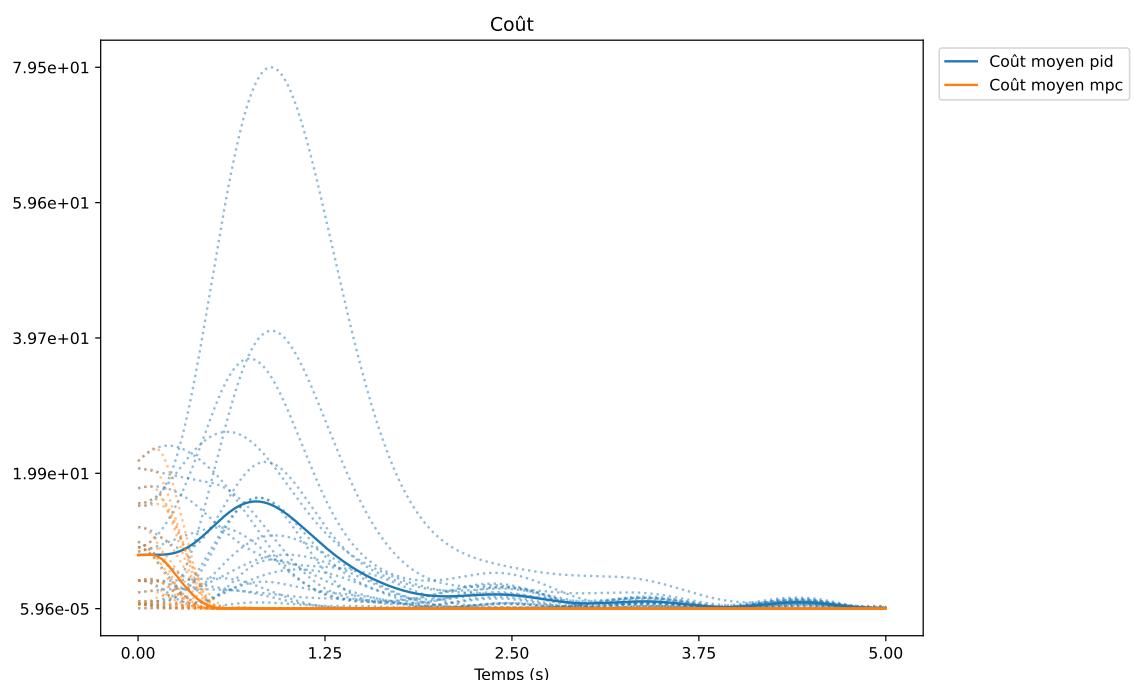


FIGURE 2.14 – Comparaison des coûts pour 20 trajectoires d'un contrôleur proportionnel et un contrôleur *MPC* en présence de perturbations. Les trajectoires individuelles sont représentées en pointillés pâles, et la valeur moyenne de ces trajectoires est en trait plein.. Les analyses faites sur des trajectoires uniques restent valides sur cet ensemble : le contrôleur *MPC* fait rapidement décroître le coût alors que le contrôleur proportionnel y parvient au bout d'un temps plus important et le retard entraîne des augmentations régulières du coût.

loi de commande pour le robot qui permet la stabilisation de celui-ci et le suivi d'une trajectoire préalablement spécifiée.

La prise en compte d'un horizon de plusieurs pas de temps dans la formulation du problème de contrôle optimal permet d'obtenir une solution qui ne soit pas totalement locale, mais le fait que cet horizon soit fini et les nécessaires approximations lors de la résolution empêchent que la solution soit un optimum global. Le contrôleur obtenu présente de meilleurs résultats qu'un contrôleur proportionnel.

Cependant, l'application en boucle ouverte des commandes solution du problème de contrôle optimal ne permet pas d'obtenir un contrôle robuste aux perturbations et effets externes non modélisés. L'utilisation d'une approche *Model Predictive Control* de résolution du problème de contrôle optimal à chaque nouveau pas de temps donne par contre une certaine robustesse du contrôleur aux perturbations non connues. Cependant, la résolution du problème de contrôle optimal peut s'avérer coûteuse en temps de calcul, que ce soit en boucle ouverte, mais plus encore avec le MPC.

De plus, certains écarts entre le modèle utilisé pour résoudre le problème du contrôle optimal et la dynamique réelle, ou même celle du simulateur, peuvent mener à des erreurs statiques. Une illustration de ce phénomène pour les drones pourrait être la présence d'un effet de sol, absent du modèle utilisé pour la résolution du problème de contrôle optimal. Dans un tel cas, si le contrôleur souhaite stabiliser le robot à une certaine altitude, celui-ci va calculer une commande qui fait décroître l'altitude du robot, mais arrivé assez près du sol, cette commande sera suffisante pour stabiliser le robot à une altitude supérieure à celle spécifiée au contrôleur. La solution obtenue par résolution du contrôle optimal planifiera une petite baisse de l'altitude qui, compte tenu de l'effet de sol, sera en fait un maintien de l'altitude actuelle.

En conséquence, la conception du modèle dynamique avec lequel est formulé le problème de contrôle optimal est particulièrement sensible. En particulier, pour le vol de drone dans des espaces confinés, de tels effets peuvent apparaître.

## Chapitre 3

# Perturbations à l'intérieur d'un tuyau

### 3.1 Introduction

Nous avons choisi de nous concentrer sur des tuyaux de forme cylindrique pour y faire voler un Crazyflie et plus particulièrement à des tuyaux dont le rayon est entre 20cm et 35cm. Les dimensions du Crazyflie, de l'ordre de 10cm de diagonale, sont assez importantes par rapport à celles du tuyau, ce qui ne laisse que peu de marge d'erreur pour la navigation et éviter les collisions avec l'environnement. La figure 3.1 illustre ces dimensions.

À notre connaissance, c'est le premier travail qui s'intéresse à de telles circonstances de vol, le rapport entre diamètre du tuyau et celui du quadrotor est entre 4 et 6. C'est un environnement très étroit.

Pour tenir compte de perturbations liées à l'interaction avec l'environnement, l'idéal serait de pouvoir mesurer ces perturbations durant le vol du robot selon différentes trajectoires afin de pouvoir obtenir un modèle dynamique de ces perturbations et fonction de l'historique de la vitesse, position et de l'orientation du robot.

Pour pouvoir conduire des mesures de perturbations durant le vol, l'approche la plus naturelle serait de mesurer la position et l'orientation du robot de manière très précise pour des vols dans le tuyau, puis de dériver ces mesures pour obtenir la vitesse, l'accélération et donc les forces qui s'exercent sur le robot. Conjointement, enregistrer les forces exercées par chacun des rotors afin d'obtenir les forces prévues par le modèle, puis de soustraire celles-ci aux forces mesurées afin d'obtenir les perturbations. Cette approche pose plusieurs problèmes.

Le premier de ces problèmes consiste à faire voler avec succès le Crazyflie à l'intérieur du tuyau. Les perturbations observées et présentées dans la section 3.4 rendent le vol dans le tuyau très instable. Un pilotage manuel permet d'obtenir des vols dans un tuyau de 65cm de diamètre, mais ceci est difficile pour les tuyaux plus petits (40cm et 50cm de diamètre) et le pilotage automatique mène à des crashes rapidement. Pour collecter des données de vol dans le tuyau, il faut d'abord savoir faire voler le robot dans le tuyau.

Un second problème réside dans le fait de réaliser une mesure précise de la position. Dans l'environnement du tuyau, l'utilisation des systèmes de capture de mouvement basés sur des caméras externes comme l'OptiTrack est difficile en raison de la difficulté à placer des caméras suffisamment



FIGURE 3.1 – Le Crazyflie dans un tuyau de 20cm de rayon.

espacées et ayant une ligne de vue sur le robot au vu des dimensions réduites de l'environnement. De plus, les rotors du Crazyflie ne donnent pas directement une mesure de leur vitesse de rotation, et donc de la force exercée, ce qui oblige à l'inférer à partir des commandes PWM utilisées pour contrôler ces moteurs, avec pour conséquence des délais potentiels importants dans l'estimation des forces (ANTONSSON 2015). En outre, la faible autonomie de la batterie du Crazyflie rend difficile la conduite d'une collecte de données en vol dans des conditions de vol comparables, en situation de batterie faible les capacités des moteurs du Crazyflie sont substantiellement modifiées, rendant l'estimation de la force exercée par chaque moteur plus difficile à réaliser.

Par ailleurs, en vol libre, les perturbations peuvent déstabiliser le drone, et provoquer une correction du contrôleur. Cette réaction, et c'est particulièrement le cas pour les multirotors qui ne sont pas complètement actionnés et doivent donc changer leur inclinaison pour compenser des perturbations horizontales, a une conséquence sur la direction et l'intensité des flux d'air créés par les rotors, et donc en retour, sur les perturbations elles-mêmes. En vol libre, ces perturbations sont donc le résultat de l'interaction entre l'environnement, le robot et son contrôleur, ce qui signifie que des données collectées avec un contrôleur pourraient perdre de leur pertinence une fois un autre contrôleur utilisé. L'intégration des connaissances des perturbations lors de la conception d'un contrôleur serait plus difficile à mener.

Pour ces raisons, et faute d'une autre solution pratique pour la mesure précise de la position et de l'orientation du robot dans le tuyau, nous avons opté pour une approche de mesure de perturbations en régime permanent uniquement. Plutôt que d'enregistrer les perturbations durant un vol, nous avons choisi d'enregistrer celles-ci en statique à une position fixée à l'intérieur du tuyau. Cette approche permet plusieurs choses :

- stabiliser le fonctionnement des moteurs à l'aide d'une alimentation externe ;
- utiliser un capteur de force 6 axes qui permet de mesurer directement les perturbations plutôt que de passer par deux dérivées numériques successives de la position et de l'orientation ;
- mesurer les forces qui s'exercent sur le robot en dehors de son interaction avec l'environnement du tuyau plutôt que de les calculer ;
- isoler les perturbations propres à l'environnement des perturbations dues à un écart entre le modèle du robot et le fonctionnement du matériel.

## Objectifs

Ce chapitre a pour objectif de décrire un protocole et un dispositif pour réaliser la collecte de données sur les perturbations présentes dans un tuyau, une méthode pour apprendre un modèle prédictif de ces perturbations à partir des données récoltées et d'intégrer ce modèle à une commande optimale.

## Plan

Dans ce chapitre, je présente d'abord quelques travaux traitant de la modélisation des perturbations dues à l'environnement, l'apprentissage de modèles de ces perturbations ou de la dynamique de drones et l'intégration de ces modèles à la commande de ces drones. Je décris ensuite le protocole utilisé pour la collecte des données de perturbations et leur traitement, puis une analyse des perturbations observées dans trois tuyaux de diamètre (40,50 et 65cm). Je détaille ensuite deux approches pour généraliser les mesures réalisées ponctuellement à l'ensemble du tuyau à

l'aide de réseaux de neurones et de processus gaussiens. J'aborde ensuite l'intégration de ces modèles prédisant les perturbations dans les contrôleurs basés commande optimale présentée dans la section 2.4.3.1. La collecte des données a été réalisée en grande partie par Lucien Renaud avec mon aide et celle de Jean-Baptiste Mouret. Le protocole de collecte et de traitement des données a été élaboré conjointement avec Lucien Renaud, et Jean-Baptiste Mouret et à la suite de discussions et d'essais successifs avec les instruments présentés plus loin.

## 3.2 Travaux connexes

### Modélisations de l'effet de sol

La plus connue des perturbations dues à l'environnement pour les robots à rotors est l'effet de sol. L'étude de celui-ci par CHEESEMAN et BENNETT 1955 sur un hélicoptère est ancienne et présente une approximation de cet effet selon la vitesse transversale, l'altitude et les caractéristiques de l'hélicoptère. DANJUN et al. 2015 adaptent ce modèle à une architecture de type quadrotor et proposent d'intégrer un mécanisme de compensation de cet effet à un contrôleur PID. SANCHEZ-CUEVAS, HEREDIA et OLLERO 2017 étudient cet effet de sol pour le cas particulier de quadrotor pour lesquels uniquement certains des moteurs sont sujets à cet effet, spécialement quand le terrain sous le quadrotor n'est pas plat et chacun des rotors ne se trouve donc pas à la même distance du sol. Les auteurs présentent une méthode pour quantifier cet effet de sol partiel et détaillent des pistes pour intégrer sa prise en compte dans un contrôleur. POWERS et al. 2013 décrivent un effet similaire : l'effet de plafond qui crée une aspiration pour un quadrotor qui serait proche du plafond de son environnement. SHUKLA et KOMERATH 2018 présentent une étude des interactions et caractéristiques des flux d'air créés par les 4 moteurs d'un quadrotor à l'aide de visualisations par Vélocimétrie par Image de Particule (PIV).

### Apprentissage de modèle des perturbations

De nombreuses approches pour la quantification de telles perturbations se reposent sur la capacité à collecter des données et à apprendre ou chercher un modèle dans une classe de modèles génériques. SHI et al. 2019 apprennent ainsi un réseau de neurones qui modélise la force résiduelle (qui est observée en plus de ce que prévoit un modèle standard de la dynamique) subie par un quadrotor dans un contexte d'effet de sol. Ils mobilisent pour cela les données des capteurs à bord ainsi que celles récoltées par un système de capture de mouvement. BANSAL et al. 2016 présentent deux réseaux de neurones modélisant respectivement les forces et les moments qui s'exercent sur un quadrotor à partir de données collectées par les capteurs embarqués et ceux d'un système de capture de mouvement (VICON). LAMBERT et al. 2019 apprennent un modèle complet de la dynamique d'un quadrotor à partir uniquement des données collectées par les capteurs embarqués. MAHE, PRADALIER et GEIST 2018 présentent une comparaison des capacités d'un réseau de neurones et de l'approche ARX pour modéliser la dynamique de vitesses d'un quadrotor. RICHARD et al. 2019 comparent l'efficacité de différentes architectures de réseau de neurones (perceptions multicouches, réseaux convolutionnels, réseaux récurrents) pour modéliser la dynamique des vitesses sur un quadrotor simulé, un quadrotor réel et un navire simulé.

## **Estimation en ligne des perturbations**

En parallèle des approches qui apprennent des modèles à partir de données hors ligne, certains travaux estiment directement les perturbations dues à l'environnement en ligne. PI, YE et CHENG 2021 réalisent ainsi l'estimation en temps réel des forces non modélisées, mais observées via les accéléromètres de la centrale inertie. ZHAO, HUGHES et D. M. LYONS 2020 présentent une méthode d'apprentissage supervisé pour estimer la présence d'un autre drone au dessus d'un quadrotor à partir des données de centrale inertie. L'approche présentée est testée hors ligne, mais est évaluée pour la détection avec des données d'un historique court (1s). HUGHES et D. LYONS 2021 proposent une approche pour détecter la présence et la direction de murs par un quadrotor à l'aide de forêts aléatoires et à partir d'une estimation de son orientation et des mesures de sa centrale inertie. NAKATA et al. 2020 mettent au point un capteur de pression qui permet la détection d'obstacles via la perturbation des flux d'air générés par un quadrotor à proximité de ces obstacles.

## **Intégration des modèles de perturbations dans les contrôleurs**

Une fois modélisée ou estimée en ligne, l'inclusion des informations sur les perturbations dues à l'environnement peut être réalisée de plusieurs manières. SHI et al. 2019 intègrent la prédiction des forces résiduelles dans leur contrôleur comme un terme feed forward, de manière analogue à la gravité. PI, YE et CHENG 2021 ajoutent une entrée à un contrôleur appris, contenant l'orientation nécessaire pour compenser la perturbation estimée dans l'état courant. BANSAL et al. 2016 utilisent le modèle appris pour générer des trajectoires admissibles par le robot. LAMBERT et al. 2019 se servent du modèle appris dans une commande prédictive qui s'appuie sur un échantillonnage en parallèle puis une sélection de la meilleure trajectoire parmi celles simulées par le modèle appris.

## **Conclusion**

De nombreux travaux ont été réalisés sur les perturbations liées aux robots à voilure tournante, mais à notre connaissance aucun n'aborde le cas spécifique de telles perturbations dans un environnement comme les tuyaux. Une fois modélisées, plusieurs approches ont déjà été explorées pour intégrer ces perturbations à la boucle de contrôle. Le formalisme de la commande prédictive permet cette intégration de manière aisée, et c'est cette approche que nous avons suivie.

### **3.3 Protocole de mesure et de traitement des perturbations dans un tuyau**

#### **3.3.1 Collecte des données**

Pour réaliser les mesures des perturbations dans les tuyaux, nous avons utilisé un capteur de forces 6 axes ATI FTN-Nano17 dont la sensibilité sur chacun des axes est la suivante :

$$F_x, F_y, F_z \quad \tau_x, \tau_y, \tau_z$$

$$\frac{1}{320} N \quad \frac{1}{64} N.mm$$

Ce capteur n'est pas saturé pour des mesures inférieures aux valeurs suivantes :

$$\begin{array}{lll} F_x, F_y & F_z & \tau_x, \tau_y, \tau_z \\ 12N & 17N & 120N.mm \end{array}$$

Les forces créées par les moteurs du Crazyflie sont de l'ordre de 0.16N par moteur, soit moins de 1N en tout. Le poids du Crazyflie est de l'ordre de 0.5N. Les valeurs que l'on s'attend à mesurer sont donc largement dans le domaine de mesure du capteur. La sensibilité du capteur est inférieure à la sensibilité théorique de contrôle : les rotors du crazyflie sont contrôlés par des signaux PWM sur 16 bits. Un incrément de 1 dans le signal de contrôle pourrait donc théoriquement faire varier l'intensité de la force créée par un rotor d'approximativement  $3\mu N$ , c'est-à-dire une valeur bien plus faible que la capacité de discrimination du capteur. Mais de telles variations sont très faibles devant les forces mises en jeu durant un vol. Ainsi la sensibilité du capteur est très importante devant les forces en présence et est donc suffisante. Nous avons choisi de réinitialiser le zéro de référence régulièrement durant le protocole de mesure afin de contrer cette dérive.

Cependant, les mesures des capteurs de forces ont tendance à dériver, à cause de la manière dont elles sont conduites. Les mesures effectuées sont entachées d'un biais qui varie assez lentement dans le temps (ANDRADE CHAVEZ 2019). Cette dérive est, entre autres, due à l'échauffement des pièces qui composent le capteur suite à leur activité de mesure. La figure 3.2 illustre cette dérive pour le capteur utilisé.

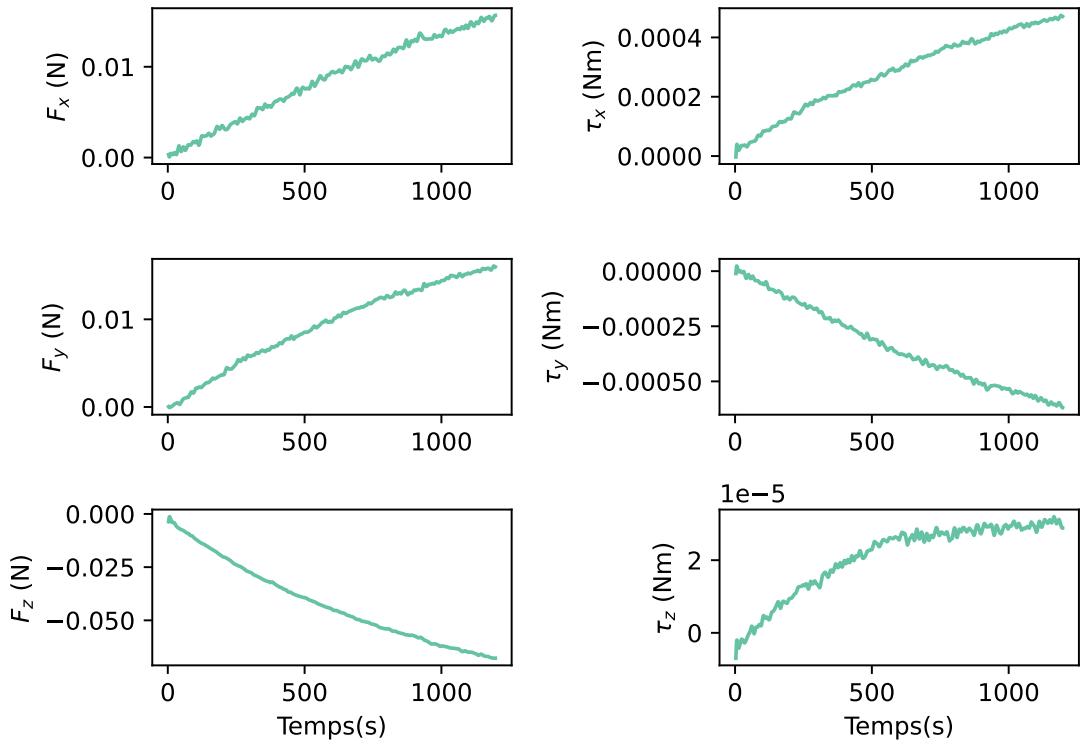
Afin de conduire de manière automatisée des mesures des perturbations subies par le drone à différentes positions à l'intérieur du tuyau, nous avons fixé le drone sur le capteur de forces 6 axes, lui-même fixé sur un bras robot (Franka Emika Panda). Ce bras robotisé permet de gérer le placement du robot et du capteur dans le tuyau de manière précise, reproductible et sans intervention humaine. Le capteur est orienté de manière à ce que son axe z soit aligné avec la verticale du repère inertiel vers le haut, l'axe y avec l'axe du tuyau et l'axe x avec l'axe transverse du tuyau. La figure 3.3 montre le dispositif de mesure comprenant un Crazyflie, le capteur de forces 6 axes, le bras robot et le tuyau.

Le protocole détaillé de collecte des mesures qui a été utilisé pour la récolte des données est décrit par l'algorithme 2.

Ce protocole permet de réaliser des mesures qui tiennent compte uniquement des perturbations liées à la présence du robot à l'intérieur du tuyau en mesurant la différence entre les forces exercées à l'étape 3 (en air libre, sans influence de l'environnement) et à l'étape 7 (dans le tuyau, avec influence de l'environnement). Le fait de tarer le capteur, à l'étape 4, permet de lutter contre la dérive de celui-ci, chaque mesure ne durant quelques secondes, la dérive attendue pour le capteur est faible. Les mesures réalisées à l'étape 4, dont on s'attend à ce que la valeur moyenne soit nulle puisque le capteur vient d'être taré, permettent d'estimer la variance des forces qui s'exercent afin de fournir un point de comparaison entre la situation à l'extérieur et à l'intérieur du tuyau et d'apporter un éclairage sur une possible dynamique des perturbations, malgré une position fixe du robot et la commande constante des rotors.

### 3.3.2 Traitement des données brutes collectées

Le capteur de forces 6 axes ATI FTN-Nano17 transmet les valeurs de forces et couples mesurées à un ordinateur à travers une carte d'acquisition Ethernet via un flux UDP à la fréquence de 7kHz.



### Dérive des capteurs au repos

FIGURE 3.2 – Évolution des mesures réalisées par le capteur de forces 6 axes au cours du temps et en l'absence de variation des forces exercées. On peut observer une dérive des mesures alors même que les forces exercées restent constantes. Cette dérive est due à l'échauffement du capteur après des mesures prolongées.

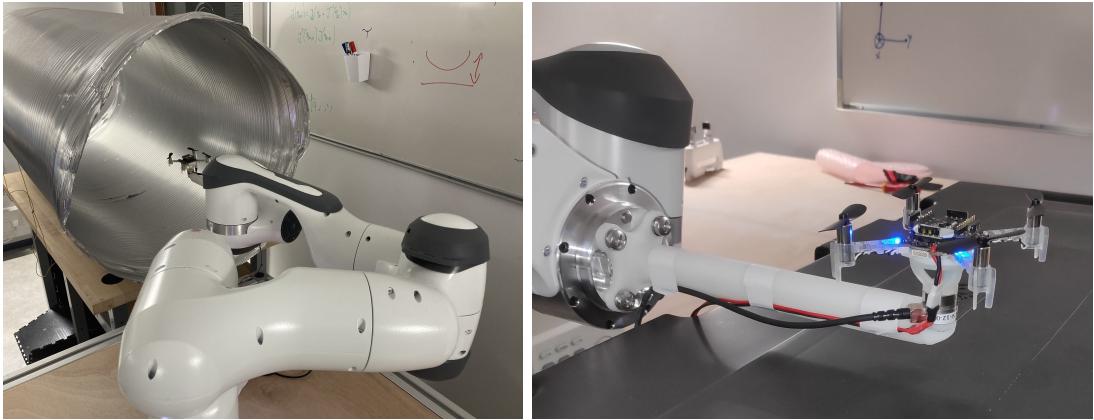


FIGURE 3.3 – À gauche le dispositif de mesure complet avec le robot positionné à l'entrée du tuyau. À droite le Crazyflie fixé sur le capteur de force lui-même fixé sur le bras robot. On distingue les câbles pour la transmission des données (noir) et ceux pour l'alimentation du robot (rouge et noir).

---

**Algorithme 2** Protocole de mesure des perturbations aérodynamiques statiques générées dans un tuyau

---

1. Le bras positionne le robot à l'extérieur du tuyau, en air libre
  2. Les rotors du drone sont allumés, reçoivent une commande identique et constante, choisie de manière à correspondre à un régime de fonctionnement qui compense la gravité et permet le vol stationnaire.
  3. Temps d'attente de quelques secondes pour permettre la stabilisation du système
  4. On tare le capteur : les mesures du capteur seront maintenant exprimées par rapport aux valeurs mesurées à l'extérieur du tuyau. Les valeurs mesurées à l'extérieur du tuyau deviennent donc le zéro de référence.
  5. Temps d'attente 5 secondes pendant lequel des mesures de référence sont conduites à l'extérieur du tuyau.
  6. Le bras positionne le robot à l'intérieur du tuyau, à la position voulue (ce positionnement dure au moins 1.7s).
  7. Les mesures des forces et couples sont effectuées pendant 10 secondes à l'intérieur du tuyau.
  8. Retour au point 1 pour effectuer une nouvelle mesure.
- 

Ces données brutes se révèlent extrêmement bruitées. Ce bruit correspond en fait aux vibrations importantes du robot et de l'ensemble du système de mesure sous l'effet des moteurs faisant tourner les rotors. La figure 3.4 présente une séquence de mesures sur les 6 axes réalisée selon le protocole 2. Cette figure montre que les mesures brutes des forces et couples sont largement supérieures à celles attendues. Les valeurs mesurées sont même supérieures aux valeurs théoriques possibles pour les moteurs du crazyflie. Malgré ce bruit on observe un changement de régime manifeste un peu après 5s, c'est-à-dire le moment où on place le robot dans le tuyau.

Pour analyser de manière plus quantitative les données sur les perturbations récoltées, il est donc nécessaire de les filtrer afin d'éliminer le bruit. On utilise pour cela une implémentation numérique d'un filtre passe-bas de Butterworth (BUTTERWORTH 1930) d'ordre 4. Ce filtre élimine les signaux haute fréquence, dont les vibrations, qui sont considérées comme du bruit dans le cadre de ces mesures de perturbations statiques, tout en préservant les signaux dont la fréquence est en dessous de la fréquence de coupure. Ce filtre est utilisé sur la séquence des données brutes puis sur la séquence filtrée préalablement inversée (les mesures de la fin au début et réciproquement) et le résultat est remis dans le bon ordre. Ce filtrage en sens direct puis en sens inverse permet d'éliminer le retard est appelé *forward-backward filtering* (SMITH 2007) et correspond à l'application d'un filtre d'un ordre double du filtre utilisé, ici cela donne filtre d'ordre 8. Ce filtre est non causal : l'application sur la séquence inversée utilise des informations provenant du "futur" de la séquence. Une telle technique ne peut donc pas être réalisée en temps réel, mais uniquement a posteriori sur une séquence complète de mesures.

Pour obtenir des mesures statiques des perturbations à un endroit donné du tuyau, on passe les mesures brutes réalisées via le protocole 2 dans la séquence de traitement décrite par l'algorithme 3.

L'application des protocoles de mesure 2 et de traitement 3 sur des séquences réalisées à diffé-

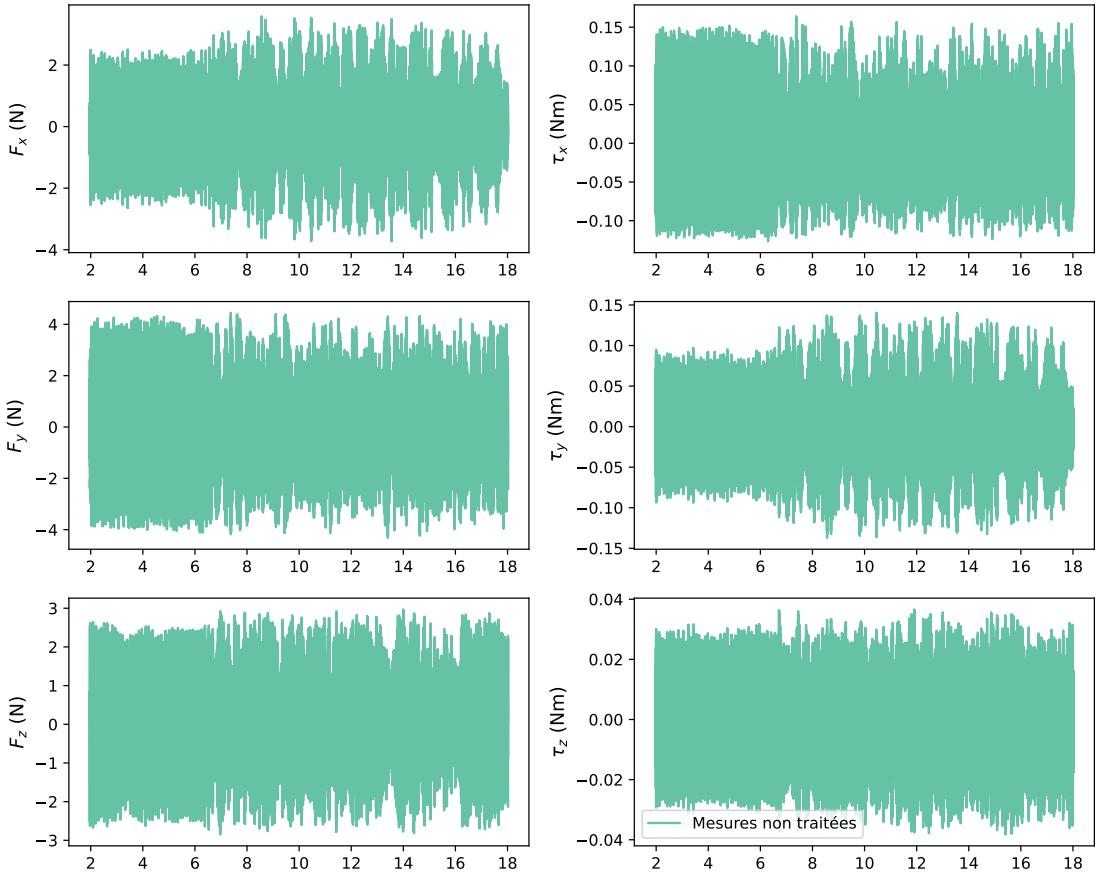


FIGURE 3.4 – Les mesures brutes effectuées par le capteur quand les moteurs du Crazyflie tournent. À gauche, de haut en bas les forces selon les axes x, y, et z en N. À droite de haut en bas, les couples selon les axes x, y, z en N.m. L’axe des abscisses représente le temps en secondes. Jusqu’à 5s le robot est en dehors du tuyau puis il est déplacé à l’intérieur.

---

### Algorithme 3 Séquence de traitement des mesures

---

- Définition de la moyenne des signaux bruts entre  $t=2s$  et  $t=4s$  comme valeur de référence
  - Soustraction la valeur de référence à toute la séquence de mesure
  - Application forward-backward d’un filtre de Butterworth d’ordre 4, fréquence de coupure 1Hz, sur la séquence de mesures compris entre 8s et 18s
  - Extraction dans cette séquence filtrée de la période [9s, 17s] afin d’éviter les effets de bord dus à l’application du filtre sur une séquence finie
  - Calcul de la moyenne de la séquence extraite, cette valeur est la mesure des perturbations statiques pour la localisation où ont été faites les mesures brutes
  - Calcul de la covariance de la séquence extraite pour estimer l’intensité de la dynamique des perturbations
-

rentes localisations d'un même tuyau, et sur des tuyaux de différents diamètres permet d'obtenir une carte des perturbations statique dans un tuyau.

La figure 3.5 présente une séquence de mesures réalisées avec le protocole 2 filtrée avec 3 filtres de Butterworth différents : avec une fréquence de coupure à 1Hz, une à 10Hz et une à 50Hz. De manière attendue, la séquence filtrée à 1Hz est plus régulière que celle filtrée à 10Hz qui est elle-même plus régulière que celle filtrée à 50Hz. Sur ces versions filtrées, les ordres de grandeur des forces et couples sont dans le domaine de ce que peut produire le Crazyflie. Avec les 3 versions, on observe un net changement de régime après que le Crazyflie soit placé dans le tuyau (début du mouvement à 5s, fin un peu avant 8s). Les mesures effectuées présentent une variance plus importante une fois dans le tuyau. Cette variance supplémentaire n'est pas celle de la vibration du robot due à l'activité des moteurs. Si c'était le cas, on pourrait l'observer aussi pendant la phase où le robot n'est pas dans le tunnel. Cette variance correspond donc à un environnement plus turbulent à l'intérieur du tuyau. On peut observer aussi que les mesures présentent un net décalage dans leur valeur moyenne une fois le robot placé dans le tuyau.

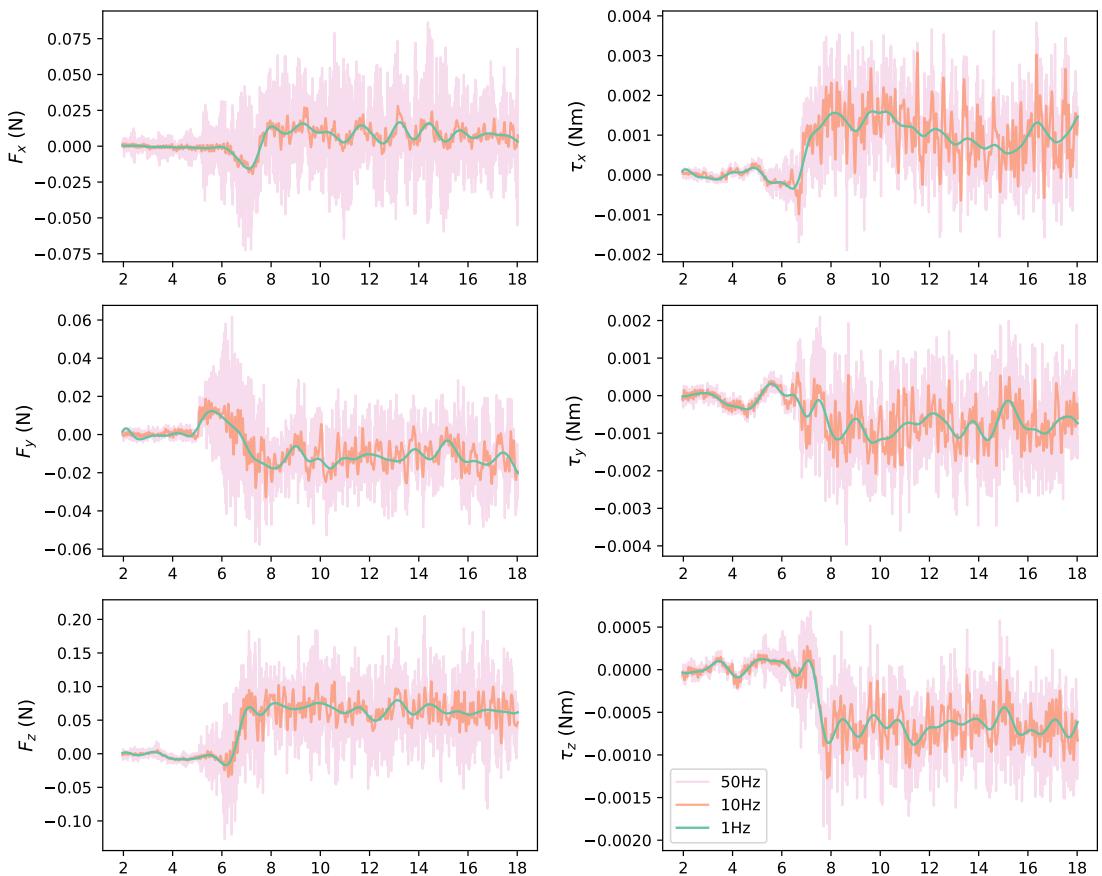


FIGURE 3.5 – Mesures du capteur 6 axes filtrées à différentes fréquences. La séquence de la figure 3.4 est passée à travers un filtre de Butterworth à différentes fréquences de coupure. Durant cette séquence, le robot est en air libre avant 5s puis est placé dans le tuyau. À gauche, de haut en bas les forces selon les axes x, y, et z en N. À droite de haut en bas, les couples selon les axes x, y, z en N.m. L'axe des abscisses représente le temps en secondes.

### 3.4 Analyses des perturbations observées dans le tuyau

Vu la symétrie de la géométrie du tuyau, nous nous sommes intéressés plus particulièrement aux perturbations en force selon les axes z et x, c'est-à-dire selon la section transverse du tuyau. Pour ces mêmes raisons de symétrie, les mesures réalisées dans le tuyau le seront sur une unique section de ce tuyau : je n'aborderais pas la variation de ces mesures le long de l'axe du tuyau. De telles variations existent probablement, particulièrement à proximité des extrémités du tuyau, mais en première approche, et pour des raisons pratiques de temps nécessaire à la réalisation de telles mesures et de capacité du bras robot à se positionner loin à l'intérieur du tuyau, celles-ci seront ignorées.

Dans ce contexte, pour représenter les résultats d'une série de mesures à différentes localisations dans le tuyau, j'utiliserais des figures présentant une section du tuyau contenant des flèches représentant le vecteur de perturbations selon les axes x et z. L'origine de chaque flèche correspondant à l'endroit où la séquence de mesures correspondant à la perturbation représentée a été effectuée. La figure 3.6 présente les mesures de perturbations statiques sur 3 tuyaux de diamètres respectifs 40cm, 50cm et 65cm.

Sur la figure 3.6 on peut observer dans les trois tuyaux des schémas similaires. Tout d'abord un effet de répulsion du sol qui n'est présent que dans la partie du tuyau la plus proche du sol. Cet effet de sol présente des intensités de perturbation plutôt importantes par rapport aux perturbations dans le reste du tuyau. Outre cet effet de sol, les autres perturbations présentent une composante verticale orientée vers le bas dans la grande majorité des localisations. Un point particulièrement intéressant est l'existence de perturbations qui sont dirigées vers les parois dans la moitié supérieure du tuyau. Ces aspirations, contrairement à l'effet de sol, ont pour conséquence de déstabiliser d'autant plus le robot puisqu'elles le déplacent vers la paroi, augmentant le risque de collision et menant le robot dans une zone plus perturbée encore. Dans ces schémas, on retrouve une symétrie axiale par rapport à l'axe vertical. Cette symétrie est conséquence de la combinaison de la symétrie centrale du tuyau, et de la direction dans laquelle sont orientés les rotors du robot, vers le bas.

La figure 3.7 présente les mêmes mesures sur lesquelles a été ajoutée une ellipse représentant la covariance des perturbations mesurées pendant les 10s de la mesure. Les dimensions et l'orientation de ces ellipses sont déterminées par la matrice de covariance de la mesure. Cette covariance représente à la fois l'incertitude qu'il est possible d'accorder à chaque point de mesure, mais aussi l'intensité de la part dynamique des perturbations mesurées dans le tuyau. On observe en particulier que cette covariance est plus importante dans le tuyau de 40cm de diamètre, en particulier dans la zone située juste au-dessus de l'effet de sol. Par ailleurs cette covariance ne semble pas corrélée à l'intensité moyenne de la perturbation, c'est-à-dire la longueur de la flèche. Ceci suggère que cette covariance serait moins une incertitude sur la mesure que la part dynamique des perturbations causées par le robot.

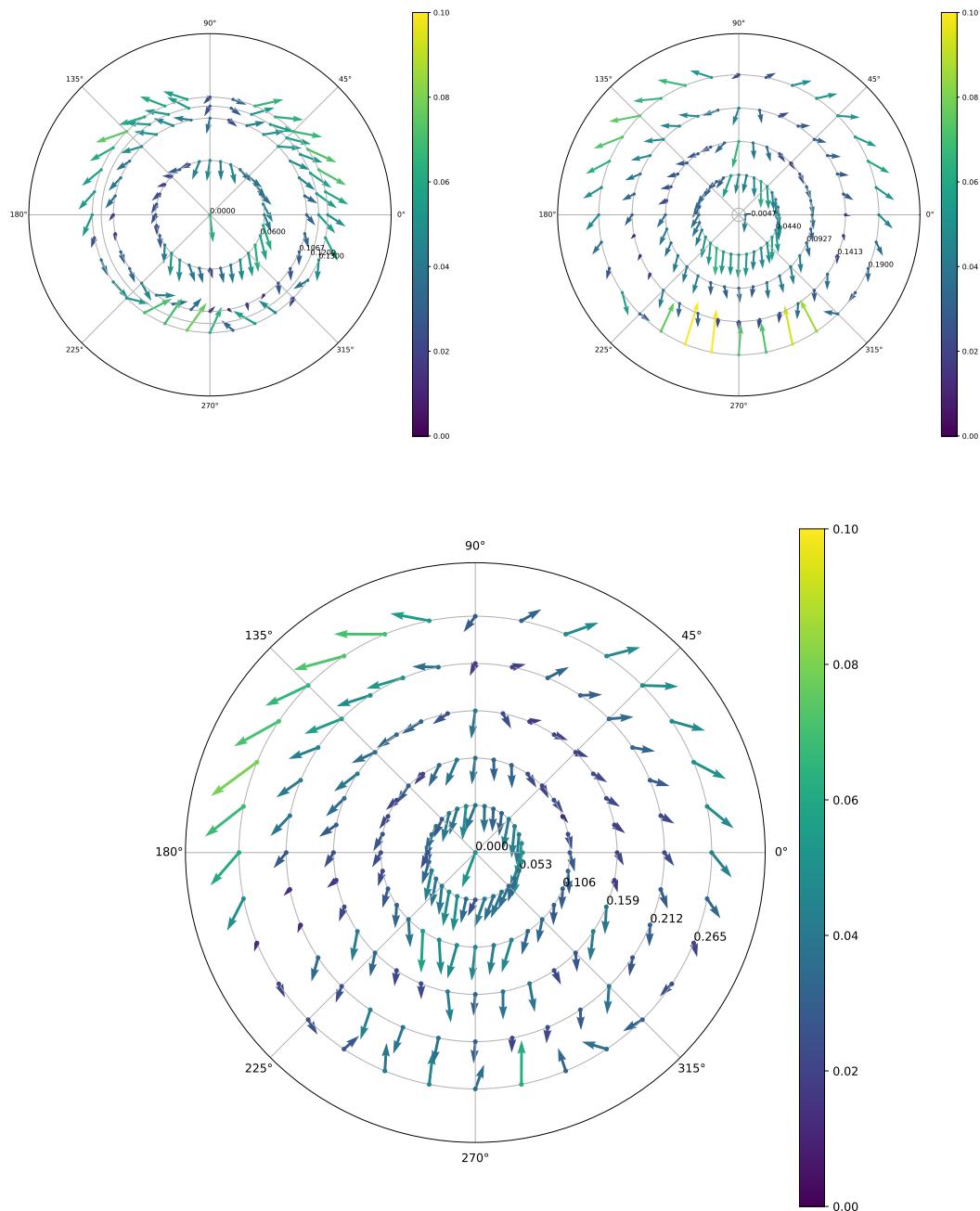


FIGURE 3.6 – Représentation graphique des mesures des perturbations statiques dans 3 tuyaux de diamètres respectivement 40cm, 50cm et 65cm. Ces mesures sont représentées dans une section de tuyau orthogonale à l'axe du tuyau, c'est-à-dire dans le plan (x,z). L'axe x correspond à l'horizontale et z la verticale. L'origine des flèches correspond au lieu de la mesure, et les composantes des flèches représentent les forces selon les axes x et z dans le tuyau. Les flèches sont représentées dans une couleur qui varie avec l'intensité de la perturbation selon l'échelle présentée à droite. Cette échelle est en N.

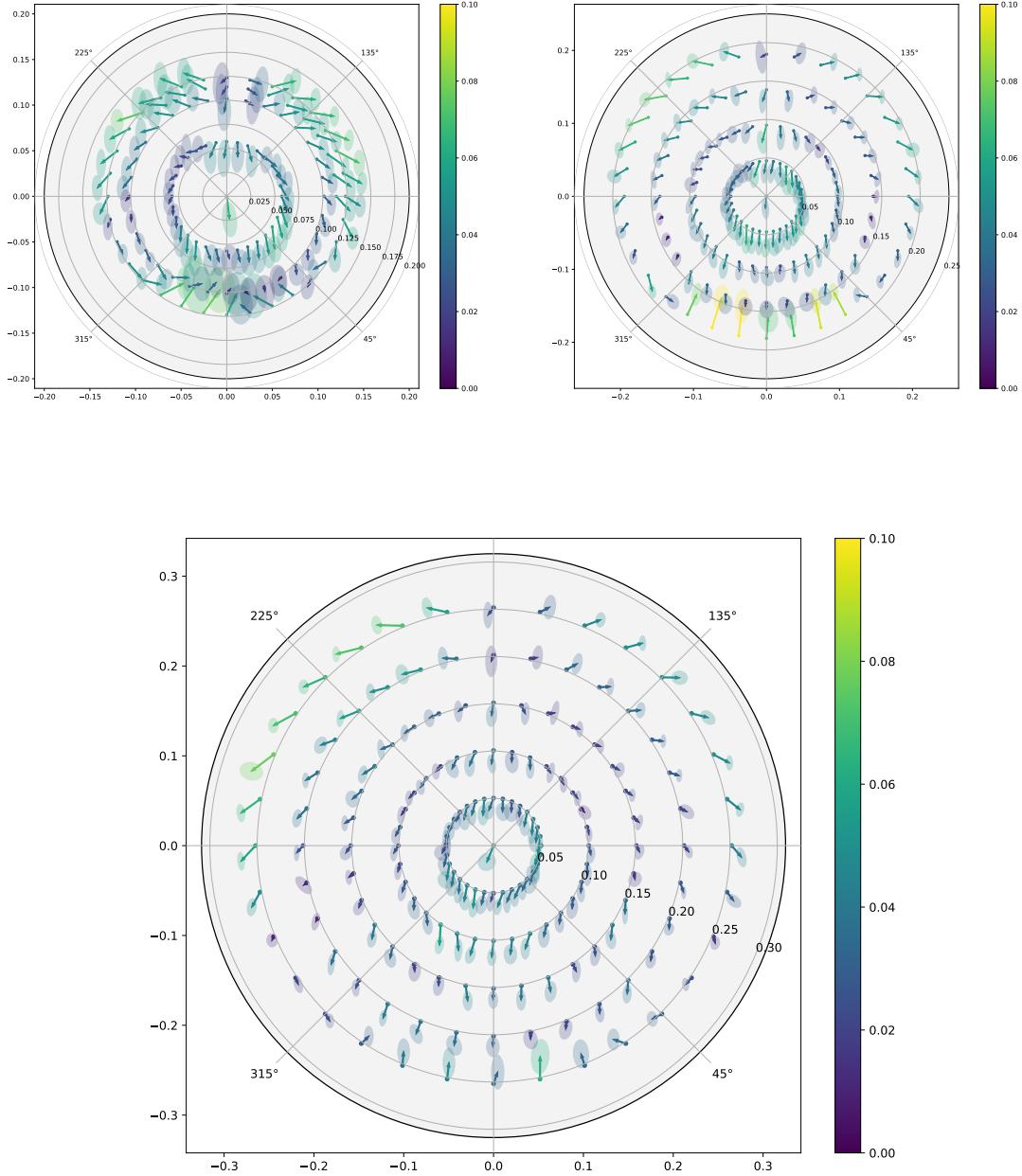


FIGURE 3.7 – Représentation graphique des mesures des perturbations statiques dans 3 tuyaux de diamètres respectivement 40cm, 50cm et 65cm avec covariance. Ces mesures sont représentées dans une section de tuyau orthogonale à l'axe du tuyau, c'est-à-dire dans le plan (x,z). L'axe x correspond à l'horizontale et z la verticale. L'origine des flèches correspond au lieu de la mesure, et les composantes des flèches représentent les forces selon les axes x et z dans le tuyau. Les flèches sont représentées dans une couleur qui varie avec l'intensité de la perturbation selon l'échelle présentée à droite. Cette échelle est en N. Autour de chaque tête de flèche une ellipse présente la covariance de la perturbation mesurée à  $1\sigma$ .

## 3.5 Apprentissage de modèles pour extrapoler les mesures

Malgré la répartition voulue uniforme des mesures, celles-ci demeurent ponctuelles, en particulier dans le tuyau de 40cm pour lequel les dimensions du dispositif de mesure empêchent de réaliser des mesures proches des parois. Pour pouvoir utiliser ces mesures dans un contrôleur comme ceux décrits dans la section 2.4.3.1, il est nécessaire de pouvoir déterminer les perturbations en chaque point du tuyau, et donc d'être capable de généraliser les mesures réalisées à tout le tuyau.

C'est l'objet de cette section dans laquelle je présente plusieurs méthodes et modèles pour inférer une estimation des perturbations entre les points de mesure à l'aide des données récoltées et traitées selon les modalités décrites dans les sections 3.3 et 3.3.1.

Je commence d'abord par une brève présentation des deux classes de modèles considérées pour l'apprentissage de ce modèle : les processus gaussiens et les réseaux de neurones. Je décris ensuite des transformations des données utilisées pour tirer parti des symétries de l'environnement et faciliter l'apprentissage. Je présente les performances de ces différentes transformations et architectures sur des jeux de données collectées dans des tuyaux de 40, 50 et 65cm de diamètre. Je termine par une analyse du champ de perturbations extrapolé par le meilleur modèle.

### 3.5.1 Apprentissage avec des processus gaussiens

Les processus gaussiens (RASMUSSEN et C. K. I. WILLIAMS 2006) sont une classe de modèles non paramétriques consistant en une collection non dénombrable de variables aléatoires gaussiennes, ou autrement dit, une fonction aléatoire dont l'entrée correspond à l'indice de la collection. On peut représenter un processus gaussien  $f$  par une fonction moyenne  $m$  et une fonction symétrique positive, la covariance  $k$  telles que pour deux entrées  $\mathbf{x}$  et  $\mathbf{x}'$  :

$$\begin{aligned} f(\mathbf{x}) &\sim \mathcal{N}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x})) \\ \text{cov}(f(\mathbf{x}), f(\mathbf{x}')) &= k(\mathbf{x}, \mathbf{x}') \end{aligned} \tag{3.1}$$

Avec ce formalisme, la valeur prédite par le processus gaussien,  $f(\mathbf{x})$ , est un scalaire. Des formulations existent pour de la régression sur des fonctions à valeur vectorielle (BONILLA, CHAI et C. WILLIAMS 2008), mais dans ce travail j'utiliserais simplement un processus gaussien différent par dimension à apprendre.

Apprendre un modèle avec un processus gaussien revient à conditionner le processus gaussien par des données observées. Dans le contexte actuel, je vais utiliser des processus gaussiens avec pour entrée la position du robot et comme image la perturbation mesurée à la position fournie en entrée.

Échantillonner un processus gaussien conditionné par des données qui sont un ensemble d'entrées  $X = (\mathbf{x}_i)$  et les sorties associées  $Y = (\mathbf{y}_i)$  revient à échantillonner selon une loi normale dont la moyenne est une combinaison linéaire des sorties  $Y$  et la covariance une combinaison linéaire des covariances entre le point auquel on échantillonne et les entrées connues  $X$ .

$$\begin{aligned} \mathbb{E}(f(\mathbf{x})|X, Y) &= (k(\mathbf{x}, \mathbf{x}_i))_i K_X^{-1}(Y - \mathbb{E}(f(X))) \\ \text{Var}(f(\mathbf{x})|X, Y) &= k(\mathbf{x}, \mathbf{x}) - (k(\mathbf{x}, \mathbf{x}_i))_i K_X^{-1}(k(\mathbf{x}, \mathbf{x}_i))_i^T \\ K_X[i, j] &= k(\mathbf{x}_i, \mathbf{x}_j) \end{aligned} \tag{3.2}$$

Un avantage des processus gaussiens, en particulier par rapport aux réseaux de neurones, est l'existence de cette covariance. Celle-ci peut être utilisée comme une mesure de l'incertitude

du modèle concernant les prédictions faites. Pour modéliser les perturbations, j'utiliserais comme fonction de covariance le classique noyau exponentiel (RASMUSSEN et C. K. I. WILLIAMS 2006) :

$$k(\mathbf{x}, \mathbf{x}') = \sigma e^{-\frac{|\mathbf{x} - \mathbf{x}'|^2}{l}} \quad (3.3)$$

$\sigma$  et  $l$  sont des paramètres de cette fonction.  $\sigma$  contrôle l'écart à la moyenne : plus  $\sigma$  est grand plus les prédictions ont une probabilité élevée de s'écartez de la moyenne.  $l$  contrôle la distance à laquelle les échantillons connus affectent les prédictions faites : plus  $l$  est faible moins un échantillon à contraindre la prédiction. Leur valeur sera optimisée de manière à maximiser la probabilité des données observées, c'est-à-dire des mesures collectées.

Les processus gaussiens sont des modèles de régression très efficaces et largement utilisés, cependant le coût computationnel pour calculer une prédiction est dépendant du nombre d'échantillons (NGUYEN, FILIPPONE et MICHIARDI 2019) (quadratique en le nombre d'échantillons pour une prédiction, cubique en le nombre total d'échantillons pour en intégrer de nouveaux) et peut donc être inadaptés à des jeux de données trop grands.

### 3.5.2 Apprentissage avec des réseaux de neurones

Les réseaux de neurones artificiels (HAYKIN 1999) permettent l'apprentissage de modèles pour la régression avec des jeux de données très grands (HE et al. 2016), et leur coût de prédiction est contrôlé par leur architecture et leurs paramètres plutôt que par la taille du jeu de données sur lequel est réalisé l'apprentissage.

Un réseau de neurones consiste en un ensemble de neurones connectés les uns aux autres en réseau. Il s'agit d'un graphe orienté dans lequel chaque neurone :

- reçoit en entrée un ensemble de nombres provenant des neurones qui lui sont connectés ;
- réalise une combinaison affine de ces entrées ;
- leur applique une fonction non linéaire ;
- envoie le résultat à tous les neurones auquel il est connecté.

Les neurones isolés comme celui représenté sur la figure 3.8 sont organisés en réseau en connectant la sortie de neurones à l'entrée d'autres neurones. Un réseau possède des entrées et des sorties qui sont respectivement l'entrée de certains neurones et la sortie d'autres neurones. Les entrées du réseau ne proviennent pas de neurones à l'intérieur du réseau, mais servent d'entrée aux arguments du réseau, vu comme une fonction. Il en est de même pour les sorties du réseau qui permettent d'identifier les différentes composantes de l'image du réseau de neurones, vu comme une fonction.

Une organisation classique des réseaux de neurones appelée *perceptron multicouche* regroupe les neurones dans des couches. En plus des couches d'entrée et de sortie, un perceptron multicouche se compose d'une séquence de couche dites cachées. Dans chaque couche, tous les neurones connectent leur sortie à chaque neurone de la couche suivante et reçoivent comme entrée la sortie de chaque neurone de la couche précédente.

La couche d'entrée est la première couche. Chacun de ses neurones ne possède qu'une entrée. Ces entrées forment les entrées du réseau. Chacun des neurones de la couche d'entrée connecte sa sortie à l'ensemble des neurones de la première couche cachée.

La couche de sortie est la dernière couche. Les sorties des neurones de la couche de sortie sont les sorties du réseau. Chacun des neurones de la couche reçoit en entrée l'ensemble des sorties des neurones de la dernière couche cachée.

Le fonctionnement d'un perceptron multicouches est illustré dans la figure 3.9 et formalisé par les équations (3.4). Les données d'entrée du réseau, sont notées  $x$ , les paramètres, c'est-à-dire

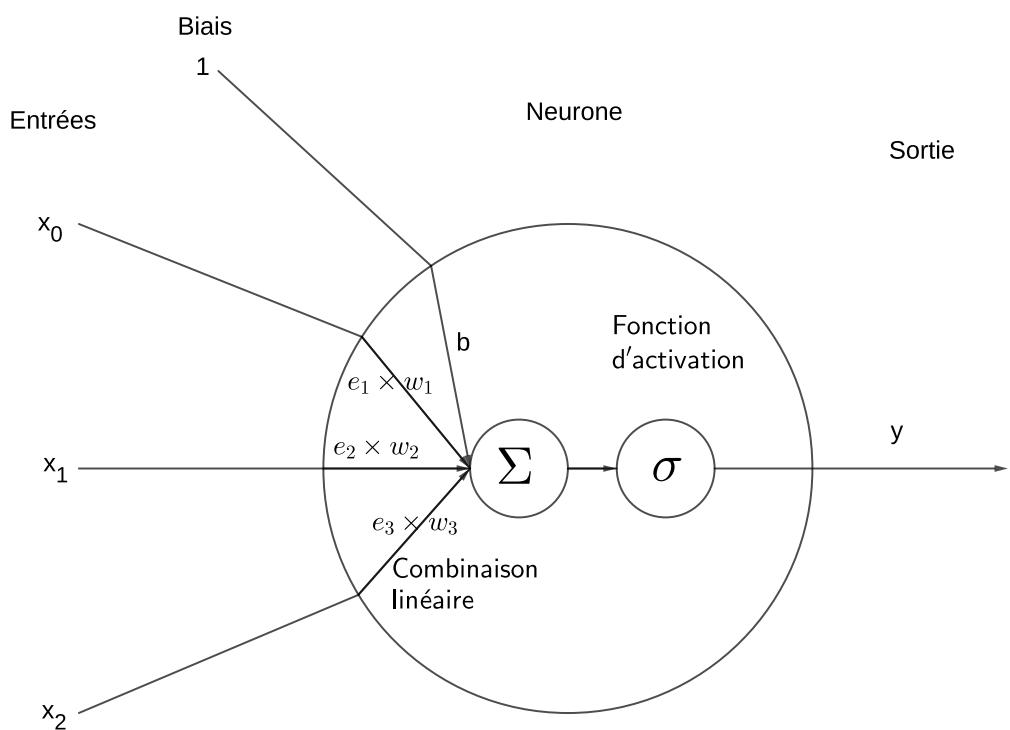


FIGURE 3.8 – Illustration du fonctionnement d'un neurone isolé. Les coefficients de la combinaison linéaire sont notés  $w_i$  et la fonction non linéaire d'activation  $\sigma$ , et le biais  $b$  correspond à la partie affine de la combinaison. Celui-ci est présenté comme une entrée comme les autres dont la valeur vaut toujours 1 et dont le poids peut être ajusté.

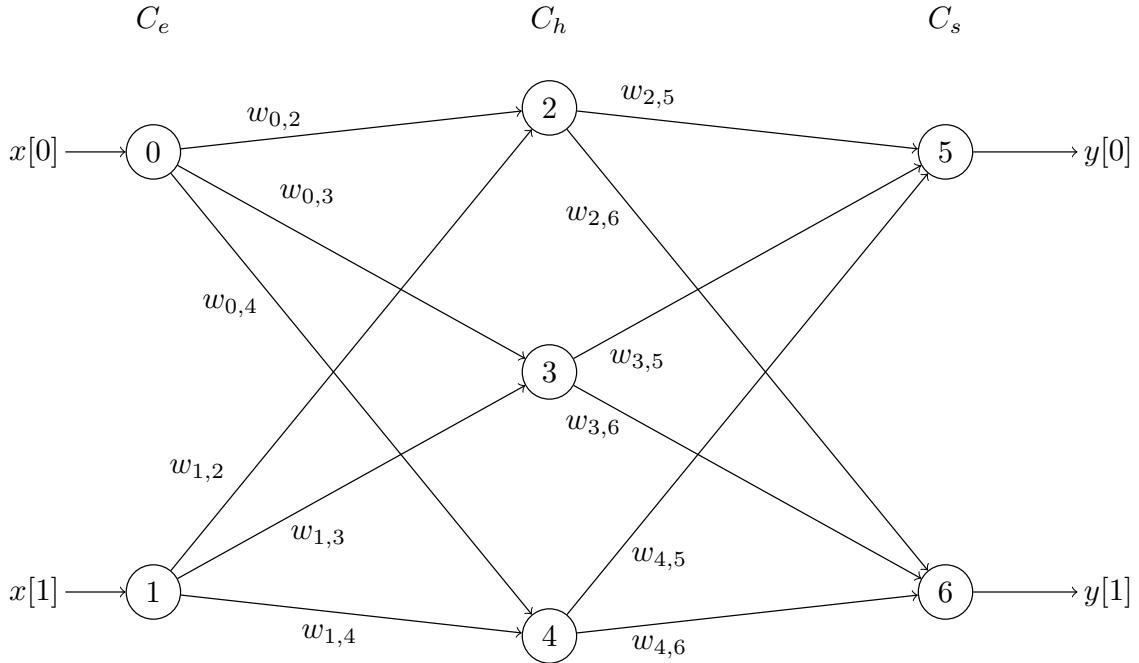


FIGURE 3.9 – Une représentation d'un réseau complètement connecté ou perceptron multi-couches (RUMELHART, HINTON et R. J. WILLIAMS 1986) avec une entrée et une sortie de taille 2 et une couche cachée de taille 3.  $C_e$  est la couche d'entrée,  $C_h$  une couche cachée et  $C_s$  la couche de sortie. La sortie d'un neurone  $i$  est l'image par  $\sigma$  de la combinaison linéaire des sorties des neurones de la couche précédente. Les coefficients de cette combinaison linéaire sont les poids  $w$ . Sur les arcs reliant les neurones on affiche les poids  $w$  qui constituent les matrices  $W$ . Les biais  $b$  ne sont pas représentés pour ne pas surcharger la figure.

les poids des combinaisons affines réalisées par les neurones artificiels,  $\theta$  et la sortie, c'est-à-dire l'image de l'entrée  $x$ , et notée  $y$ . La fonction non linéaire, appelée fonction d'activation est notée  $\sigma$ .

$$\begin{aligned}
 x_0 &= x \\
 x_{i+1,j} &= \sigma(W_{i,j}x_i + b_{i,j}) \quad \forall i \in [0, n-2], j \in [0, c_i - 1] \\
 y_j &= x_{n,j} = W_{n-1,j}x_{n-1} + b_{n-1,j} \quad j \in [0, c_n - 1] \\
 y &= (y_j)_{j \in [0, c_n - 1]} \\
 \theta &= (W_{i,j}, b_{i,j})_{i \in [0, n-1], j \in [0, n_i - 1]}
 \end{aligned} \tag{3.4}$$

La recherche de paramètres connaissant un ensemble de données d'apprentissage est rendue efficace par l'algorithme de rétropropagation du gradient de l'erreur (RUMELHART, HINTON et R. J. WILLIAMS 1986; LECUN et al. 2012) qui explore l'espace des paramètres en suivant le gradient de l'erreur c'est-à-dire la direction dans l'espace des paramètres pour laquelle l'erreur diminue le plus.

Pour la régression des modèles de perturbations, j'ai choisi d'utiliser un taux d'apprentissage variable contrôlé par l'algorithme ADAM (KINGMA et BA 2015).

Pour apprendre les perturbations statiques mesurées dans le tuyau, je dispose de quelques centaines de points, en conséquence, j'ai choisi une architecture avec une seule couche cachée

de 16 neurones afin de garder le nombre de paramètres du modèle assez faible. Une fonction ReLU (RAMACHANDRAN, ZOPH et LE 2017) est utilisée comme fonction activation en fin de couche cachée. Comme le précise l'équation 3.4, la couche de sortie n'utilise pas de fonction d'activation. La fonction de coût utilisée est l'erreur quadratique moyenne.

### 3.5.3 Transformation des mesures de perturbation pour faciliter l'apprentissage

Avant d'entraîner des réseaux de neurones et processus gaussiens sur les données, je procède à plusieurs transformations des données. Une première transformation est motivée par la propriété de symétrie observée dans la section 3.4 et justifiée par les symétries de l'environnement. Si on suppose que cette symétrie doit être présente dans les perturbations, on peut enrichir les données mesurées par leur symétrique : pour une mesure  $(f_x, f_z)$  réalisée au point  $(x, z)$  on peut ajouter la mesure  $(-f_x, f_z)$  au point  $(-x, z)$  à l'ensemble des mesures réalisées, doublant ainsi le nombre de mesures. Cette symétrie n'est bien entendu pas parfaitement observée dans les mesures, cf. la figure, 3.6 mais elle est motivée par la symétrie de l'environnement et donc, en première approximation, des causes physiques des perturbations.

Une seconde transformation concerne la représentation de la position où a été faite la mesure. L'utilisation d'une représentation polaire, malgré ses propriétés de symétrie adaptées au problème, peut poser des problèmes vis-à-vis de la continuité de la représentation de l'angle. Les positions  $(r, \epsilon)$  et  $(r, 2\pi - \epsilon)$  sont géométriquement très proches, mais leur représentation ne l'est pas. Une représentation cartésienne de la position aurait cet avantage de continuité, mais pas la propriété de symétrie. Je me propose aussi de tester une représentation hybride combinant les propriétés de symétrie de la représentation polaire à celle de continuité de la représentation cartésienne en remplaçant dans la représentation polaire l'angle par son sinus et son cosinus. Ceci correspond à une représentation cartésienne normalisée :  $\mathbf{p} = (r, \sin(\theta), \cos(\theta))$ .

L'architecture complètement connectée du réseau de neurones utilisé pour la régression a comme conséquence une mise en commun des représentations intermédiaires apprises par le réseau de neurones. Contrairement aux processus gaussiens pour lesquels chaque dimension de la fonction prédite l'est de manière indépendante des autres, les sorties du réseau de neurones ne sont pas indépendantes les unes des autres. L'architecture utilisée pour le réseau de neurones mutualise les représentations apprises dans les couches cachées qui sont utilisées par tous les neurones de sorties et sont donc modifiées par l'erreur de ceux-ci. Apprendre à prédire l'ensemble des données mesurées ( $f_y, \tau_x, \tau_y$  et  $\tau_z$  en plus de  $f_x$  et  $f_z$ ) pourrait apporter des a priori supplémentaires au réseau de neurones, d'une manière similaire à l'utilisation de l'hypothèse de symétrie.

En plus d'un changement de représentation de la position, je procède à une standardisation des mesures. Ceci correspond à une transformation linéaire des données de manière à ce que celles-ci aient une moyenne nulle et une variance unitaire. Ce traitement est appliqué aussi bien aux données d'entrée (position) qu'aux données de sorties (perturbations selon les axes x et z). Étant donné un ensemble de  $n$  échantillons  $X = (\mathbf{x}_i)_{i \in [1, n]}, \mathbf{x}_i \in \mathbb{R}^d$  et l'étiquette correspondante

$Y = (\mathbf{y}_i)_{i \in [1, n]}$ ,  $\mathbf{y}_i \in \mathbb{R}^p$  on construit d'abord un ensemble de données centrées :

$$\begin{aligned}\tilde{X} &= \{\tilde{\mathbf{x}}_i = \mathbf{x}_i - \frac{1}{n} \sum_{j=1}^n \mathbf{x}_j \mid i \in [1, n]\} \\ \tilde{Y} &= \{\tilde{\mathbf{y}}_i = \mathbf{y}_i - \frac{1}{n} \sum_{j=1}^n \mathbf{y}_j \mid i \in [1, n]\}\end{aligned}\tag{3.5}$$

Puis on divise celles-ci par leur variance :

$$\begin{aligned}\hat{X} &= \{\hat{\mathbf{x}}_i = \frac{\tilde{\mathbf{x}}_i}{\sqrt{\frac{1}{n} \sum_{j=1}^n \tilde{\mathbf{x}}_j^2}} \mid i \in [1, n]\} \\ \hat{Y} &= \{\hat{\mathbf{y}}_i = \frac{\tilde{\mathbf{y}}_i}{\sqrt{\frac{1}{n} \sum_{j=1}^n \tilde{\mathbf{y}}_j^2}} \mid i \in [1, n]\}\end{aligned}\tag{3.6}$$

$\tilde{\mathbf{x}}_j^2$  et  $\tilde{\mathbf{y}}_j^2$  sont les images de  $\tilde{\mathbf{x}}_j$  et  $\tilde{\mathbf{y}}_j$  par la fonction qui élève au carré coordonnée par coordonnée.

La transformation des données en un ensemble de moyenne nulle et de variance 1 est utile aussi bien pour les réseaux de neurones que les processus gaussiens. Pour les réseaux de neurones, cette transformation aide à la convergence de l'algorithme de rétropropagation du gradient (LECUN et al. 2012; IOFFE et SZEGEDY 2015). Pour les processus gaussiens, elle prévient les instabilités numériques dans le calcul de la matrice de covariance et de son inverse. Les performances de cette standardisation ne sont pas présentées dans la partie suivante et celle-ci est utilisée à chaque fois.

### 3.5.4 Performance des transformations des mesures

Pour déterminer les transformations qui facilitent le plus l'apprentissage, j'utilise les données des perturbations collectées dans trois tuyaux de diamètre 40cm, 50cm et 65cm pour apprendre des processus gaussiens et des réseaux de neurones sur ces données.

Pour chacun des trois tuyaux, les mesures sont séparées aléatoirement entre un ensemble d'entraînement (90% des points) et un ensemble de tests (10% des points). Le processus gaussien est conditionné par l'ensemble des mesures de l'ensemble d'entraînement - et le réseau de neurones apprend sur celles-ci - puis le modèle est testé sur l'ensemble de test pour obtenir l'erreur moyenne de prédiction. Ces trois étapes sont répliquées 30 fois avec des partitions différentes des mesures entre l'ensemble d'entraînement et l'ensemble de test.

Les figures 3.12, 3.10 présentent la distribution de l'erreur que commettent les processus gaussiens et les réseaux de neurones pour la prédiction des perturbations selon les transformations des données réalisées sur les mesures préalablement à l'apprentissage. L'utilisation de l'hypothèse de symétrie correspond à l'ajout à l'ensemble des mesures, des mesures obtenues par symétrie selon l'axe vertical des mesures existantes.

Sur la figure 3.10, on voit que les réseaux de neurones présentent par contre une distribution d'erreur de moyenne plus faible et moins étendue avec la représentation cartésienne, et dans une moindre mesure la représentation hybride, par rapport à la représentation polaire. L'intégration des données symétriques donne des distributions d'erreur plus resserrées pour la quasi-totalité des cas ainsi que des erreurs moyennes comparables, voire plus basses dans certains cas.

La figure 3.11 présente la distribution de l'erreur pour des réseaux de neurones dans deux situations : dans la première le réseau de neurones est entraîné pour ne prédire que les perturbations

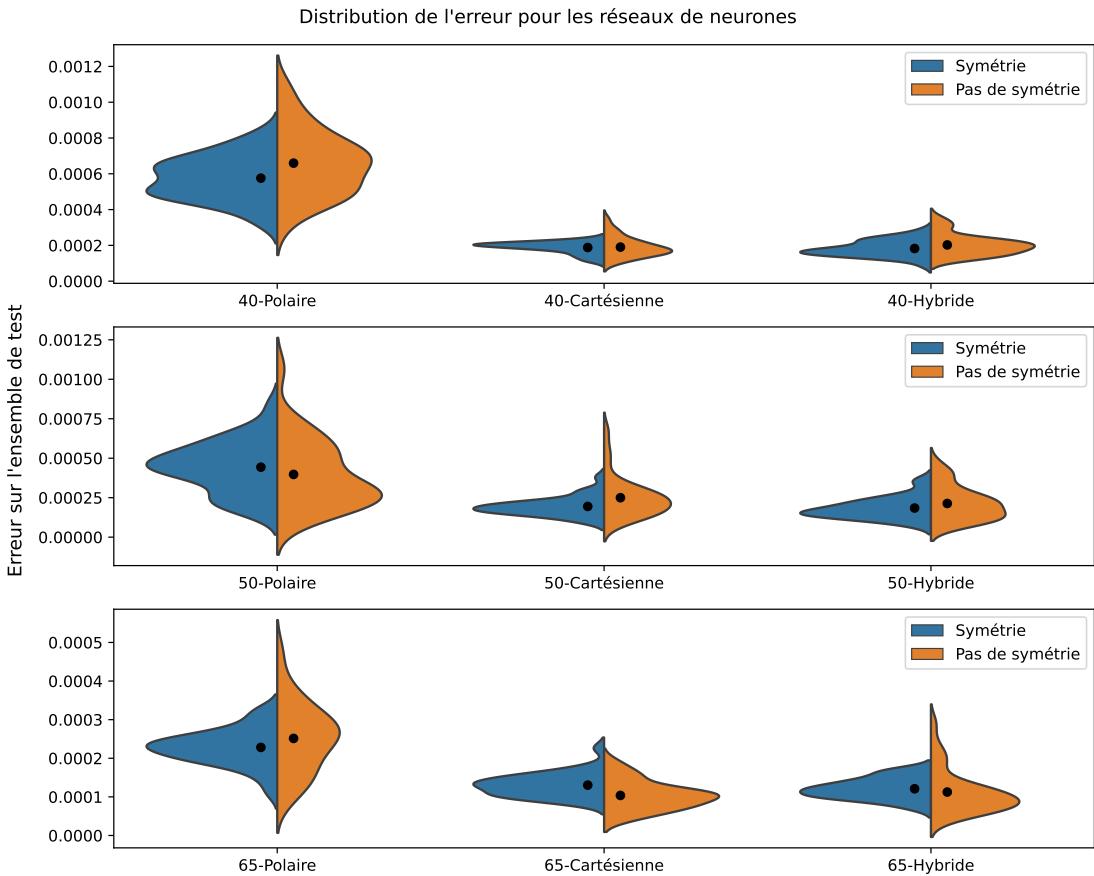


FIGURE 3.10 – Comparaison de la distribution de l'erreur des réseaux de neurones pour les 3 différentes représentations ainsi que l'utilisation de l'hypothèse de symétrie. Les données utilisées sont les mesures de perturbations dans 3 tuyaux de diamètre 40, 50 et 65cm. Ces données sont partitionnées aléatoirement en un ensemble de test (10% des points) et un ensemble d'entraînement (90%). Un réseau de neurones est ensuite entraîné les données de l'ensemble d'entraînement et on évalue son erreur sur les données de l'ensemble de test, dont le modèle n'a pas connaissance avant prédiction. On répète 30 fois ces étapes. La distribution de ces erreurs est présentée de manière verticale, alignée sur l'axe des ordonnées qui représente l'erreur moyenne sur l'ensemble de test. Le cercle noir au cœur de la distribution signale la valeur moyenne de celle-ci. Pour une distribution et un diamètre de tuyau donnés, les distributions correspondant à l'erreur du modèle avec et sans enrichissement des données sont placées l'une contre l'autre afin de rendre la comparaison plus aisée.

en force sur les axes x et z  $f_x$  et  $f_z$ , dans la seconde le réseau de neurones est entraîné pour prédire toutes les perturbations (force et couples selon x,y,z). Vu les résultats observés, les données utilisées le sont avec une représentation cartésienne et sont enrichies à l'aide de l'hypothèse de symétrie. On peut observer que le fait de prédire les perturbations sur les 6 axes n'aide pas particulièrement le réseau de neurones dans sa généralisation. Les erreurs sont distribuées de manière similaire, avec une moyenne légèrement plus faible quand la prédiction des perturbations est partielle. Ceci peut s'expliquer par le fait que l'entraînement des réseaux de neurones a été réalisé pour minimiser l'erreur de prédiction sur tous les axes de perturbation et est ensuite évalué seulement sur une partie d'entre eux.

Sur la figure 3.12 on peut observer que l'influence des différentes représentations de la position est bien moins marquée pour les processus gaussiens que pour les réseaux de neurones, à l'exception des données du tuyau de 50cm pour lesquelles la représentation cartésienne obtient une erreur plus basse et moins variable que les deux autres représentations.

En ce qui concerne l'hypothèse de symétrie, les processus gaussiens présentent une erreur plus faible quand l'hypothèse de symétrie n'est pas utilisée. Sur les données du tuyau de 65cm, la distribution des erreurs est bien moins large quand l'hypothèse de symétrie est utilisée.

Il n'est pas anormal que l'utilisation de cette hypothèse de symétrie mène à une erreur plus importante car les données collectées ne sont pas parfaitement symétriques. En effet, l'introduction de ces points symétrique mène régulièrement à l'existence de localisations dans le tuyau pour lesquelles il y a deux mesures de perturbations différentes, rendant impossible la réalisation d'une prédiction parfaite. Ceci est illustré par la figure 3.14 où l'on voit clairement ces points à double mesure et le compromis choisi par le modèle.

Les niveaux d'erreur des processus gaussiens sur les données de l'ensemble d'apprentissage avec la symétrie, c'est-à-dire sur des données connues du modèle au moment de la prédiction, sont de l'ordre de  $5 \cdot 10^{-5}$ . Cette valeur est du même ordre que les écarts entre les moyennes sur les distributions pour processus gaussiens (figure 3.12). Ainsi, les processus gaussiens présentent sur l'ensemble d'apprentissage une erreur de même ordre de grandeur que celle présente directement dans les données et l'utilisation de l'hypothèse de symétrie ne perturbe pas l'apprentissage même si elle augmente l'erreur moyenne.

La figure 3.13 présente la distribution de l'erreur pour les réseaux de neurones et les processus gaussiens. Les deux modèles ont été entraînés sur des données enrichies avec l'hypothèse de symétrie, qui améliore les performances pour les réseaux de neurones et est fortement motivée par les symétries de l'environnement. Comme la représentation cartésienne ne dégrade pas les performances d'apprentissage pour les processus gaussiens et qu'elle les améliore pour les réseaux de neurones, c'est cette représentation qui a été utilisée pour comparer les deux classes de modèles.

Sur chacun des trois tuyaux les processus gaussiens obtiennent une meilleure erreur moyenne et une distribution plus resserrée. Les processus gaussiens obtiennent donc des erreurs plus faibles et moins variables que les réseaux de neurones.

À partir des observations faites sur les figures, 3.10, 3.11, 3.12, 3.13 j'ai décidé d'utiliser dans la suite de cette thèse des modèles des perturbations qui sont des processus gaussiens conditionnés par les mesures réalisées et leur symétrie. La position est de ces mesures est représentée sous forme cartésienne.

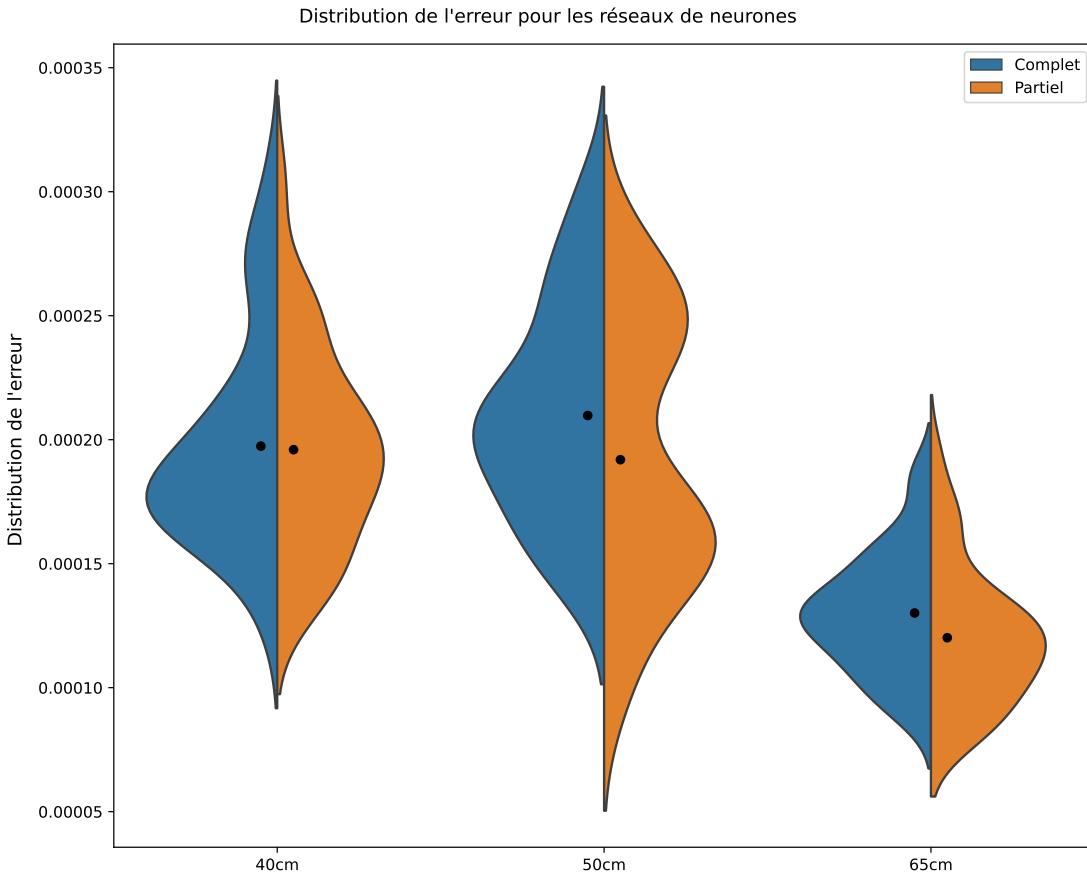


FIGURE 3.11 – Comparaison de la distribution de l'erreur des réseaux de neurones pour une prédiction partielle ( $f_x, f_z$  uniquement) et complète ( $f_x, f_y, f_z, \tau_x, \tau_y, \tau_z$ ) des perturbations. Les erreurs présentées sont les erreurs sur la prédiction des perturbations partielles uniquement. Les données utilisées sont les mesures de perturbations dans 3 tuyaux de diamètre 40, 50 et 65cm. Ces données sont partitionnées aléatoirement en un ensemble de test (10% des points) et un ensemble d'entraînement (90%). Un réseau de neurones est ensuite entraîné les données de l'ensemble d'entraînement et on évalue son erreur sur les données de l'ensemble de test, dont le modèle n'a pas connaissance avant prédiction. On répète 30 fois ces étapes. La distribution de ces erreurs est présentée de manière verticale, alignée sur l'axe des ordonnées qui représente l'erreur moyenne sur l'ensemble de test. Le cercle noir au cœur de la distribution signale la valeur moyenne de celle-ci. Pour une distribution et un diamètre de tuyau donnés, les distributions correspondant à l'erreur du modèle pour une prédiction complète et partielle sont placées l'une contre l'autre afin de rendre la comparaison plus aisée.

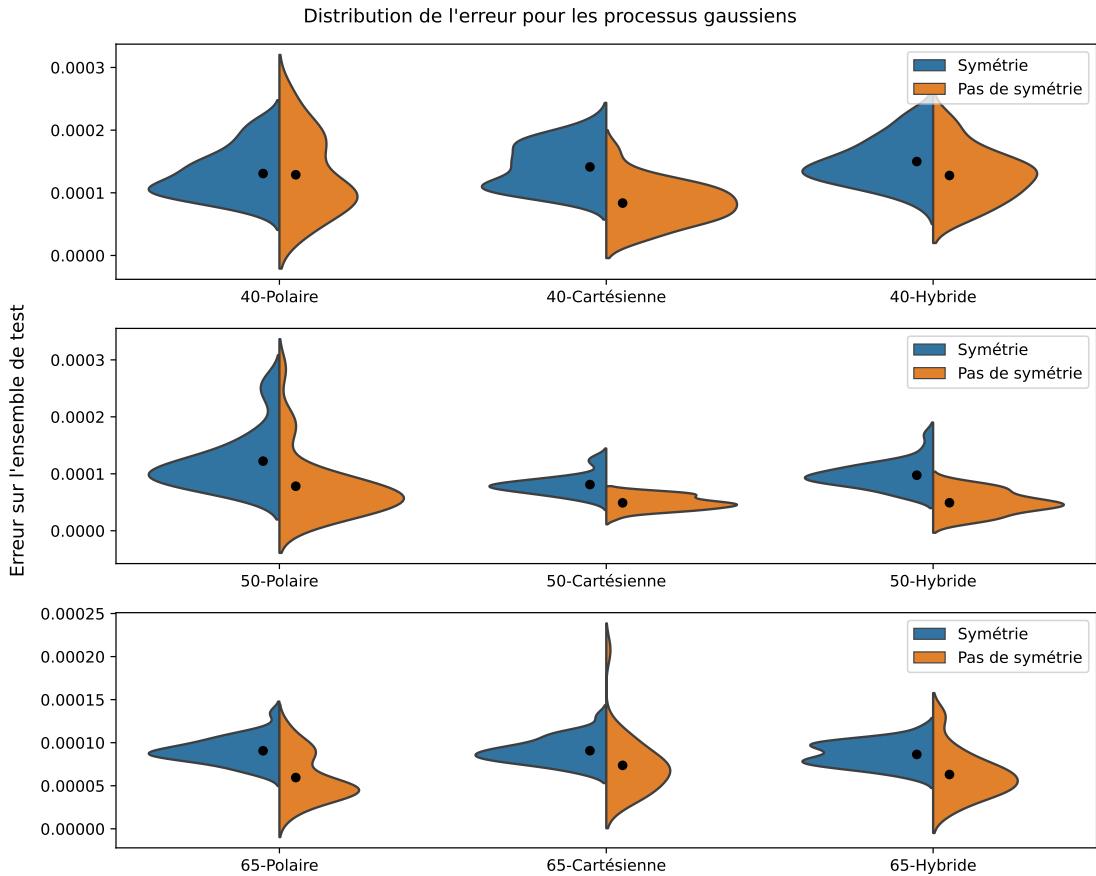


FIGURE 3.12 – Comparaison de la distribution de l'erreur des processus gaussiens pour les 3 différentes représentations de la position et l'utilisation de l'hypothèse de symétrie. Les données utilisées sont les mesures de perturbations dans 3 tuyaux de diamètre 40, 50 et 65cm. Ces données sont partitionnées aléatoirement en un ensemble de test (10% des points) et un ensemble d'entraînement (90%). Un processus gaussien est ensuite conditionné par les données de l'ensemble d'entraînement et on évalue son erreur sur les données de l'ensemble de test, dont le modèle n'a pas connaissance avant prédiction. On répète 30 fois ces étapes. La distribution de ces erreurs est présentée de manière verticale, alignée sur l'axe des ordonnées qui représente l'erreur moyenne sur l'ensemble de test. Le cercle noir au cœur de la distribution signale la valeur moyenne de celle-ci. Pour une distribution et un diamètre de tuyau donnés, les distributions correspondant à l'erreur du modèle avec et sans enrichissement des données sont placées l'une contre l'autre afin de rendre la comparaison plus aisée.

Comparaison de la distribution de l'erreur pour les réseaux de neurones et les processus gaussiens

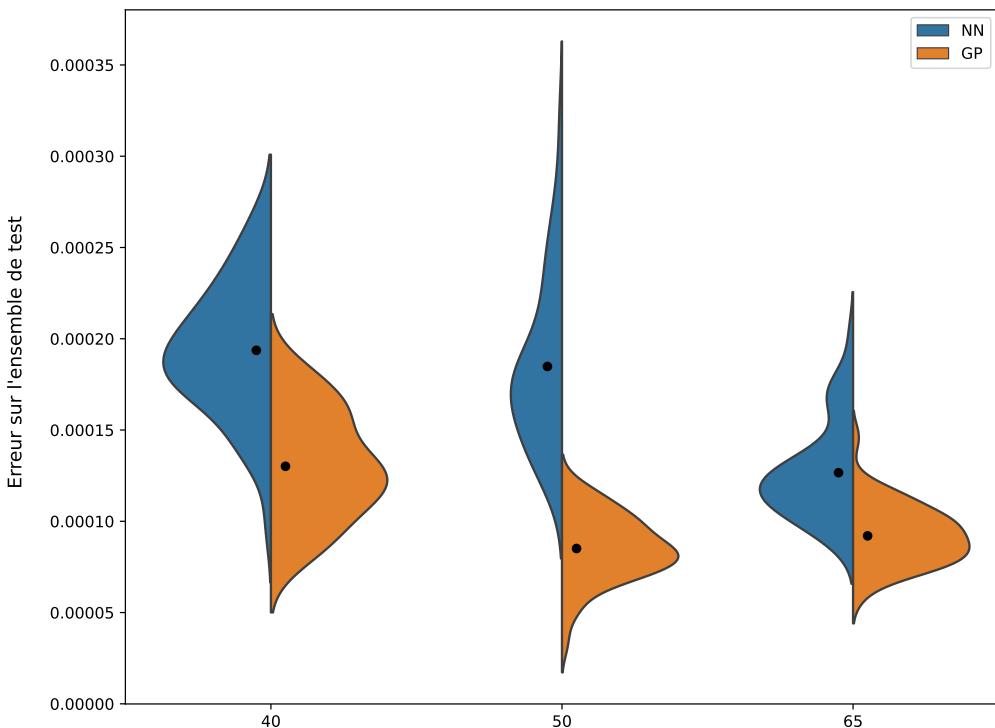


FIGURE 3.13 – Comparaison de la distribution de l'erreur pour des processus gaussiens et des réseaux de neurones. Les données utilisées sont les mesures de perturbations dans 3 tuyaux de diamètre 40, 50 et 65cm. Ces données sont partitionnées aléatoirement en un ensemble de test (10% des points) et un ensemble d'entraînement (90%). Un processus gaussien est ensuite conditionné par les données de l'ensemble d'entraînement et on évalue son erreur sur les données de l'ensemble de test, dont le modèle n'a pas connaissance avant prédiction. Un réseau de neurones est ensuite entraîné les données de l'ensemble d'entraînement et on évalue son erreur sur les données de l'ensemble de test, dont le modèle n'a pas connaissance avant prédiction. On répète 50 fois ces étapes. La distribution de ces erreurs est présentée de manière verticale, alignée sur l'axe des ordonnées qui représente l'erreur moyenne sur l'ensemble de test. Le cercle noir au cœur de la distribution signale la valeur moyenne de celle-ci.

### 3.5.5 Analyse des perturbations prédictes par le modèle appris

Une fois la classe de modèles et la représentation fixées, on peut s'intéresser au comportement du modèle appris.

La figure 3.14 présente les mesures enrichies via l'hypothèse de symétrie ainsi que le compromis trouvé par le processus gaussien pour la prédiction quand la symétrie n'est pas exactement respectée.

Ces prédictions sont faites à des points pour lequel le modèle a connaissance des mesures. On peut observer qu'en cas d'absence de symétrie, le modèle fait une prédiction à mi chemin entre les deux mesures connues.

Une fois le modèle appris sur les données collectées, il est possible de réaliser des prédictions à des positions pour lesquelles il n'y a pas de mesures et de couvrir ainsi le tuyau plus finement. La figure 3.15 présente ces prédictions à des positions du tuyau formant une grille de 30 points de côté. Ces nouvelles prédictions dessinent plus précisément et extrapolent les schémas observés sur la figure 3.6. En particulier, pour les trois tuyaux on observe une zone, en bas au centre, dans laquelle les perturbations sont nulles. Les perturbations autour de cette zone sont dirigées vers celles-ci, ce qui fait de cette zone une zone stable. Si le robot s'en écarte il y sera ramené.

D'autres zones de perturbations quasi nulles se dessinent aussi à l'interface entre le centre et les zones d'attraction par les parois des quarts supérieurs gauche et droit. Mais ces zones peu perturbées ne sont pas des zones stables.

L'observation de ces schémas de perturbations donne une première piste pour la conception d'un contrôleur permettant un vol stable dans le tuyau. Si on s'en tient aux mesures statiques représentées sur ces schémas, faire voler le robot dans la zone stable au centre en bas pourrait être une approche de prévention des problèmes rencontrés durant un vol dans un tuyau. La section suivante présente une approche pour intégrer ce modèle des perturbations dans le contrôle afin d'utiliser les informations sur ces perturbations pour stabiliser le vol d'un quadrotor dans un tuyau.

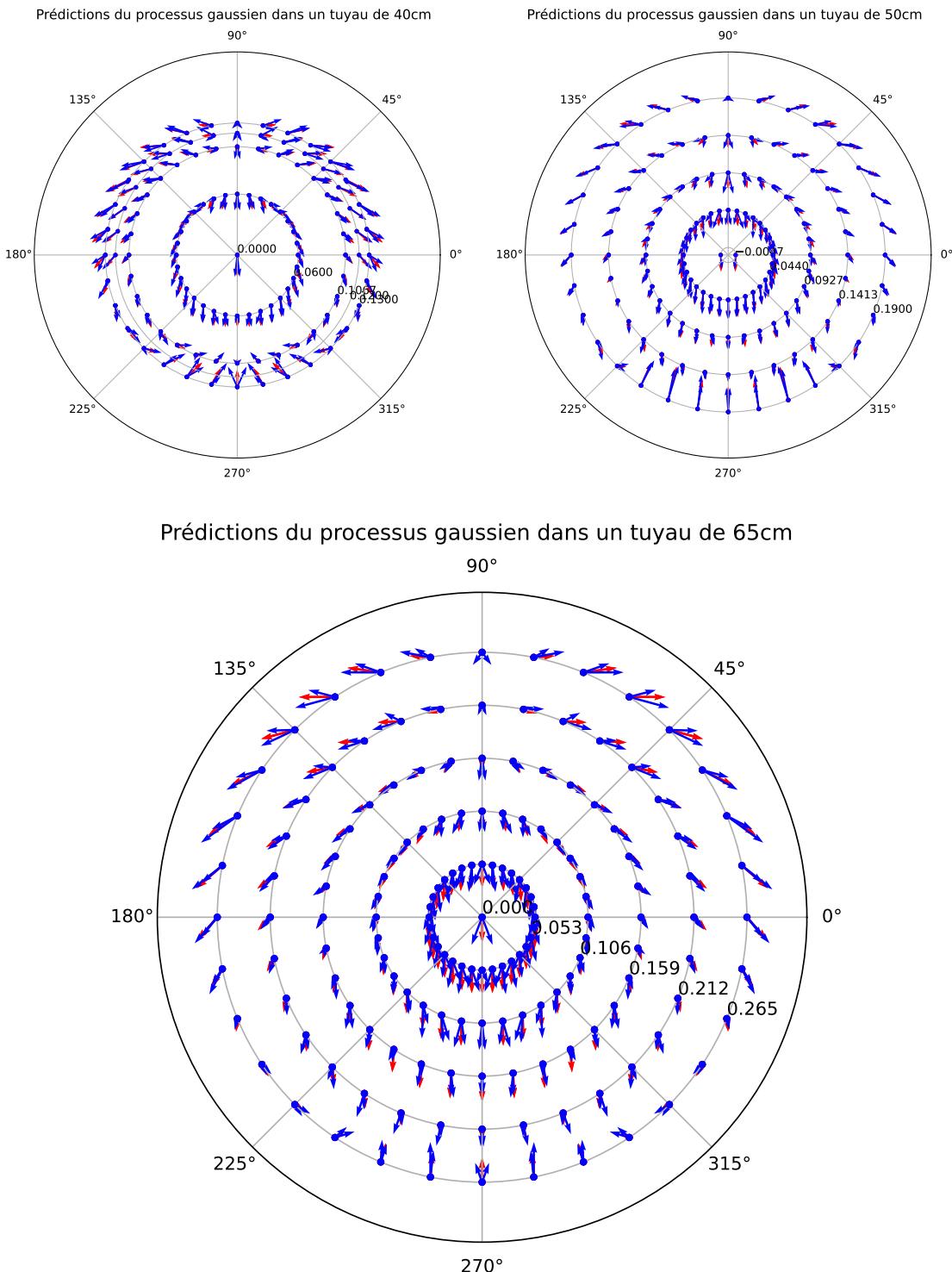


FIGURE 3.14 – Comparaison des prédictions d'un processus gaussien (en rouge) avec les mesures (en bleu) réalisées dans trois tuyaux de diamètre 40cm (haut gauche), 50cm (haut droite) et 65cm (bas) et enrichies à l'aide de l'hypothèse de symétrie : pour chaque mesure collectée, la mesure symétrique est ajoutée au jeu de données. C'est cet ajout qui conduit à la présence de deux vecteurs bleus en chaque point. Un de ces vecteurs correspond directement à une mesure réelle et le second au symétrique d'une mesure réelle collecté en un point symétrique.

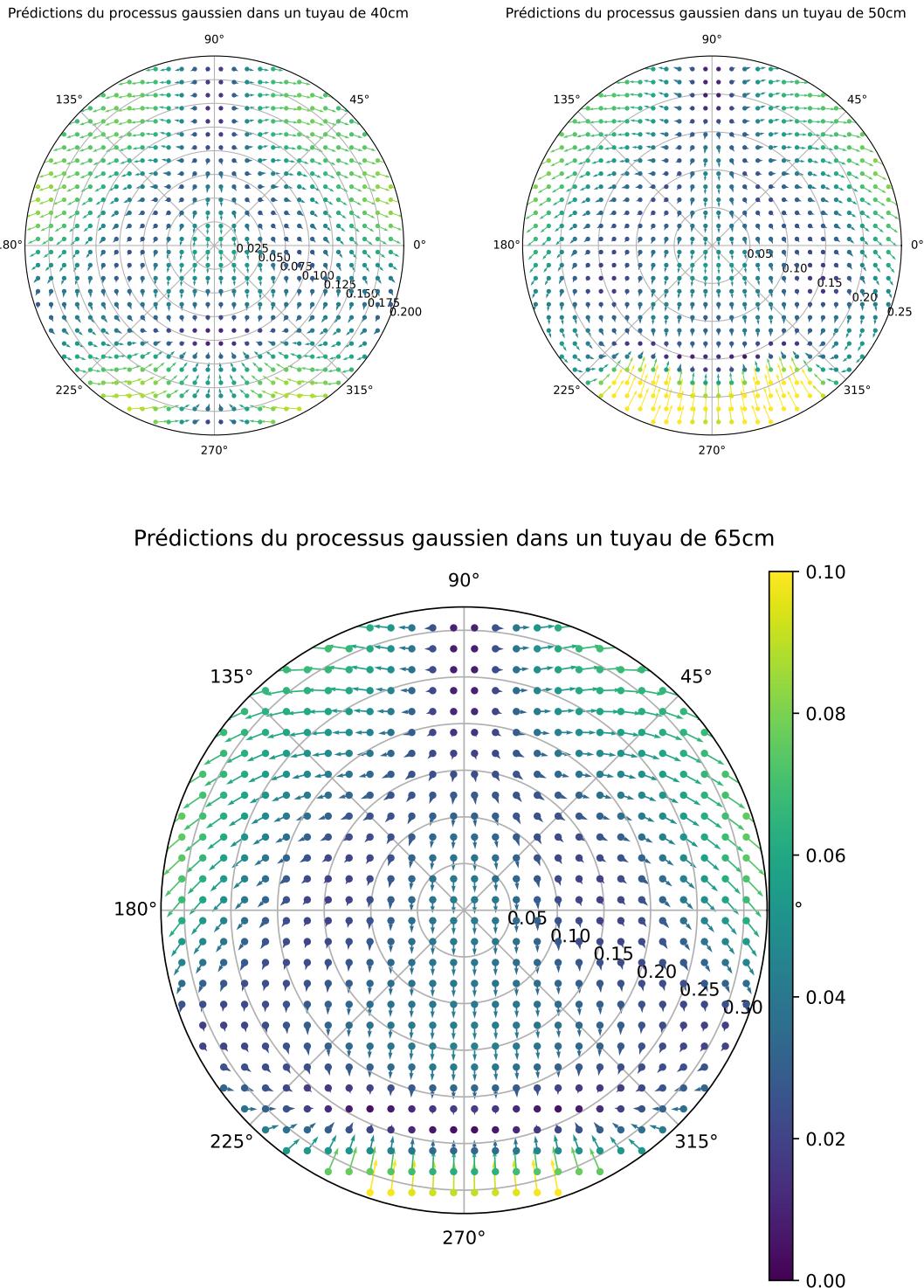


FIGURE 3.15 – Prédiction d'un processus gaussien entre les points de mesures pour des tuyaux de 40cm (haut gauche), 50cm (haut droite) et 65cm (bas) de diamètre. Les flèches représentent les perturbations ( $f_x, f_z$ ) prédictes par le processus gaussien au point d'origine de la flèche. Les couleurs des flèches sont déterminées par la norme de la perturbation selon l'échelle de la figure du bas, exprimée en N. Les points de prédiction correspondent à une grille de 30 points de côté couvrant le tuyau et dont on a exclu les points qui étaient extérieurs au tuyau.

### 3.6 Intégration du modèle des perturbations dans les contrôleurs basés contrôle optimal

L'intégration du modèle de l'environnement (ici les perturbations dans le tuyau) se fait en ajoutant les perturbations prédictes par le modèle aux équations 2.6. Ceci couple ainsi la position du robot aux forces qui s'exercent sur celui-ci et donne les équations suivantes :

$$\begin{aligned} [\dot{\mathbf{v}}_W]_{C_W} &= \frac{1}{m} \left( \sum_{i=1}^{n_p} f_i [R_B^W]_{C_W} [\mathbf{t}_i]_{C_B} + [\text{pert}_f(\mathbf{p}_w)]_{C_W} \right) - g[\mathbf{z}_w]_{C_W} \\ [\dot{\boldsymbol{\omega}}_{WB}]_{C_B} &= [J]_{C_B}^{-1} \left( \left( \sum_{i=1}^{n_p} f_i ([\mathbf{p}_{a_i}]_{C_B} \wedge [\mathbf{t}_i]_{C_B} + c_i [\mathbf{t}_i]_{C_B}) \right) + [\text{pert}_\tau(\mathbf{p}_w)]_{C_B} - [[\boldsymbol{\omega}_{WB}]_{C_B}] \wedge [J]_{C_B} [\boldsymbol{\omega}_{WB}]_{C_B} \right) \end{aligned} \quad (3.7)$$

où  $\text{pert}_f(\mathbf{p}_w)$  et  $\text{pert}_\tau(\mathbf{p}_w)$  sont respectivement les forces et les couples de perturbations prédictes par le modèle à la position du robot  $\mathbf{p}_W$ . Les équations de (2.6) qui ne sont pas reproduites sont laissées identiques. Nous nous sommes limités ici à un modèle de prédiction des forces causées par ces perturbations et nous avons ainsi fixé  $\text{pert}_\tau(\mathbf{p}_w) = 0$ .

Pour illustrer l'intérêt de l'intégration de ces perturbations au modèle utilisé pour la résolution du problème de contrôle optimal, je présente des trajectoires dans un environnement intégrant le modèle de perturbations appris sur les mesures réalisées dans le tuyau. Pour évaluer la commande dans les zones où les perturbations sont importantes, une trajectoire passant proche des parois d'un tuyau de 65cm de diamètre sera utilisée. Je ne présente pas ici d'évaluation sur une tâche de stabilisation puisque celle-ci n'illustre pas aussi bien l'intérêt du modèle de perturbations en ne faisant pas voler le quadrotor dans les zones les plus perturbées. La figure 3.15 montre qu'autour de l'axe du tuyau, les perturbations prédictes sont faibles, et de direction uniforme (vers le bas). Une tâche de stabilisation par le MPC avec et sans connaissance des perturbations est présenté dans la section 4 sur la figure 4.6.

J'utilise comme référence une trajectoire similaire à celle décrite par les équations 2.25 avec une amplitude verticale et horizontale de 25cm. Cette trajectoire diffère de la précédente par un décalage de phase entre les oscillations sur l'axe  $x$  et  $z$ , ainsi qu'une augmentation linéaire de l'amplitude de ces oscillations pour atteindre 25cm à 2s de trajectoire. C'est une trajectoire en hélice qui tourne autour de l'axe du tuyau et s'approche petit à petit des parois.

Ces changements sont effectués afin de proposer une trajectoire proche des parois pendant une longue durée. La distance aux parois reste constante et égale à 7.5cm à partir de 2s et pousse le robot à voler dans les zones les plus perturbées. Contrairement aux simulations hors tuyau, aucune perturbation aléatoire telles que décrite dans la section 2.4.3.3 et inconnue des contrôleurs n'est appliquée. Des simulations avec perturbations ont été réalisées, mais aucun des contrôleurs utilisés ne parvenait à prévenir les collisions avec les parois du tunnel en présence de telles perturbations et avec une telle proximité à ces parois.

La figure 3.16 présente deux trajectoires du contrôleur *MPC*, l'une avec connaissance du modèle de perturbations dues à la présence dans le tuyau et l'autre sans. Le contrôleur ayant connaissance des perturbations parvient suivre la trajectoire proche des parois jusqu'au bout (5s) avec un écart à celle-ci faible ( $<1\text{cm}$ ). Le contrôleur qui n'a pas connaissance des perturbations est incapable de suivre cette même trajectoire et fait entrer en collision le robot et la paroi du tuyau à 3.99s. Avant

même cette collision, l'écart à la trajectoire de référence est plus important qu'avec la connaissance de l'environnement comme l'illustre la valeur du coût de la trajectoire.

Le contrôleur proportionnel décrit dans la section 2.4.3.2 peut être modifié pour intégrer les perturbations à la position actuelle en addition de la compensation à la gravité. La figure 3.17 présente deux trajectoires comparant un contrôleur proportionnel avec la prise en compte des perturbations et un sans.

Que ce soit avec les informations sur les perturbations ou sans, le contrôleur proportionnel ne parvient pas à suivre la trajectoire à une telle proximité des parois du tuyau. Le contrôleur avec intégration du modèle de l'environnement suit une trajectoire de forme similaire à celle de la référence, celui-ci prend un peu trop de vitesse sur l'axe  $x$  et entre en collision avec la paroi au bout de presque 3s. Sans le modèle de l'environnement, le contrôleur n'est pas capable de compenser les perturbations et suit une trajectoire qui ne ressemble que très peu à la trajectoire de référence et entre en collision avec la paroi après un peu moins de 2s.

## 3.7 Conclusion

Le protocole et le dispositif détaillés dans cette section permettent de collecter des mesures concernant les perturbations statiques en force et en couple observées en présence d'un Crazyflie dans un tuyau. L'analyse de ces perturbations révèle des schémas attendus (effet de sol) et d'autres moins (aspiration par les parois, zone calme).

L'apprentissage supervisé d'un modèle de ces perturbations à l'aide d'un processus gaussien permet de réaliser des prédictions de ces perturbations en tout point de l'environnement.

En intégrant les perturbations prédites au contrôleur *MPC* et au contrôleur proportionnel, ceux-ci peuvent naviguer dans le tuyau de manière bien plus stable. Le contrôleur *MPC* présente bien une meilleure capacité à suivre une trajectoire et à prendre en compte les perturbations modélisées. Ceci lui permettant de suivre des trajectoires très près des parois du tuyau.

Le modèle des perturbations qu'utilisent le contrôleur *MPC* et le contrôleur proportionnel est le modèle correct de ces perturbations : c'est selon ce modèle que les perturbations sont appliquées dans la simulation. Bien entendu ceci n'est pas une hypothèse raisonnable à faire pour un contrôleur qui va voler sur un véritable robot puisque le modèle appris des perturbations ne sera évidemment pas exact même s'il est appris sur des données réelles. Les approximations faites pour la modélisation (caractère statique, influence du dispositif de mesure sur les mesures, erreur d'apprentissage non nulle) ont pour conséquence que les contrôleurs présentés ici auraient une performance moins bonne avec un modèle inexact des perturbations réelles.

Il serait donc utile d'évaluer la robustesse de ces contrôleurs aux différences entre le modèle des perturbations et les perturbations réelles. Le contrôleur n'ayant pas connaissance des perturbations est un cas particulier où un modèle très inexact des perturbations est utilisé. Avec ce modèle très inexact, aucun des deux contrôleurs n'est capable de suivre la trajectoire à proximité des parois.

Le temps nécessaire pour calculer les commandes du contrôleur *MPC* est assez important (500 fois plus lent que le temps réel comme précisé par la figure 4.5). L'implémentation que j'utilise pour les résultats présentés ne permet pas de calculer ces commandes en temps réel et encore moins de les embarquer à bord du robot.

Le chapitre 4 présente une approche pour obtenir un contrôle tournant en temps réel à bord du Crazyflie approchant les commandes du contrôleur *MPC* en utilisant l'apprentissage par imitation.

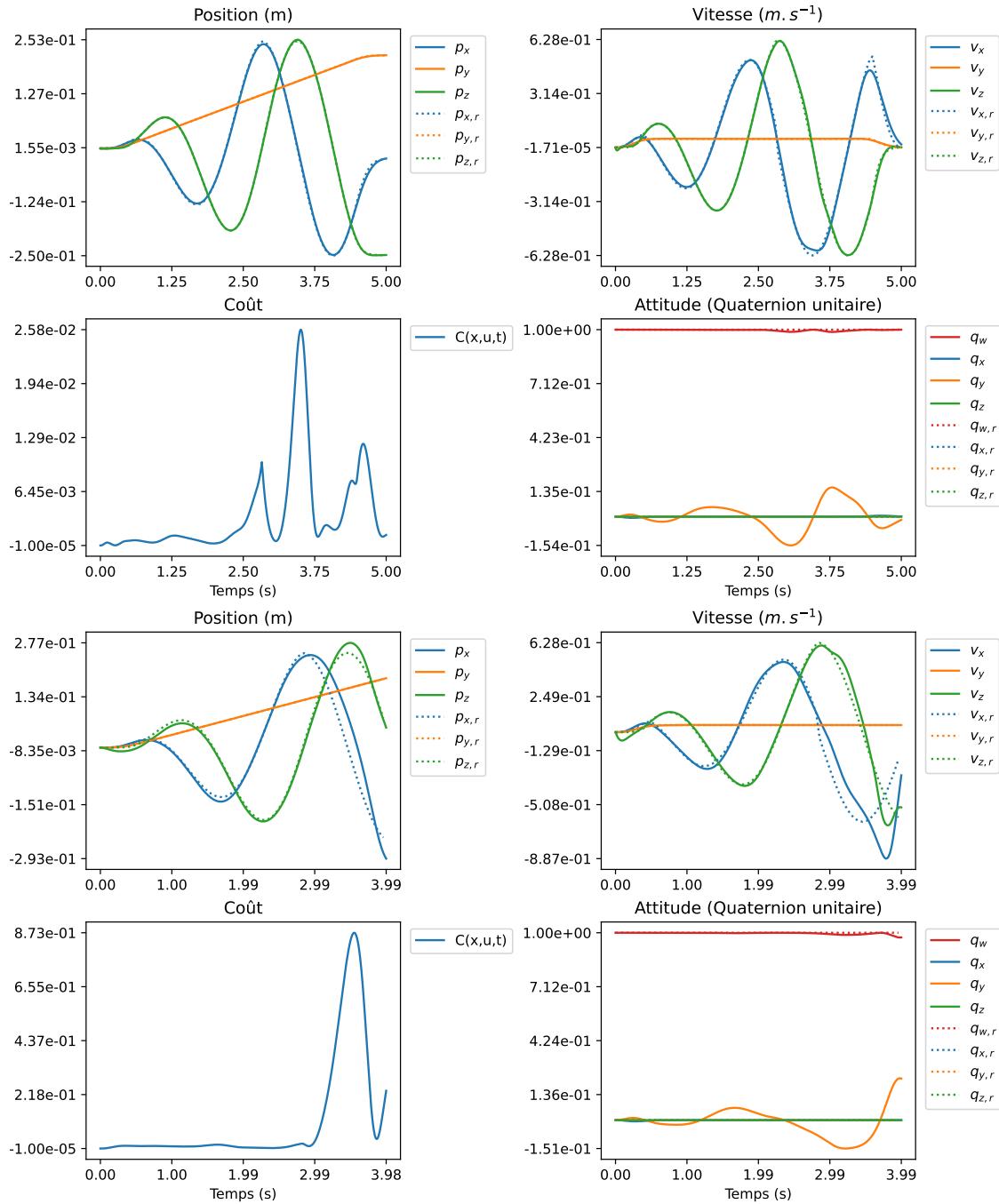


FIGURE 3.16 – Comparaison de trajectoires dans un tuyau de 65cm de diamètre pour un contrôleur MPC avec connaissance des perturbations dues au tuyau (en haut) et sans connaissance de ces perturbations (en bas). L'état de référence est indiqué par un trait pointillé et l'indice  $r$ . Quand cet état de référence n'est pas visible c'est qu'il est couvert par la courbe de la trajectoire et que ceux-ci coïncident. La trajectoire sans connaissance est stoppée prématurément suite à la collision entre le robot et la paroi.

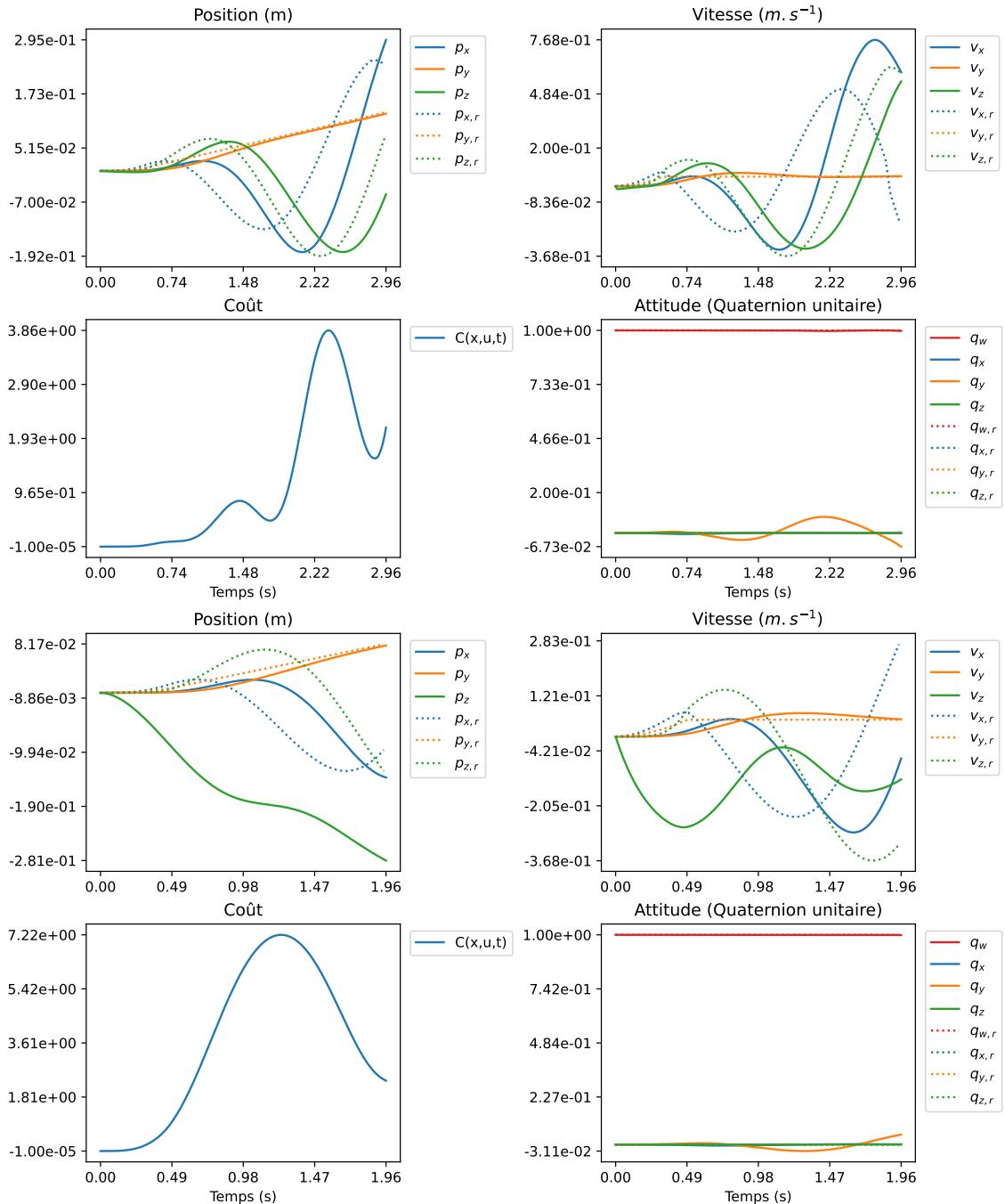


FIGURE 3.17 – Comparaison de trajectoires dans un tuyau de 65cm de diamètre pour un contrôleur proportionnel avec compensation des perturbations dues au tuyau (en haut) et sans connaissance de ces perturbations (en bas). L'état de référence est indiqué par un trait pointillé et l'indice  $r$ . Quand cet état de référence n'est pas visible c'est qu'il est couvert par la courbe de la trajectoire et que ceux-ci coïncident. Les deux trajectoires sont stoppées prématurément suite à une collision entre le robot et la paroi du tuyau.

## Chapitre 4

# Apprentissage par imitation du contrôle optimal

### 4.1 Introduction

Le contrôleur MPC basé contrôle optimal présenté dans le chapitre 2.4.2 permet un contrôle précis d'un robot de type quadrotor et parvient à intégrer des informations d'un environnement comme le tuyau présenté dans le chapitre 3.6 pour naviguer dans un environnement perturbé. Cependant, le coût en calcul de ce contrôleur est bien trop important pour être embarqué directement à bord du robot.

L'implémentation de ce contrôleur MPC en python qui a été utilisée pour produire les données présentées dans les sections 2.4.3 et 3.6 est calculée sur une machine fixe (bien plus rapide que le matériel qu'embarque le Crazyflie) 500 fois plus lentement que le temps réel. Je ne me suis pas investi beaucoup dans l'optimisation cette implémentation. Le contrôle optimal calculé à un instant  $t$  n'est en particulier pas repris comme valeur de départ pour la recherche à l'instant suivant et pourrait grandement accélérer les calculs du contrôleur MPC. Mais ces optimisations de cette implémentation pourraient ne pas suffire.

Une alternative pour utiliser un tel contrôleur peut-être de déporter les calculs sur une machine externe au robot possédant une puissance bien supérieure à celle embarquée. Cela demanderait tout de même d'importantes optimisations de l'implémentation actuelle. Par ailleurs, les contraintes de temps de communication entre la machine réalisant les calculs déportés posent problème. Les délais de communication par radio avec un robot comme le crazyflie ne sont pas négligeables et même trop importants pour pouvoir contrôler directement le régime moteur comme le fait le contrôleur présenté dans le chapitre 2.4.3.1.

Un contrôleur MPC déporté est réalisé et appliqué au Crazyflie dans CARLOS et al. 2020, mais celui-ci doit exercer un contrôle à un niveau plus élevé (consignes de position angulaire) et avec une fréquence plus faible (70Hz) ainsi que réaliser un système de compensation du délai dû aux communications. L'approche que j'ai choisie pour pouvoir exploiter un contrôleur type MPC est de chercher à en réduire drastiquement le coût en calculs.

La réduction de la durée de l'horizon ou de la fréquence dans le problème de contrôle optimal ne sont pas suffisantes pour réduire le coût en calcul du contrôleur MPC au point de le rendre embarquable et un horizon trop court ou une fréquence trop faible réduisent la qualité du contrôle.

L'approche que nous avons choisie pour réduire le coût de ce contrôleur est d'apprendre une

approximation de celui-ci. Le fait de pouvoir calculer la commande optimale dans un état donné permet la génération d'un ensemble de données qu'il est ensuite possible d'apprendre à imiter à l'aide des techniques d'apprentissage supervisé. Cette imitation permet d'obtenir un contrôleur qui approxime les commandes imitées pour un coût en calcul plus faible. Les travaux de MOLCHANOV et al. 2019 en particulier embarquent un contrôleur appris à bord d'un Crazyflie et celui-ci calcule des commandes en temps réel à 500Hz (0.8ms pour l'évaluation du contrôleur lui-même) et nous avons pu vérifier des temps de calcul similaires lors de nos expérimentations.

La recherche d'un contrôle par une approche d'apprentissage par renforcement est aussi possible, mais ces approches ne font pas aussi directement usage de la capacité à générer de bons contrôles qui est disponible grâce au contrôleur MPC présenté dans la section 2.4.3.1 et auquel on peut intégrer le modèle des perturbations comme présenté dans la section 3.6.

#### 4.1.1 Objectifs

Ce chapitre a pour objectif d'obtenir un contrôleur dont le coût en calcul permette de l'embarquer et de l'exécuter en temps réel à bord d'un Crazyflie. À cet effet, nous voulons déterminer un algorithme qui tire parti de la capacité à calculer une commande optimale hors ligne pour procéder à l'apprentissage supervisé d'un réseau de neurones approximant la commande optimale.

#### Plan

J'aborde d'abord dans ce chapitre une description de plusieurs travaux qui traitent de l'apprentissage de contrôleur pour les multirotors, puis je formule le problème d'apprentissage par imitation et propose une modification de l'algorithme DAGGER pour conduire l'apprentissage. Je détaille ensuite nos choix concernant l'architecture du contrôleur appris et les résultats obtenus en simulation avec un tel contrôleur appris. J'aborde ensuite les modifications nécessaires pour y intégrer un modèle de l'environnement ainsi que pour gérer la tâche de suivi de trajectoire.

### 4.2 Travaux connexes

L'apprentissage de contrôleur pour des quadrotors est un sujet souvent abordé dans la littérature.

#### Apprentissage par renforcement

L'apprentissage par renforcement en particulier permet la recherche de contrôleurs en ne spécifiant qu'un modèle et un coût à optimiser, à l'instar du contrôle optimal cette approche est coûteuse en calculs. WASLANDER et al. 2005 apprennent ainsi un contrôleur linéaire pour le contrôle de l'altitude d'un quadrotor. L'optimisation des paramètres de ces contrôleurs linéaires est réalisé par sélection de paramètres tirés aléatoirement et selon leur réussite sur un modèle constitué de processus gaussiens. KOCH et al. 2019 comparent trois algorithmes d'apprentissage par renforcement (DDPG, PPO et TRPO) pour l'apprentissage d'un réseau de neurones contrôleur de la vitesse angulaire d'un quadrotor sur un simulateur dit *jumeau numérique*. MOLCHANOV et al. 2019 présentent l'apprentissage d'un réseau de neurone (PPO) pour le contrôle d'un quadrotor. Le contrôleur est appris à l'aide d'une approche dite *domain randomization* qui consiste à entraîner un contrôleur capable de faire voler plusieurs robots dont les caractéristiques d'architecture (poids, puissance des moteurs, etc) différent. Ceci permet la recherche d'un contrôleur qui soit

robuste à la diversité des dynamiques et donc capable de fonctionner sur un robot malgré des différences entre la dynamique de celui-ci et son modèle dans la simulation. Le contrôleur appris est suffisamment robuste pour être embarqué à bord de plusieurs quadrotors avec des architectures différentes et parvient à stabiliser ces robots malgré des états initiaux très éloignés de l'état dans lequel le robot finit par se stabiliser. HWANGBO et al. 2017 décrivent un algorithme d'apprentissage par renforcement et une méthode d'exploration adaptée aux systèmes instables comme les quadrotors. Ils apprennent avec cet algorithme un réseau de neurones qui permet de stabiliser un quadrotor. L'apprentissage par renforcement nécessite une quantité de calculs et d'exploration assez importante.

## Apprentissage par imitation

La capacité à générer des trajectoires ou commandes de qualité satisfaisantes, même hors ligne et au prix de calculs importants, permet d'apprendre des contrôleurs non pas par renforcement, mais par imitation.

ABBEEL, COATES et NG 2010 détaillent ainsi une méthode pour inférer la trajectoire d'une manœuvre acrobatique pour un hélicoptère à partir de plusieurs démonstrations de cette manœuvre et malgré une éventuelle diversité des instances de cette manœuvre. Cette manœuvre est ensuite raffinée par une commande optimale hors ligne pour être ensuite exécutée par un contrôleur LQR en ligne. GANDHI, PINTO et GUPTA 2017 proposent une approche auto-supervisée : un quadrotor conçu pour éviter de recevoir des dommages lors d'un crash est utilisé pour collecter des données de trajectoires contenant ou non des crashes. Un réseau de neurones convolutionnel est ensuite entraîné de manière supervisée pour apprendre à prédire quelles sont les directions de vol qui ont pour conséquence un crash. KAUFMANN et al. 2019 entraînent un contrôleur à prédire la position de portes à partir d'un flux visuel pour un quadrotor de course. Le réseau de neurone convolutionnel est entraîné à partir de données collectées par un système de capture de mouvement qui permet d'associer à une image la position relative de la porte apparaissant sur l'image. Cette position est ensuite fournie à un contrôleur MPC qui peut ainsi planifier la trajectoire à travers la porte. S. LI et al. 2020 présentent deux approches sur un quadrotor dont la dynamique est restreinte à deux dimensions (x et z). Ces deux approches consistent en l'apprentissage d'un réseau de neurones par imitation de trajectoires. Dans la première approche, la trajectoire est générée par contrôle optimal. Dans la seconde elle est générée à l'aide de la propriété de *differential flatness* d'un quadrotor qui permet d'exprimer une trajectoire et une commande en fonction de la position, du lacet et de leurs dérivées. TOMIĆ, MAIER et HADDADIN 2014 proposent d'apprendre une primitive de mouvement dynamique (DMP) à partir de trajectoires calculées par contrôle optimal sur un quadrotor en 2 dimensions.

## Mécanisme de guidage de l'apprentissage par imitation

S. LEVINE et KOLTUN 2013 décrivent une approche (Guided Policy Search, GPS) pour l'imitation d'un contrôle optimal dans lequel le contrôle optimal est biaisé pour produire de commandes plus proches de celles que réaliserait le contrôleur appris pour un même état. Une procédure pour prévenir un décalage de distribution entre le contrôleur appris et la manière dont sont collectées les exemples à imiter est aussi proposée. KAHN et al. 2017 étendent l'approche GPS pour des systèmes instables comme les multirotors. Durant la procédure de collecte des données, plutôt que d'exécuter une commande du contrôleur appris, c'est un contrôleur MPC biaisé pour ressembler au contrôleur appris, comme pour GPS, qui est exécuté. L'utilisation d'un MPC, même biaisé,

permet d'éviter les crashes au multirotor tout en collectant des données proches de ce que rencontrerait le contrôleur appris. Kahn et al. détaillent aussi une méthode qui permet d'apprendre le contrôleur ayant pour entrée des observations éventuellement partielles de l'état alors que le contrôle optimal a accès à la totalité de l'état durant l'optimisation. MORA et al. 2021 proposent une approche similaire à GPS, mais plutôt que de biaiser le contrôle optimal pour l'inciter à fournir des commandes proches de ce que fournirait le contrôleur appris, les commandes générées par le contrôleur appris sont optimisées localement, à l'aide d'une méthode d'ordre 2, puis imitées par le contrôleur appris. La recherche de commande optimale est ainsi réalisée dans l'espace des trajectoires puis les paramètres du contrôleur sont mis à jour, par apprentissage supervisé, pour refléter cette recherche dans l'espace des trajectoires.

LIN et al. 2019 utilisent conjointement l'apprentissage par imitation et par renforcement. Un réseau de neurones pour la planification de trajectoires et un réseau de contrôle sont appris de manière supervisée par imitation d'un algorithme de planification et d'un contrôleur géométrique. Ces deux réseaux sont ensuite réunis en un seul pour le contrôle d'un quadrotor et raffinés par apprentissage par renforcement (TRPO) en simulation.

## Conclusion

Les résultats de la section 3.6 montrent la possibilité de générer des commandes pertinentes pour la stabilisation d'un multirotor à l'aide d'une approche de commande optimale et en intégrant un modèle des perturbations de l'environnement. C'est pour cette raison que nous avons décidé de procéder à l'apprentissage d'un contrôleur par imitation.

Une approche d'apprentissage par renforcement aurait aussi pu être mise en œuvre en intégrant le modèle des perturbations dans un simulateur utilisé pour la collecte des données pour l'apprentissage par renforcement. Les algorithmes d'apprentissage par renforcement sont cependant plus difficiles d'usage que les algorithmes d'apprentissage supervisé. Les approches par renforcement ont une sensibilité plus grande aux hyperparamètres des algorithmes et sont plus difficiles à faire converger. L'approche par imitation permet de découpler le problème de recherche du contrôleur pertinent en celui de la recherche de commandes pertinentes, résolu via le problème de contrôle optimal, et celui de la recherche d'une fonction approximant ces commandes. Ce découplage permet de profiter de l'efficacité des outils du contrôle optimal pour ne pas résoudre en même temps les deux problèmes.

Les approches hybrides comme celles de S. LEVINE et KOLTUN 2013 ; KAHN et al. 2017 ; MORA et al. 2021 introduisent des mécanismes pour réintroduire un couplage plus restreint entre les deux problèmes et ainsi fournir une aide à l'algorithme de recherche du contrôleur.

Pour la résolution des problèmes abordés par cette thèse, l'utilisation de ces mécanismes supplémentaires n'a pas été nécessaire et nous nous sommes limités à une approche par imitation.

## 4.3 Apprentissage par imitation

Apprendre un contrôleur par imitation consiste à déterminer les paramètres qui caractérisent le contrôleur qui présente la plus faible différence avec un ensemble d'exemples que ce contrôleur doit imiter au mieux.

On note  $\mathcal{D} = \{(x, u) \in \mathbb{R}^{n \times m}\}$  un ensemble de couples (état, commande) obtenus grâce à un contrôleur *expert* donné  $\mathcal{C}_E$ , le contrôleur MPC par exemple.

On se donne  $\mathcal{C} = \{\mathcal{C}_\theta \in \mathcal{F}(\mathbb{R}^n \mapsto \mathbb{R}^m) | \theta \in \mathbb{R}^p\}$  un ensemble de contrôleurs caractérisés par un jeu de paramètres  $\theta$  et associant une commande à un état du robot.

Le problème de l'apprentissage par imitation consiste donc à chercher les paramètres  $\theta^*$  qui correspondent au contrôleur  $\mathcal{C}_{\theta^*}$  minimisant la distance avec  $\mathcal{C}_E$  pour l'ensemble des entrées possibles :

$$\theta^* = \arg \min_{\theta} \int_{x \in \mathbb{R}^n} |\mathcal{C}_E(x) - \mathcal{C}_\theta(x)|^2 p(x) dx \quad (4.1)$$

On peut pondérer l'importance des entrées en introduisant une distribution de probabilité sur ces états afin d'exprimer des préférences sur les états pour lesquelles les images de  $\mathcal{C}_E$  et  $\mathcal{C}_{\theta^*}$  doivent être les plus proches.

Une difficulté importante de ce problème réside dans le fait que cette distribution n'est pas nécessairement connue à l'avance. En particulier, quand il s'agit d'apprendre à imiter un contrôleur, la distribution des états rencontrés lors d'une trajectoire dépend du contrôleur. Ceci peut s'illustrer de la manière suivante : si l'application contrôleur  $\mathcal{C}_E$  dirige le robot vers une zone particulière de l'espace d'état, on peut souhaiter apprendre un contrôleur  $\mathcal{C}_\theta$  qui présentera un comportement similaire. Pour ça, il peut être nécessaire d'accorder une importance particulière aux états effectivement rencontrés par le contrôleur  $\mathcal{C}_\theta$ .

Si on possède déjà un ensemble de couples (état, commande) correspondant aux entrées et sorties du contrôleur  $\mathcal{C}_E$ , on peut procéder de la manière classique en recherchant par descente de gradient la valeur de  $\theta$  qui minimise l'écart entre  $\mathcal{C}_E$  et  $\mathcal{C}_\theta$  pour les couples connus. Une fois un contrôleur similaire obtenu, pour déterminer quels sont les états à privilégier pour obtenir un comportement similaire à  $\mathcal{C}_E$ , on peut réaliser une trajectoire à l'aide du contrôleur appris et ajouter un poids plus important aux états présents sur cette trajectoire, puis chercher à nouveau un contrôleur minimisant l'écart avec  $\mathcal{C}_E$  mais en utilisant ce poids plus important pour pondérer le coût total.

Cette démarche est celle de l'algorithme DAGGER (ROSS, GORDON et BAGNELL 2011). Plutôt que de procéder par une réévaluation de l'importance de certains états de l'ensemble, l'algorithme DAGGER propose d'ajouter à  $\mathcal{D}$  le couple  $(x, \mathcal{C}_E(x))$  pour chaque état  $x$  rencontré sur la trajectoire. Ainsi, l'ensemble des couples (état, commande) à partir duquel on minimise la distance au contrôleur  $\mathcal{C}_E$  comprend des états vus par le contrôleur appris. Le détail des étapes de cet algorithme est décrit par l'algorithme 4.

L'utilisation d'un contrôleur MPC comme contrôleur expert pour le calcul d'une commande optimale dans un état donné revient en fait à un calcul de la trajectoire optimale sur un horizon donné. Dans un contrôleur MPC, seule la première de ces commandes est appliquée puis le contrôleur résout à nouveau le problème de contrôle optimal. À la différence de l'algorithme DAGGER, j'ai donc choisi d'intégrer les couples (état, commande) de la trajectoire optimale sur toute sa durée à l'ensemble  $\mathcal{D}$  plutôt qu'uniquement le premier. Lors de l'étape 5, pour chaque état  $x_{i,j}$  de  $T_i$  le contrôleur MPC calcule une trajectoire optimale  $T_{i,j}$  et la commande correspondante. Tous les couples (état, commande) de  $T_{i,j}$  sont ajoutés à  $\mathcal{D}$  au lieu de ne garder que le premier couple de chaque trajectoire.

J'ai essayé une version plus proche de l'algorithme original pour laquelle je n'ajoute que le premier couple (état, commande) de  $T_{i,j}$  à  $\mathcal{D}$ . À nombre d'itérations égal cette version n'a pas

---

**Algorithme 4** DAGGER

---

- 1: Initialisation d'un contrôleur  $\mathcal{C}_{\theta_0}$
  - 2: Initialisation d'un ensemble  $\mathcal{D}$
  - 3: **for**  $i \in [0, k]$  **do**
  - 4:   Génération d'une trajectoire  $T_i = (x_{i,j})_j$  à l'aide de  $\mathcal{C}_{\theta_i}$
  - 5:   Génération des commandes de l'expert  $\mathcal{D}_i = \{(x, \mathcal{C}_E(x)) \mid x \in T_i\}$
  - 6:   Ajout des nouvelles données à  $\mathcal{D} = \mathcal{D} \cup \mathcal{D}_i$
  - 7:   Apprentissage d'un contrôleur sur les nouvelles données
  - 8:    $\theta_{i+1} = \arg \min_{\theta} \sum_{(x,u) \in \mathcal{D}} |u - \mathcal{C}_{\theta}(x)|^2$
  - 9: **end for**
- 

permis l'apprentissage d'un contrôleur imitant suffisamment précisément le contrôleur expert pour stabiliser le vol du robot.

Une approche consistant à biaiser le problème contrôle optimal pour qu'il produise des commandes proches de celles du contrôleur courant a été explorée avec l'algorithme Guided Policy Search (S. LEVINE et KOLTUN 2013 ; ZHANG et al. 2016) et l'algorithme PLATO (KAHN et al. 2017). Ce biais du contrôleur expert vers des commandes que pourrait réaliser le contrôleur appris permet une meilleure convergence de l'apprentissage et pourrait réduire le nombre d'itérations nécessaire pour obtenir un contrôleur satisfaisant. Cependant, ajouter une pénalité de distance au contrôleur courant, comme le fait l'algorithme Guided Policy Search, rend le problème de contrôle optimal plus difficile puisqu'il nécessite d'optimiser aussi à travers le contrôleur. Pour cette raison, je n'ai pas intégré cette approche dans l'algorithme présenté dans ce chapitre.

Comme précisé dans l'algorithme 4, aucun couple (état, commande) n'est retiré de l'ensemble d'apprentissage. Ceci est rendu possible par le fait que chacun de ces couples contient une commande pour l'état correspondant. Ainsi, même si les états ne sont plus visités par le contrôleur courant, les couples collectés dans les premières itérations sont pertinents vis-à-vis du comportement du contrôleur appris.

Les commandes successives de cette trajectoire ne sont pas toutes aussi optimales les unes que les autres. Les premières commandes sont optimales vis à vis d'un horizon plus long que les suivantes, et cette optimalité est donc de plus en plus locale. Cependant, l'utilisation de ces commandes s'est révélée utile expérimentalement. Une hypothèse qui pourrait justifier cette utilité constatée réside dans la capacité de ce contrôleur optimal à stabiliser le robot à vitesse et inclinaison nulle. Il est possible que plus le robot soit proche d'un tel état, moins l'obtention d'une commande satisfaisante (par résolution du problème de contrôle optimal) ne nécessite un horizon long. Ainsi, un horizon court serait suffisant pour les états dont l'inclinaison et la vitesse sont proches de 0. Ce sont ces états qui sont en fin de trajectoire et qui sont ajoutés au jeu d'apprentissage, malgré une différence dans la nature de leur optimalité.

Pour contrôler le nombre d'états ajoutés à l'ensemble  $\mathcal{D}$  à chaque itération, je détaille avec l'algorithme 5 l'étape 5 de l'algorithme 4. Plutôt que de générer une commande experte pour chaque état de la trajectoire  $T_i$  on extrait un nombre donné d'états répartis uniformément sur cette trajectoire.

L'espace des états est vaste. Pour que l'ensemble de données d'apprentissage puisse couvrir cet espace et que leur collecte puisse se faire en un temps raisonnable, j'ai choisi de collecter des données correspondant uniquement à un contexte où le contrôleur optimal tente de stabiliser le robot à une vitesse nulle, à l'origine et à l'horizontale, comme décrit dans la section 2.4.3.3. L'adaptation d'un contrôleur qui stabilise pour une tâche de suivi de trajectoire est abordée dans la section 4.7.

En conséquence, les trajectoires générées par le contrôleur expert tenteront uniquement de stabiliser le robot. Pour assurer que l'ensemble d'apprentissage contient les données suffisantes pour permettre la stabilisation du robot, l'algorithme que je détaille perturbe les états initiaux à partir desquels sont calculées les commandes expertes. Ceci permet en particulier une couverture du voisinage des états rencontrés par le contrôleur appris, et en particulier de l'état de stabilisation du robot. L'algorithme 5 détaille l'intégration de ces perturbations dans l'étape 5 de l'algorithme 4.

---

**Algorithme 5** Protocole de collecte de données pour l'algorithme d'apprentissage par imitation.

---

- 1: Extraction de  $n$  états de la trajectoire  $T_i$  uniformément espacés dans le long de celle ci
  - 2: Application d'une perturbation à ces  $n$  états
  - 3: Résolution d'un problème de contrôle optimal à partir de chacun de ces états perturbés
  - 4: Application des commandes obtenues en partant de chacun des  $n$  états perturbés dans un environnement simulé sans perturbations aléatoires (i.e. selon le modèle connu par le contrôleur, mais en assurant la normalisation des grandeurs non vectorielles comme l'orientation)
  - 5: Ajout de chacun des couples (état,commande) pour chacune de ces trajectoires à l'ensemble  $\mathcal{D}$
- 

## 4.4 Modèles et architecture des contrôleurs

L'algorithme présenté dans la section 4.3 recherche un contrôleur dans un ensemble de fonctions paramétrées. De nombreux choix sont possibles pour  $\mathcal{C}$ , qu'il s'agisse de polynômes (paramétrés par leurs coefficients), de fonctions linéaires, ou d'autres.

Nous avons choisi d'utiliser comme classe de contrôleurs des réseaux de neurones artificiels (HAYKIN 1999). Ces fonctions sont génériques : comme pour les polynômes elles peuvent approximer n'importe quelle fonction pour peu que leur taille soit suffisante (HORNIK, STINCHCOMBE et WHITE 1989). Leur usage est par ailleurs très largement répandu dans le domaine de l'apprentissage automatique, qu'il soit supervisé ou par renforcement et ces fonctions ont en particulier été utilisées comme contrôleur pour des robots volants dans de nombreux travaux (MOLCHANOV et al. 2019 ; HWANGBO et al. 2017 ; KOCH et al. 2019 ; LAMBERT et al. 2019 ; ZHANG et al. 2016 ; KAHN et al. 2017 ; MORA et al. 2021). Je fais une rapide description du fonctionnement des réseaux de neurones dans la section 3.5.2.

L'état d'entrée que choisi pour le contrôleur appris est le même que celui décrit pour le contrôleur MPC et du contrôleur proportionnel (cf. section 2.4.3), c'est-à-dire l'état du robot : la position,

la vitesse, l'attitude, la vitesse angulaire et l'intensité des forces produite par chaque moteur. Les sorties du contrôleur sont les commandes appliquées au robot, c'est-à-dire la variation des intensités moteur.

Pour limiter les dimensions de l'entrée nous avons choisi de limiter le contrôleur à une architecture purement réactive : celui-ci ne reçoit en entrée que l'état courant et ne stocke pas en mémoire les états passés. L'attitude est représentée par un quaternion unitaire plutôt qu'une matrice de rotation. Ceci permet de représenter l'attitude sur 4 dimensions plutôt que 9. Les équations de passage d'une représentation à l'autre sont décrites dans SOLÀ, DERAY et ATCHUTHAN 2021. Par ailleurs, afin d'augmenter la capacité de ce contrôleur à faire face à de nouvelles situations, la position et la vitesse sont exprimées dans le système de coordonnées du robot. Ceci rend le lien entre la commande et l'entrée plus court d'une étape puisque les actionneurs du robot agissent dans le système de coordonnées du robot.

Ces choix pour l'architecture fixent le nombre d'entrées du réseau à 17 (position 3, vitesse 3, attitude 4, vitesse angulaire 3, moteur 4) et le nombre de sorties à 4, une pour contrôle chacun des moteurs. Le nombre de neurones dans chaque couche cachée ainsi que le nombre de couches cachées n'est pas contraint par ces choix. Pour limiter la dimension de l'espace de recherche du contrôleur, c'est-à-dire le nombre de paramètres, j'ai choisi de limiter le réseau à deux couches cachées de 64 neurones chacune. Cette architecture est d'une taille similaire à celle des contrôleurs utilisés pour le contrôleur de quadrotor dans les travaux déjà réalisés (MOLCHANOV et al. 2019 ; HWANGBO et al. 2017 ; CANO LOPES et al. 2018). Cette architecture comprend donc

$$(17 + 1) \times 64 + (64 + 1) \times 64 + (64 + 1) \times 4 = 5572 \quad (4.2)$$

paramètres. Pour chaque liaison d'une couche à  $n_1$  paramètres vers une couche à  $n_2$  paramètre possède  $n_1 \times n_2$  paramètres correspondants aux coefficients de la combinaison linéaire plus  $n_2$  paramètres pour la partie affine de la combinaison souvent appelée biais.

D'autres architectures, plus étendues ou plus réduites pourraient être explorées, mais l'augmentation du nombre de paramètres peut entraîner une augmentation de la quantité de données nécessaires pour l'apprentissage et allonger ainsi le temps d'apprentissage. La fonction d'activation utilisée dans le réseau est tanh, mais des essais préliminaires avec ReLU ont donné des résultats semblables à ceux décrits plus loin.

La figure 4.1 résume les choix d'architecture pris pour le contrôleur.

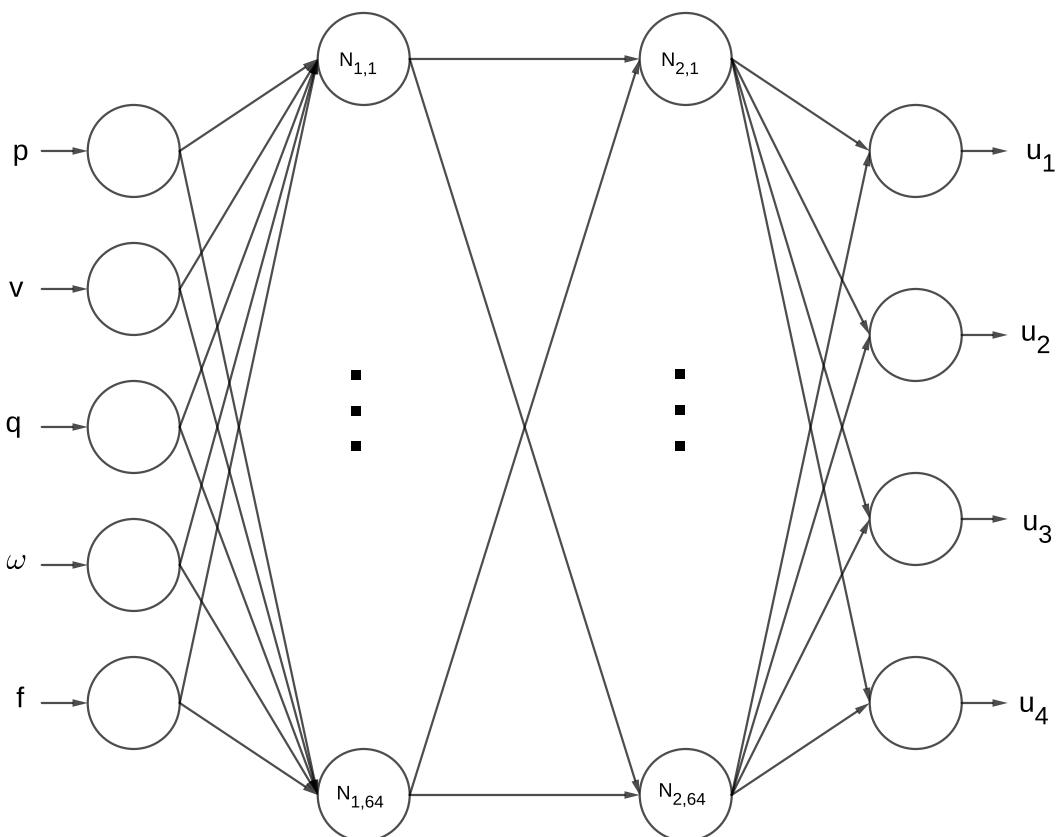


FIGURE 4.1 – Architecture du contrôleur du quadrotor. Ce réseau dispose de deux couches cachées comprenant chacun 64 neurones. Les entrées du réseau sont la position (p), la vitesse (v), l'orientation (q), la vitesse angulaire ( $\omega$ ) et la force exercée par chaque rotor (f). Les sorties correspondent à la variation des forces exercées par chaque rotor, c'est-à-dire à la commande du drone telle que définie dans le chapitre 2.

## 4.5 Résultats expérimentaux pour l'apprentissage d'un contrôleur de stabilisation

Je détaille dans cette section les paramètres utilisés pour l'apprentissage d'un contrôleur pour la stabilisation d'un quadrotor à l'aide de l'approche décrite dans la section 4.3 et avec une architecture décrite dans la section 4.4.

Comme pour le contrôleur MPC présenté dans la section 2.4.3.3, le contrôleur optimal est calculé sur un horizon de 1s avec une fréquence de 20Hz, c'est-à-dire que la trajectoire optimale contient 20 états. Il est nécessaire de formuler le problème de contrôle optimal sur une durée assez longue au regard du temps caractéristique de la dynamique du robot. Un problème formulé avec un horizon de 0.1s pourrait avoir une solution qui serait trop locale pour qu'il soit pertinent de l'appliquer pour une trajectoire qui dure en réalité plus longtemps. En pratique, un horizon dont la durée est supérieure à 0.5s semble suffisant pour obtenir des solutions exploitables.

Par ailleurs, vu le caractère uniquement réactif du contrôleur appris, le coût de l'écart à la vitesse optimisé par le contrôle optimal (décrit dans la section 2.4.3) est modifié de  $10^{-3}$  à 1 afin d'aider à une forme de planification.

Le nombre d'états extrait de chacune des trajectoires, et donc le nombre de problèmes de contrôle optimal résolu à chaque itération de l'algorithme 4 est fixé à 1000. Ce qui signifie que chaque itération de l'algorithme 4 ajoute 20 000 couples (état commande) à l'ensemble d'apprentissage.

Les trajectoires dont sont extraits les états initiaux s'étendent sur 10s. Celles-ci sont cependant interrompues prématurément si le robot s'écarte trop de l'origine (plus de 20m, de  $40m.s^{-1}$  ou de  $80\text{rad}.s^{-1}$ ). L'état initial de chacune de ces trajectoires est lui-même aléatoirement de la manière suivante :

- la position initiale est distante d'au plus 0.2m de l'origine ;
- la vitesse initiale est d'au plus  $0.5m.s^{-1}$  ;
- le robot est tourné d'au plus  $30^\circ$ .

Durant cette trajectoire le robot subit des perturbations externes comme celles décrites dans la section 2.4.3 dont les forces dont des intensités inférieures à  $1.10^{-2}N$  et les couples à  $4.10^{-4}Nm$ . L'algorithme d'apprentissage réalise  $k = 16$  itération pour l'apprentissage d'un contrôleur. On observe sur la figure 4.2 que le processus d'apprentissage converge pour un nombre d'itérations inférieur.

Comme pour l'apprentissage de modèles des perturbations, le taux d'apprentissage, c'est-à-dire la distance parcourue dans le sens du gradient, est mis à jour via l'algorithme ADAM (KINGMA et BA 2015). Pour chaque itération, les paramètres du contrôleur sont appris durant 500 epochs avec des mini-batchs de taille 4096. C'est à dire que les paramètres du réseau sont mis à jour à partir d'une erreur agrégée pour des groupes de 4096 couples (état, commande) et que chaque couple (état, commande) est vu 500 fois pour une itération d'apprentissage.

La figure 4.2 présente la progression du processus d'apprentissage. À la fin de chaque itération, une copie du contrôleur courant est réalisée. Grâce à cette copie du contrôleur tel qu'il était durant le processus d'apprentissage, on peut réaliser des trajectoires et évaluer l'évolution de la qualité du contrôleur durant l'apprentissage. De ces trajectoires on calcule une erreur, le coût optimisé par le contrôle optimal. Je présente donc la distribution de cette erreur pour chaque itération. Cette distribution est réalisée à partir de 20 états initiaux et perturbations.

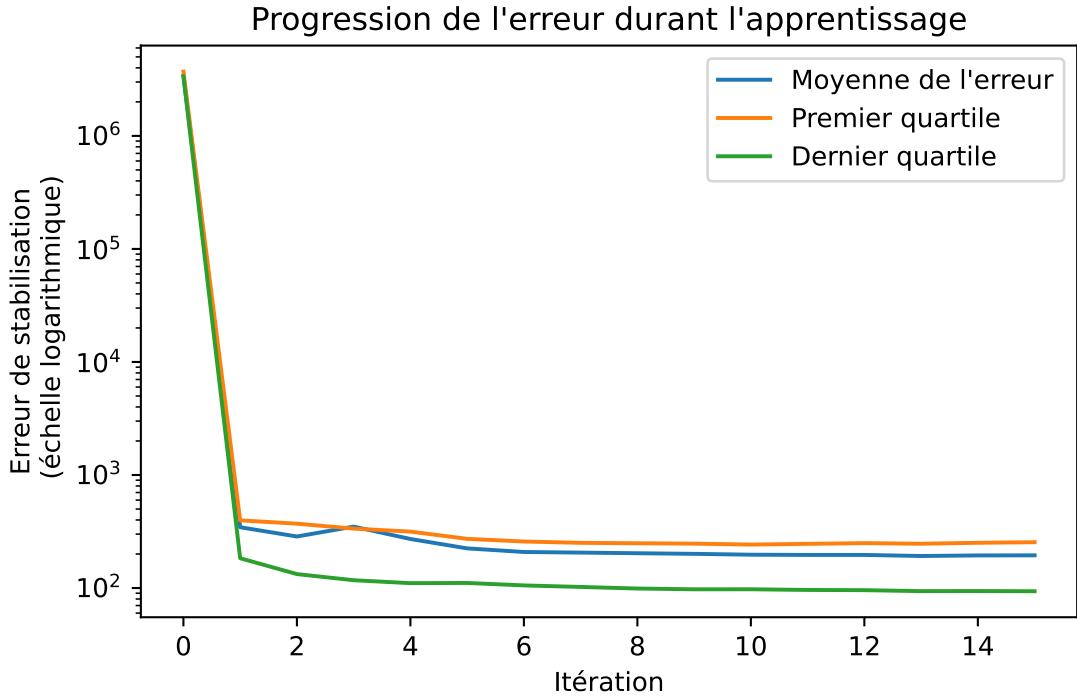


FIGURE 4.2 – Erreurs réalisées par le contrôleur appris pour chaque itération de l'algorithme. Cette erreur est celle définie pour une trajectoire pour le problème de contrôle optimal et est détaillée dans la section 2.4.3. Ces erreurs sont agrégées pour 20 trajectoires avec des états initiaux et des perturbations échantillonées de manière indépendante. Une échelle logarithmique est utilisée pour permettre de distinguer la convergence après la 5<sup>e</sup> itération. L'axe des ordonnées représente le coût d'une trajectoire qui est sans unité.

Ceci remplace une évaluation du réseau de neurone sur un ensemble de test contenant des couples (état, commande) non utilisés durant l'apprentissage. On observe que la première itération améliore grandement le contrôleur initial (l'échelle est logarithmique), et qu'ensuite les améliorations sont plus réduites, mais se poursuivent. Ceci est dû au fait que la capacité à stabiliser la trajectoire réduit de manière très importante le coût : le quadrotor étant un système instable, son état diverge rapidement. Une fois le contrôleur capable de stabiliser le robot, des améliorations sont encore possibles, mais celles-ci ont un impact bien plus faible sur l'erreur totale de la trajectoire.

Une fois le processus d'apprentissage réalisé, on peut comparer les trajectoires réalisées avec le contrôleur appris, dans sa version finale, à celles réalisées par le contrôleur MPC et le contrôleur proportionnel. La figure 4.3 présente l'exemple d'une de ces trajectoires réalisée pour des profils de perturbations et des états initiaux similaires à ceux des trajectoires utilisées durant l'apprentissage. On y observe une proximité importante entre le contrôleur appris par imitation et le contrôleur MPC qui est imité. Le contrôleur appris stabilise le coût et la position bien mieux que le contrôleur proportionnel et presque aussi bien que le contrôleur MPC.

La figure 4.4 présente une comparaison de la distribution de l'erreur pour 20 trajectoires pour chacun des trois contrôleurs. Le contrôleur MPC et le contrôleur appris par imitation ont une

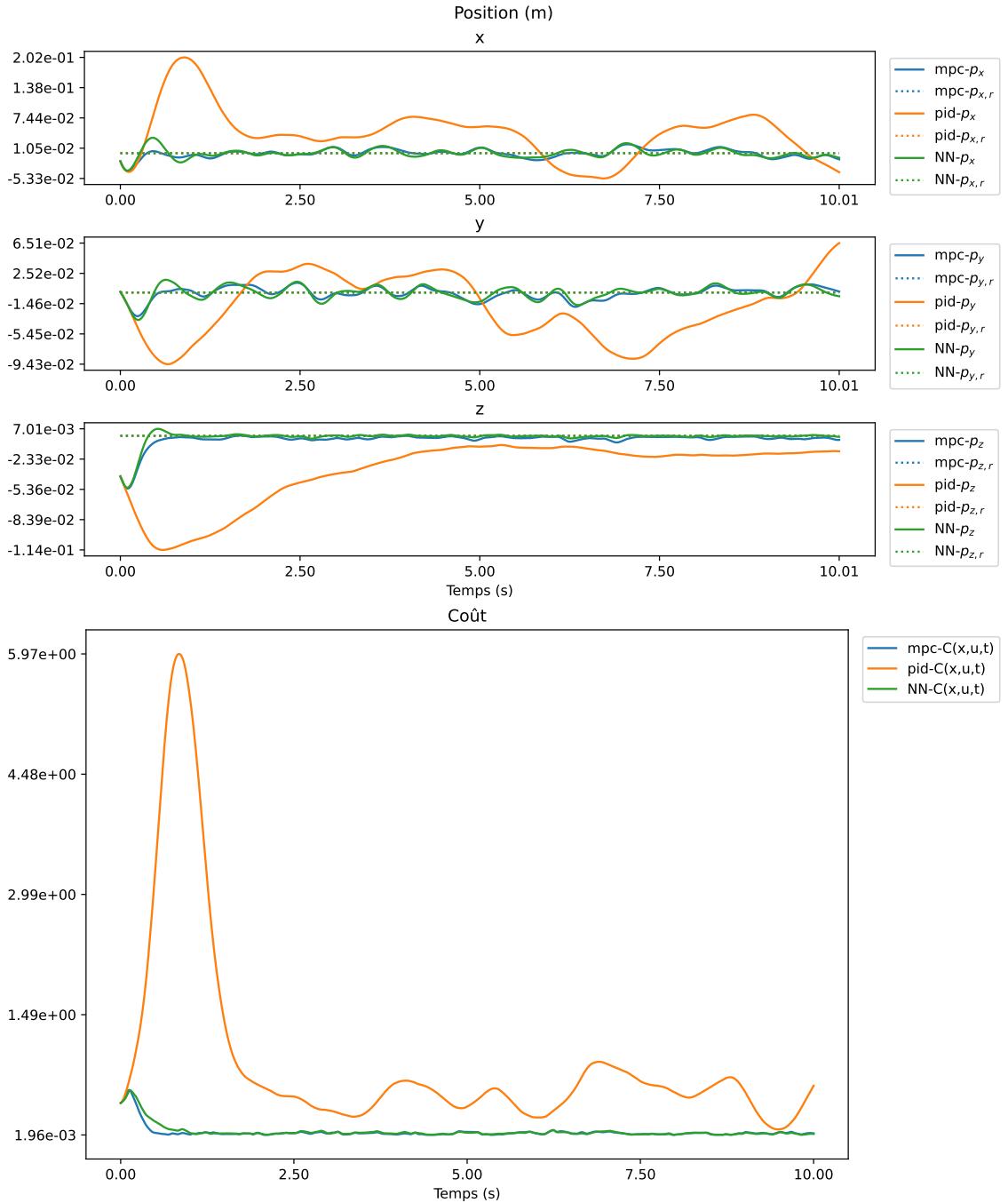


FIGURE 4.3 – Comparaison du contrôleur appris avec le contrôleur proportionnel et le contrôleur MPC. L'état de référence est indiqué par un trait pointillé et l'indice  $r$ . Quand cet état de référence n'est pas visible c'est qu'il est couvert par la courbe de la trajectoire et que ceux-ci coïncident. Les trois figures du haut présentent l'évolution de la position, et celle du bas présente l'évolution du coût de l'état courant. Le profil de position du contrôleur appris suit de très près celui du contrôleur MPC et il en est de même pour le profile de coût.

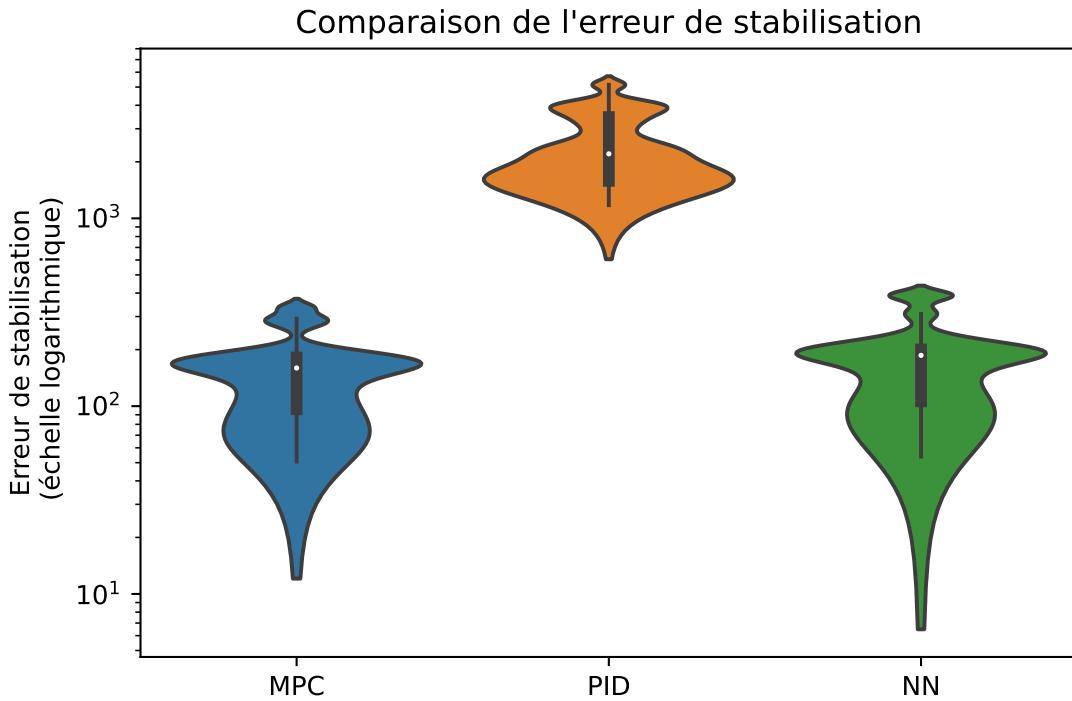


FIGURE 4.4 – Distribution de l'erreur de stabilisation pour le contrôleur MPC (bleu), proportionnel (PID, orange) et le contrôleur appris par imitation (NN, vert). La densité de la distribution à une ordonnée fixée est donnée par la distance jusqu'où s'étend la zone colorée. Plus celle-ci s'éloigne du centre plus la distribution présente est dense. Le contrôleur MPC et le contrôleur appris présentent une distribution similaire. L'erreur pour le contrôleur proportionnel est plus importante que celles deux autres contrôleurs.

distribution similaire. L'erreur du contrôleur appris est un peu plus haute que celle du contrôleur MPC, mais bien plus basse que celle du contrôleur proportionnel. Cette figure confirme la similarité de comportement entre le contrôleur appris et le contrôleur MPC déjà observée sur la figure 4.3. Ces deux figures montrent que l'apprentissage par imitation d'un contrôleur basé contrôle optimal permet d'obtenir une bonne approximation de ce contrôleur.

Le contrôleur appris a un comportement satisfaisant en ce qui concerne la stabilisation du robot. Son coût en calcul, qui la raison pour laquelle nous avons décidé de procéder à l'imitation du contrôleur MPC, est bien inférieur à celui du contrôleur MPC. La figure 4.5 présente la distribution du temps de calcul nécessaire à chaque contrôleur pour calculer la commande pour 1s de trajectoire. Ces valeurs sont obtenues à partir du temps de calcul total nécessaire au contrôleur pour une trajectoire, divisé par la durée de la trajectoire. Pour chaque contrôleur, 20 trajectoires de stabilisation ont été réalisées pour obtenir les distributions présentées.

On observe que le contrôleur MPC a besoin de 500s pour calculer la commande pour 1s, c'est-à-dire qu'il tourne 500 fois trop lentement pour le temps réel. Le contrôleur proportionnel tourne 20 fois plus vite que le temps réel et le contrôleur appris tourne deux fois plus vite que le temps réel. Ces trajectoires ont été calculées sur une machine fixe et non à bord du robot, et la figure 4.5 ne présente donc pas la preuve que le contrôleur appris peut être embarqué. Des essais préliminaires réalisés par Adrien Guénard, ingénieur au LORIA indiquent cependant qu'il

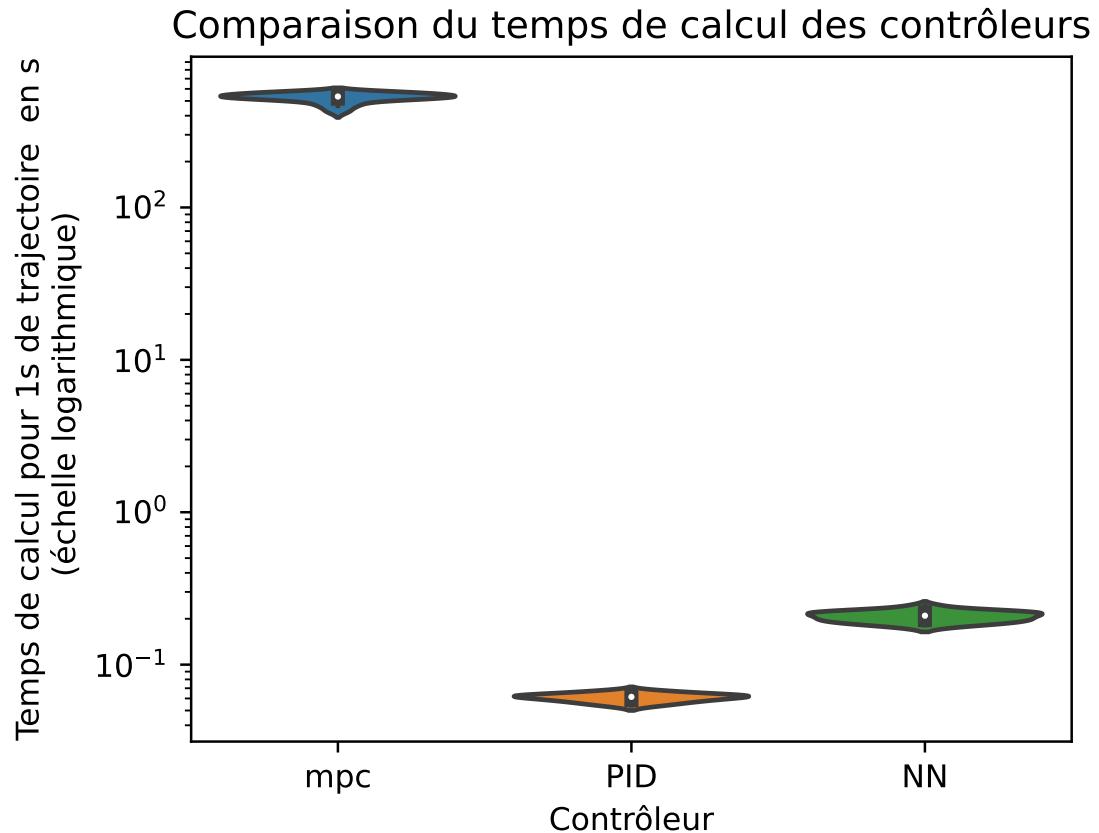


FIGURE 4.5 – Distribution du temps nécessaire pour calculer 1s de trajectoire pour le contrôleur MPC (bleu), proportionnel (PID, orange) et le contrôleur appris par imitation (NN, vert). La densité de la distribution à une ordonnée fixée est donnée par la distance jusqu' où s'étend la zone colorée. Plus celle-ci s'éloigne du centre plus la distribution présente est dense. Cette distribution est calculée à partir de 20 trajectoires de stabilisation pour chaque contrôleur. L'axe des ordonnées représente le coût d'une trajectoire qui est sans unité.

est possible de faire tourner en temps réel un tel contrôleur à bord du Crazyflie.

Le temps de calcul très important que demande le contrôleur MPC peut s'expliquer par plusieurs points.

Le premier concerne la non réutilisation des trajectoires optimale calculée pour l'instant précédent. Une approche classique dans les contrôleurs MPC consiste à utiliser cette trajectoire de l'instant précédent comme solution initiale à partir de laquelle chercher la nouvelle trajectoire optimale. Je ne réutilise pas ces trajectoires pour des raisons de complexité technique dans le cas où le pas de temps du contrôleur MPC et celui du contrôle optimal sur lequel est basé le MPC. Quand ces deux pas de temps ne sont pas égaux, ceci demande une transformation de la trajectoire optimale précédente.

Un deuxième point réside probablement dans le solveur utilisé pour résoudre le problème non linéaire de contrôle optimal. KAUFMANN et al. 2019 utilisent aussi CasADi et IPOPT pour leur contrôleur MPC en simulation, mais pour le déploiement de ce contrôleur MPC à bord d'un

robot ils changent pour la bibliothèque ACADO (Houska, Ferreau et Diehl 2011) et le solveur qpOASES(FERREAU et al. 2014) et optent pour un contrôle à 50Hz. Le fait que ce contrôle soit réalisé à 50Hz sur un robot disposant de capacités de calcul embarquées bien plus importante a confirmé notre estimation initiale des difficultés importantes pour embarquer un contrôleur MPC à bord du Crazyflie. L'utilisation d'un contrôleur MPC déporté pour le Crazyflie par CARLOS et al. 2020 confirme aussi cette estimation.

## 4.6 Influence de la prise en compte de l'environnement

L'introduction d'un modèle de l'environnement est une des motivations pour l'utilisation d'un contrôleur basé sur la résolution du problème de contrôle optimal. L'apprentissage par imitation peut bien entendu s'appliquer aussi pour un cas où le problème de contrôle optimal tient compte d'un modèle de l'environnement. On procède de manière analogue à ce qui est décrit dans la section 3.6 pour intégrer le modèle des perturbations d'un tuyau de diamètre 65cm à base de processus gaussiens. L'algorithme d'apprentissage par imitation, ses paramètres et son contexte de mise en œuvre sont identiques à celui de la section précédente.

Au vu des analyses des perturbations réalisées dans la section 3.4, une consigne de stabilisation au centre du tuyau ne pourra pas illustrer pleinement l'intérêt d'une prise en compte de l'environnement par le contrôleur. Les mesures de perturbations au centre du tuyau montrent en effet qu'à cet endroit seule une force verticale vers le bas s'exerce sur le robot.

Cependant, l'état initial de la trajectoire reste choisi au hasard, et lors de la navigation depuis cet état vers l'état de stabilisation, les contrôleurs doivent tenir compte des perturbations dont la direction et l'intensité sont susceptibles de changer. En outre, section 4.7 présente des tâches de suivi de trajectoire qui permettent d'évaluer de manière plus complète le comportement du contrôleur appris par imitation dans un environnement tuyau selon la connaissance d'un modèle d'environnement ou non.

La figure 4.6 présente le profil de position et de coût pour des trajectoires réalisées par des contrôleurs MPC et appris avec et sans intégration d'un modèle des perturbations. Le contrôleur proportionnel, même quand il intègre un modèle de l'environnement, s'écarte nettement plus de la position de référence à laquelle le robot est censé se stabiliser. Le coût de du contrôleur est donc aussi bien plus élevé. Pour des raisons de lisibilité, les contrôleurs MPC et appris sont donc les seuls à être présentés sur cette figure.

Comme attendu, on observe peu de différence sur les axes horizontaux entre les contrôleurs intégrant le modèle de perturbation et ceux n'en ayant pas connaissance. Sur ces axes, il y a une petite différence entre les contrôleurs MPC et les contrôleurs appris. Sur l'axe vertical, on observe une nette différence entre les contrôleurs intégrant un le modèle des perturbations qui parviennent à se stabiliser au centre du tuyau, et ceux n'ayant pas connaissance de cet environnement qui se stabilisent, mais un peu sous le centre du tuyau. Ceci est la conséquence des perturbations au centre du tuyau qui poussent le robot vers le bas, comme présenté dans la section 3.4. L'effet de cette perturbation sur le coût domine les petites différences entre contrôleurs MPC et appris sur les axes horizontaux, et les coûts des trajectoires des contrôleurs appris ressemblent à ceux des contrôleurs qu'ils imitent.

La figure 4.7 présente la distribution de l'erreur pour les trois contrôleurs (MPC, appris et proportionnel) selon l'intégration ou non d'un modèle des perturbations dues à l'environnement.

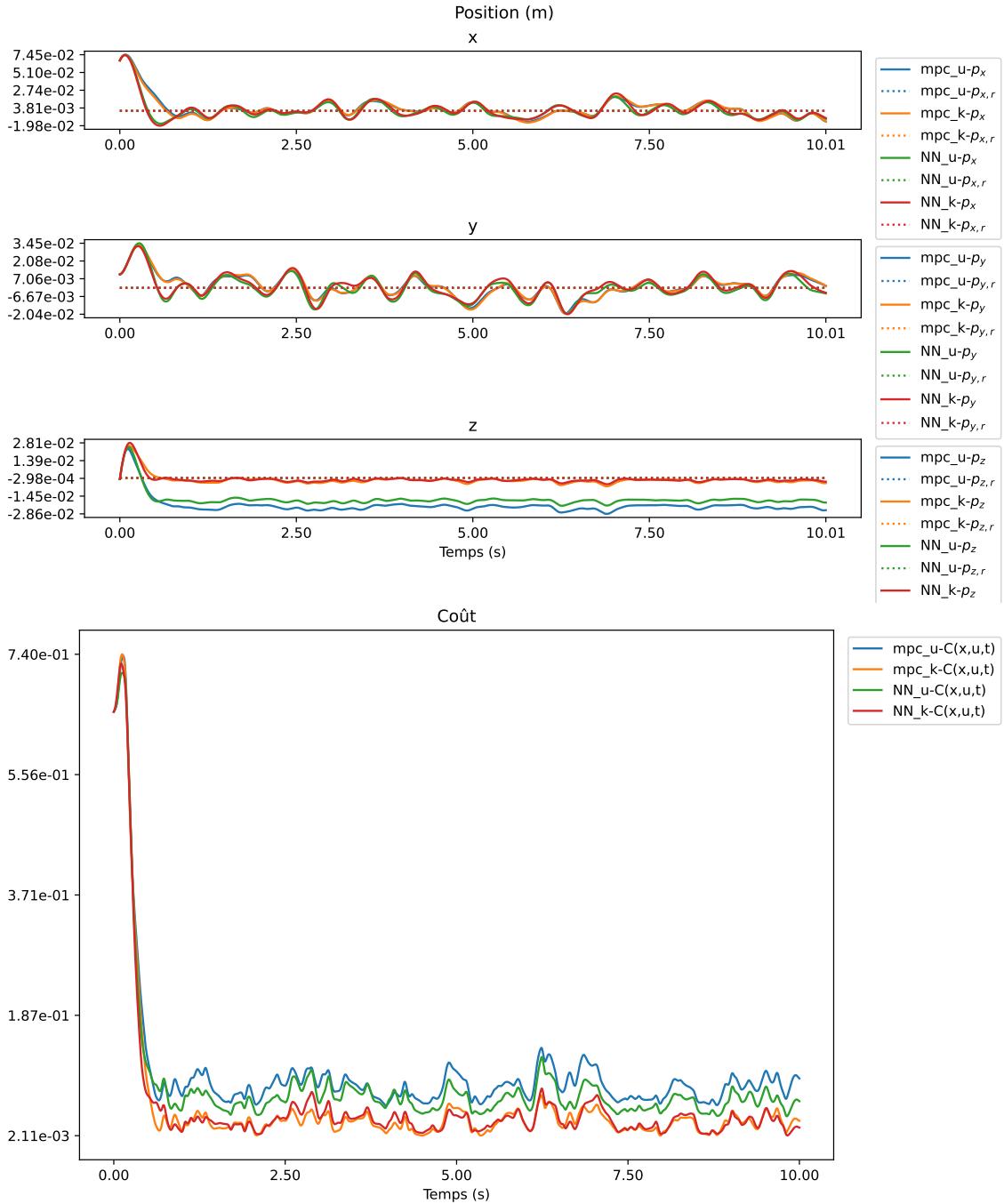


FIGURE 4.6 – Comparaison du contrôleur appris (NN) avec le contrôleur MPC, avec ( $_k$ ) et sans connaissance des perturbations de l'environnement ( $_u$ ). Les trajectoires commencent d'un état initial identique et subissent les mêmes perturbations externes. L'état de référence est indiqué par un trait pointillé et l'indice  $r$ . Quand cet état de référence n'est pas visible c'est qu'il est couvert par la courbe de la trajectoire et que ceux-ci coïncident. Les trois figures du haut présentent l'évolution de la position, et celle du bas présente l'évolution du coût de l'état courant. Sur les axes latéraux (x et y) les deux contrôleurs MPC ont le même comportement et les deux contrôleurs appris ont le même comportement. Sur l'axe vertical, les deux contrôleurs avec connaissance d'un modèle des perturbations ont un comportement similaire et les deux contrôleurs sans cette connaissance ont un comportement similaire.

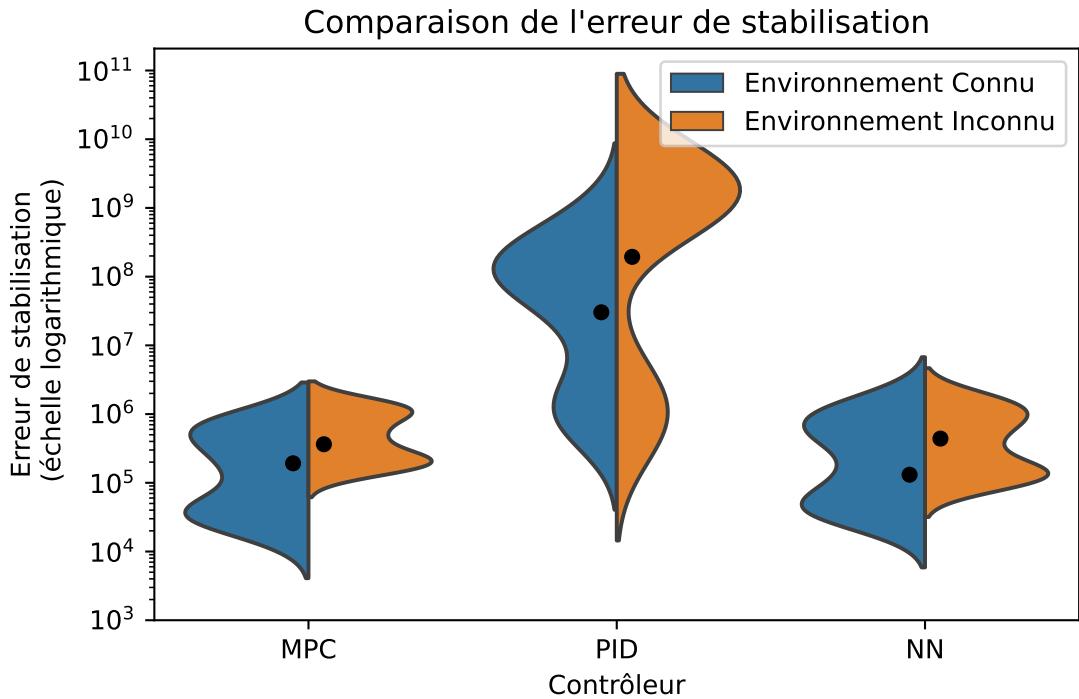


FIGURE 4.7 – Distribution de l'erreur de stabilisation pour le contrôleur MPC, proportionnel (PID) et le contrôleur appris par imitation (NN) dans tuyau de 65cm de diamètre. La densité de la distribution à une ordonnée fixée est donnée par la distance jusqu'où s'étend la zone colorée. Plus celle-ci s'éloigne du centre plus la distribution présente est dense. Le contrôleur MPC et le contrôleur appris présentent une distribution similaire. L'erreur pour le contrôleur proportionnel est plus importante que celles deux autres contrôleurs. L'erreur pour les contrôleurs intégrant un modèle de l'environnement est plus faible. Le cercle noir au cœur de la distribution signale la valeur moyenne de celle-ci. L'axe des ordonnées représente le coût d'une trajectoire qui est sans unité.

On observe, que les trajectoires réalisées avec le contrôleur proportionnel ont une erreur bien plus importante. Par ailleurs, comme pour les données en dehors de l'environnement, le contrôleur MPC et le contrôleur appris par imitation ont des erreurs distribuées de manière similaire. Le contrôleur appris présente des erreurs un peu plus importantes que le contrôleur MPC qu'il imite.

On observe aussi que l'intégration d'un modèle de l'environnement dans le contrôle améliore l'erreur des trajectoires, mais que cette amélioration est bien plus nette pour le contrôleur MPC et le contrôleur appris. Dans le cas du contrôleur proportionnel, la connaissance du modèle réduit la dispersion de l'erreur et réduisant la densité des trajectoires dont l'erreur est importante alors que dans le cas du contrôleur MPC et du contrôleur appris, la dispersion est augmentée en conséquence d'une augmentation importante des trajectoires dont le coût est plus faible.

On observe donc que l'intégration d'un modèle de l'environnement se transfère bien vers le contrôleur appris qui est donc capable de tenir compte du modèle des perturbations statiques élaboré à partir des mesures dont la collecte est décrite dans la section 3.3.

## 4.7 Adaptation pour le suivi de trajectoire

Les contrôleurs proportionnels et MPC permettent le suivi de trajectoires données en argument. La tâche de stabilisation est d'ailleurs un cas particulier d'une trajectoire constante dans le temps. Le contrôleur appris décrit dans ce chapitre n'a appris qu'à suivre cette trajectoire constante et est incapable en l'état de suivre une autre trajectoire.

Pour permettre au contrôleur appris de suivre une trajectoire, une extension immédiate serait d'ajouter des entrées au réseau pour indiquer un état de référence. Les contrôleurs proportionnel et MPC ont une telle entrée à partir de laquelle ils calculent la commande pour atteindre cet état de référence.

Cependant, cette augmentation de l'espace d'état nécessite que l'ensemble de données sur lequel le contrôle est appris couvre cette diversité de nouveaux états. Le nombre de points pour couvrir un espace augmente exponentiellement avec la dimension de celui-ci, en conséquence, l'ajout aux entrées du réseau d'un état de référence augmenterait de manière importante la quantité de données nécessaires pour apprendre le contrôleur par imitation. Vu le caractère synthétique de la génération des données d'apprentissage décrite dans ce chapitre, cela serait possible, mais demanderait une puissance de calcul importante et beaucoup de temps.

Comme esquissé en début de ce chapitre, j'ai plutôt choisi de maintenir cet espace d'état réduit en faisant l'hypothèse que le contrôleur appris avait pour objectif de ramener l'état du robot à l'origine. Pour réaliser le suivi d'une trajectoire, je propose de me ramener au problème de stabilisation en réalisant une transformation de l'état donné en entrée au contrôleur appris.

Pour ceci, il suffit que la transformation appliquée préalablement à l'état fourni au contrôleur appris soit la différence entre l'état du robot et l'état de référence. Le principe de cette transformation est très répandu et c'est en particulier ce principe qui est utilisé pour que le contrôleur proportionnel puisse suivre une trajectoire.

Pour l'orientation du robot, la transformation ne réalise pas directement une différence entre les deux représentations cette orientation sous forme de quaternions unitaires, mais plutôt la multiplication de l'orientation courante par l'inverse de l'orientation de référence. Cette transformation représente la rotation qu'il faut réaliser pour atteindre l'orientation de référence à partir de l'orientation courante.

Cette transformation permet donc de garder la partie apprise de l'architecture du contrôleur identique et de n'y ajouter qu'une transformation en amont de cette partie apprise.

L'équation (4.3) détaille cette transformation. L'opérateur  $\cdot$  représente la multiplication de quaternion et correspond ici à une composition de rotations. La rotation résultante  $\tilde{q}(q, q_r)$  représente la rotation nécessaire pour transformer l'orientation de référence en l'orientation courante.

$$\begin{aligned}\tilde{p}(p, p_r) &= p - p_r \\ \tilde{v}(v, v_r, p, p_r) &= v - v_r \\ \tilde{q}(q, q_r) &= q \cdot q_r^{-1} \\ \tilde{\omega}(\omega, \omega_r) &= \omega - \omega_r\end{aligned}\tag{4.3}$$

Dans le cas où l'état de référence est celui décrit par l'équation 2.24, cette transformation est neutre : elle ne change pas l'état courant du robot.

La figure 4.8 montre un suivi de trajectoire pour chacun des trois contrôleurs en l'absence de perturbations du tuyau. La trajectoire de référence est celle présentée dans la section 3.6 avec

une amplitude de 20cm pour les consignes de position. Des perturbations aléatoires comme celles décrites dans la section 2.4.3.4 sont appliquées durant le vol.

On peut observer que le contrôleur appris est un peu moins capable de suivre la trajectoire que le contrôleur MPC, et en particulier semble avoir des difficultés à anticiper les changements de vitesse importants (les extrema de position). Ceci est attendu puisque celui-ci ne peut anticiper ces changements, uniquement réagir à un changement de la référence en vitesse. Le contrôleur appris semble montrer une différence avec le contrôleur MPC plus importante que pour la tâche de stabilisation.

La figure 4.9 présente la distribution de l'erreur de suivi de trajectoire et confirme que le contrôleur appris est plus éloigné de la performance du contrôleur MPC sur la tâche de suivi de trajectoire que sur celle de stabilisation.

Cette transformation de l'entrée du réseau de neurones est cependant peu pertinente dans le cas d'un vol dans un tuyau. En effet celle-ci remplace l'entrée en position, qui caractérise les perturbations dues à l'environnement, par l'écart à la position de référence, rendant le contrôleur incapable de déduire les perturbations dues à l'environnement qu'il va subir. Dans un tel contexte, l'imitation d'un contrôleur ayant intégré un modèle de l'environnement peut même nuire à la performance du contrôleur puisque celui-ci va tenir compte de perturbations qui n'existent pas.

La figure 4.10 présente la distribution des erreurs de suivi pour un contrôleur appris à partir d'un problème de contrôle optimal avec et sans intégration d'un modèle de l'environnement pour des trajectoires réalisées dans un tuyau de 65cm de diamètre. On observe que le fait d'intégrer la connaissance des perturbations au modèle durant l'apprentissage réduit effectivement la capacité du contrôleur à suivre une trajectoire et augmente l'erreur de suivi que celui-ci réalise.

Cette extension du contrôleur appris au suivi de trajectoires ne permet donc pas directement l'intégration d'un modèle de l'environnement en l'état.

Pour permettre au contrôleur appris de suivre une trajectoire et d'intégrer la connaissance de perturbations sans pour autant augmenter la dimension de l'état en entrée, je propose de laisser la véritable position en entrée du réseau de neurones, mais d'ajouter un terme à l'erreur en vitesse qui soit proportionnel à la différence entre la position courante et la position de référence, de manière analogue à ce qui est fait dans le contrôleur proportionnel. J'adapte aussi le problème de contrôle optimal en modifiant le coût pour ne plus tenir compte de la position dans l'erreur et d'augmenter la contribution de la vitesse à cette erreur. Par rapport au coût décrit dans la section 2.4.3, la contribution de la position passe de 10 à 0 et celle de la vitesse de  $10^{-3}$  à 5. À l'erreur en vitesse est ajoutée  $k = 2$  fois l'erreur entre la position courante et la position de référence. On passe ainsi à une architecture hiérarchique avec un contrôleur proportionnel en position qui alimente un contrôleur en vitesse, appris par imitation.

L'équation (4.4) détaille cette transformation à partir de l'état courant  $(p, v, q, \omega)$  et celui de référence  $(p_r, v_r, q_r, \omega_r)$ . Le gain  $k$  utilisé pour transmettre l'erreur de position en une erreur de vitesse permet au contrôleur de corriger les erreurs de position tout en conservant en entrée la position véritable qui permet de déterminer les perturbations de l'environnement. La vitesse donnée en entrée est exprimée dans les coordonnées du repère du robot  $(B)$ . Ceci permet au contrôleur de ne pas distinguer entre deux consignes de vitesse identiques pour deux erreurs d'orientation différentes.

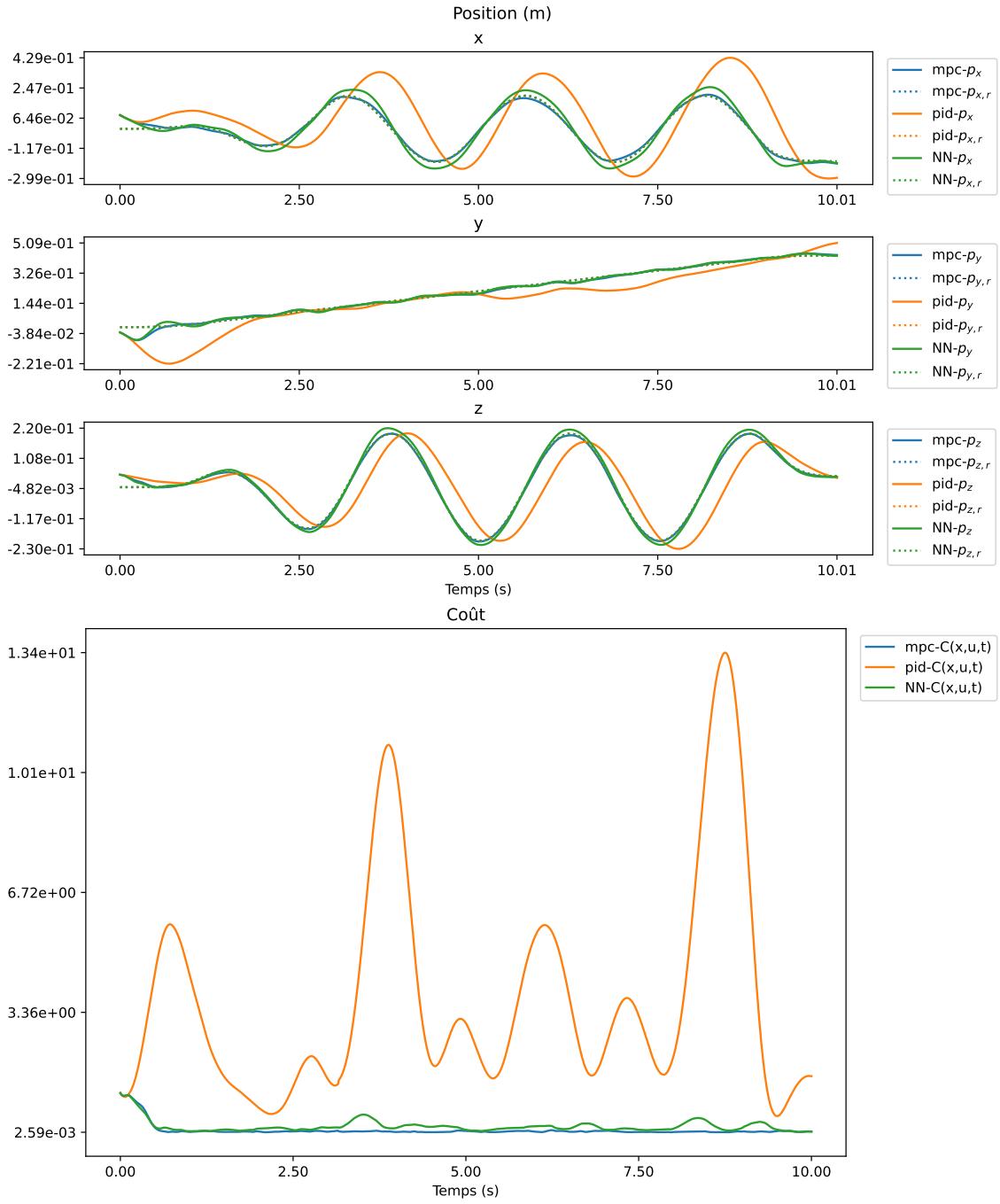


FIGURE 4.8 – Suivi de trajectoire réalisé par le contrôleur MPC (bleu), proportionnel (PID, orange) et appris (NN, vert). L'état de référence est indiqué par un trait pointillé et l'indice  $r$ . Quand cet état de référence n'est pas visible c'est qu'il est couvert par la courbe de la trajectoire et que ceux-ci coïncident. La position initiale, la trajectoire de référence et les perturbations extérieures sont les mêmes pour les trois contrôleurs. Les trois figures du haut montrent la position du robot sur chacun des trois axes et la figure du bas montre l'évolution du coût optimisé dans le problème de contrôle optimal, c'est-à-dire la distance à l'état de référence courant. La trajectoire de référence est la même que celle décrite dans la section 3.6. Le contrôleur MPC atteint rapidement et suit cette trajectoire de référence. Le contrôleur appris présente des petits écarts au moment du changement de direction. Le contrôleur proportionnel présente un retard par rapport à la référence et des dépassemens de l'amplitude demandée. La trajectoire du coût confirme ceci.

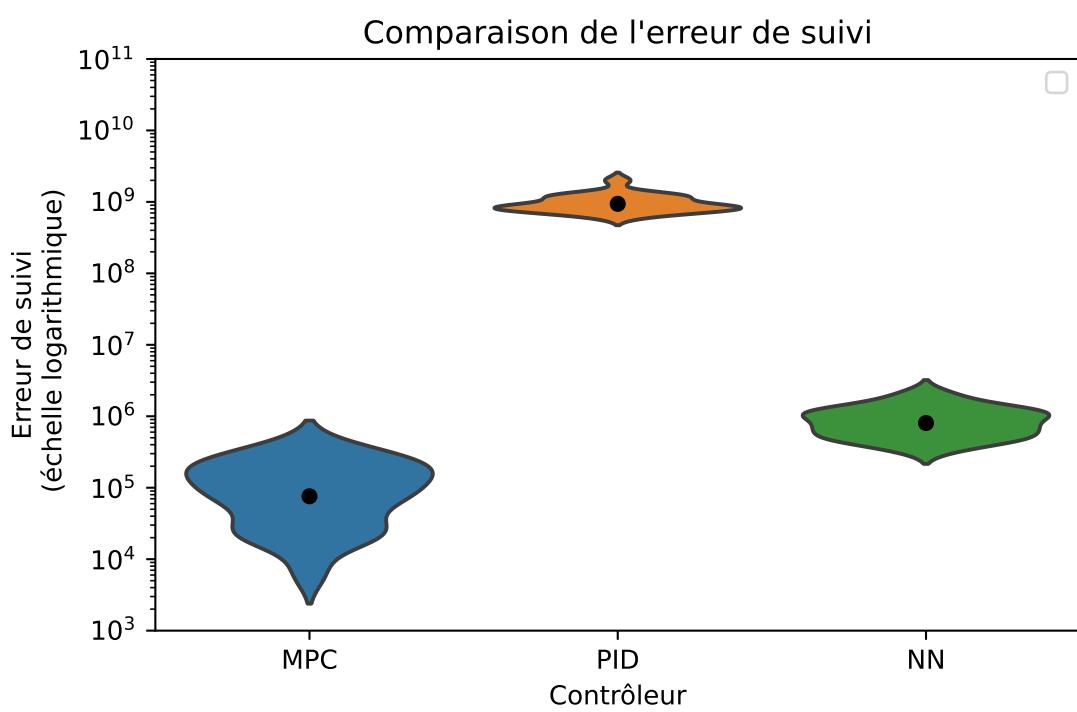


FIGURE 4.9 – Distribution de l'erreur de suivi pour le contrôleur MPC (bleu), proportionnel (PID, orange) et le contrôleur appris par imitation (NN, vert) pour 20 trajectoires. La densité de la distribution à une ordonnée fixée est donnée par la distance jusqu'où s'étend la zone colorée. Plus celle-ci s'éloigne du centre plus la distribution présente est dense. Le cercle noir au cœur de la distribution signale la valeur moyenne de celle-ci. L'axe des ordonnées représente le coût d'une trajectoire qui est sans unité.

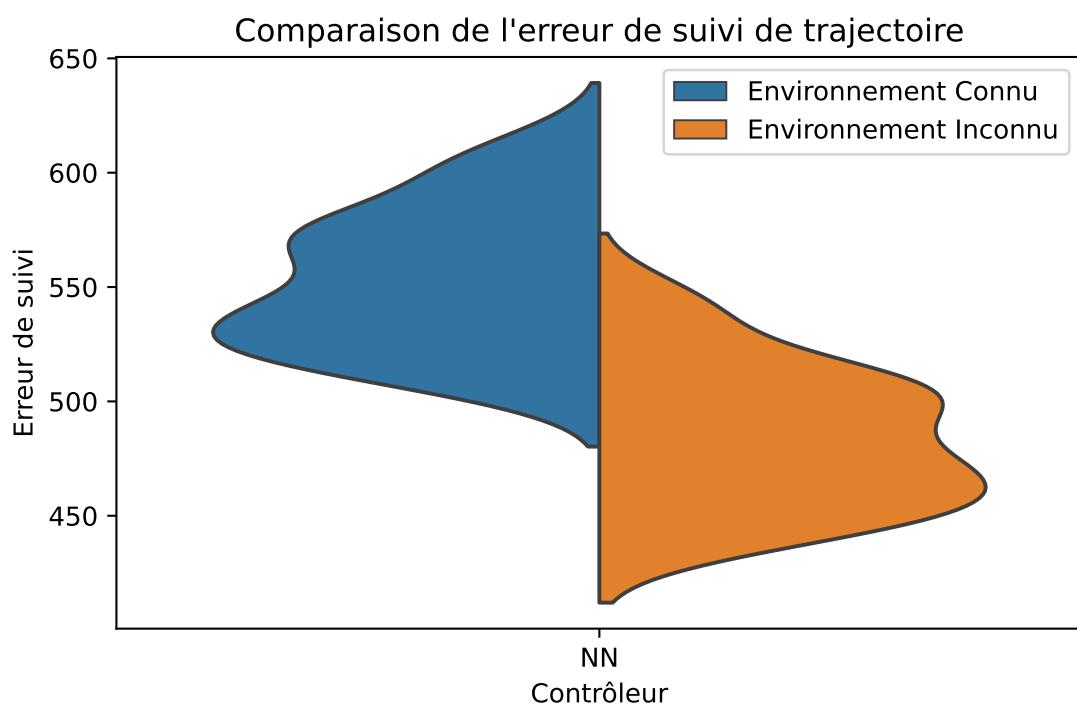


FIGURE 4.10 – Distribution de l'erreur de suivi pour un contrôleur appris avec connaissance d'un modèle de l'environnement et sans, réalisée pour 20 trajectoires dans un tuyau de 65cm de diamètre. La densité de la distribution à une ordonnée fixée est donnée par la distance jusqu'où s'étend la zone colorée. Plus celle-ci s'éloigne du centre plus la distribution présente est dense. L'axe des ordonnées représente le coût d'une trajectoire qui est sans unité.

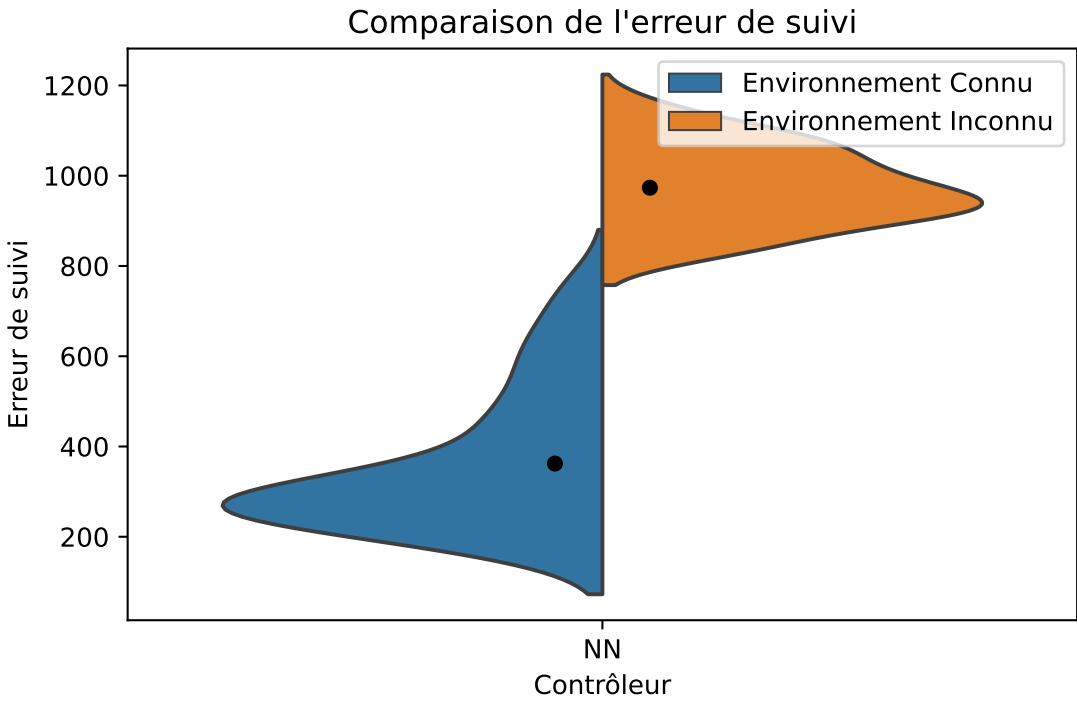


FIGURE 4.11 – Distribution de l'erreur de suivi pour un contrôleur appris avec connaissance d'un modèle de l'environnement et sans, réalisée pour 20 trajectoires dans un tuyau de 65cm de diamètre. La densité de la distribution à une ordonnée fixée est donnée par la distance jusqu'où s'étend la zone colorée. Plus celle-ci s'éloigne du centre plus la distribution présente est dense. Le contrôleur appris prend en entrée la véritable position et suit la position de référence par une modification de sa vitesse de référence. L'axe des ordonnées représente le coût d'une trajectoire qui est sans unité.

$$\begin{aligned}
 \tilde{p}(p, p_r) &= p \\
 \tilde{v}(v, v_r, p, p_r) &= [v_W - v_{W,r} + k(p_W - p_{W,r})]_B \\
 \tilde{q}(q, q_r) &= q \cdot q_r^{-1} \\
 \tilde{\omega}(\omega, \omega_r) &= \omega - \omega_r
 \end{aligned} \tag{4.4}$$

La figure 4.11 présente l'effet de cette modification sur la distribution de l'erreur. Contrairement à ce qu'on observe dans la figure 4.10, l'intégration de la connaissance des perturbations de l'environnement permet au contrôleur appris de réaliser des trajectoires dont l'erreur est bien plus faible.

La figure 4.12 présente la distribution de l'erreur pour une tâche de suivi de trajectoire avec et sans connaissance de l'environnement dans un tuyau de 65cm de diamètre. La trajectoire de référence est celle présentée dans la section 3.6 pour laquelle l'amplitude des positions de référence est réduite (15cm). Des perturbations aléatoires sont appliquées selon les modalités décrites dans la section 2.4.3.3. Le contrôleur appris peut tirer parti de cette connaissance via la transformation de son entrée et la version ayant appris sur un contrôleur ayant connaissance de l'environnement présente une distribution de l'erreur de suivi de trajectoire plus basse que celle ayant appris sans cette connaissance. On peut aussi observer que la différence entre les distributions avec et sans

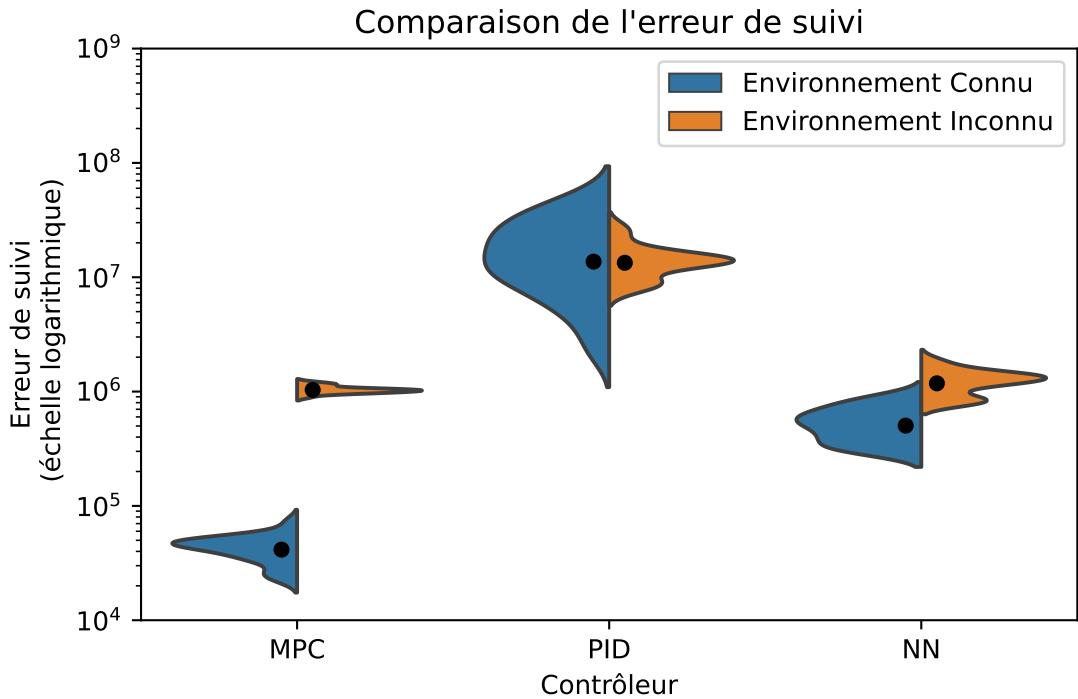


FIGURE 4.12 – Distribution de l'erreur de suivi pour un contrôleur MPC, un contrôleur proportionnel et un contrôleur appris avec et sans connaissance d'un modèle de l'environnement réalisée pour 20 trajectoires dans un tuyau de 65cm de diamètre. La densité de la distribution à une ordonnée fixée est donnée par la distance jusqu'où s'étend la zone colorée. Plus celle-ci s'éloigne du centre plus la distribution présente est dense. Le cercle noir au cœur de la distribution signale la valeur moyenne de celle-ci. Le contrôleur appris prend en entrée la véritable position et suit la position de référence par une modification de sa vitesse de référence. L'axe des ordonnées représente le coût d'une trajectoire qui est sans unité.

connaissance est plus importante pour la tâche de suivi que pour la tâche de stabilisation. Ceci découle directement du fait que, pour le suivi de trajectoires, le robot doit se déplacer à travers des zones qui présentent des perturbations différentes les unes des autres. De plus, la trajectoire suivie passe plus proche des parois que la tâche de stabilisation, ce qui impose aux contrôleurs de gérer des perturbations d'intensité plus importante. Le contrôleur proportionnel semble ne pas profiter particulièrement de l'intégration du modèle des perturbations. La distribution des coûts est bien plus étendue pour les contrôleurs proportionnels avec intégration du modèle que sans, contrairement à ce qui peut être observé sur la figure 3.17 où le contrôleur proportionnel semblait profiter de l'intégration du modèle de l'environnement. À la différence des trajectoires présentées dans la figure 3.17, des perturbations aléatoires affectent le robot pour les trajectoires de la figure 4.12 et la trajectoire de référence passe plus loin de parois, ce qui pourrait expliquer les différences de comportement du contrôleur réseau de neurones.

Une autre approche pourrait être d'intégrer directement à l'état les mesures des capteurs de distance décrits dans la section 5.3, ou bien d'intégrer directement la prédiction de la perturbation courante due à l'environnement obtenue grâce au modèle de celui-ci. Ces deux approches ont l'inconvénient d'augmenter la dimension de l'entrée du contrôleur et donc de nécessiter une aug-

mentation de l'ensemble d'apprentissage afin de couvrir l'espace d'état de manière satisfaisante. La seconde permettrait de découpler le modèle des perturbations et l'apprentissage du contrôleur au prix d'une reformulation du problème de contrôle optimal. Elle permettrait ainsi de remplacer le modèle des perturbations par n'importe quel autre modèle, voire par un estimateur en temps réel, sans devoir apprendre à nouveau le contrôleur.

## 4.8 Conclusion

Nous avons donc montré que l'utilisation d'une adaptation de l'algorithme DAGGER permet de conduire l'apprentissage supervisé d'un contrôleur. Ceci permet de tirer parti de la formulation et de la résolution hors ligne du problème de contrôle optimal pour obtenir une approximation d'un contrôleur type MPC sous la forme d'un contrôleur réactif avec une architecture de réseau de neurones. Ce contrôleur a un coût en calcul bien plus faible (1ms pour calculer une commande à bord du Crazyflie) et peut être exécuté en temps réel sur le Crazyflie.

L'évaluation en simulation de ce contrôleur appris indique que celui-ci est capable de stabiliser avec succès le Crazyflie même si cette stabilisation est moins rapide qu'avec le contrôle optimal. L'apprentissage supervisé d'une commande optimale intégrant des informations sur les perturbations présentes dans l'environnement permet une stabilisation un peu meilleure du robot qu'en ne tenant pas compte de l'environnement.

La substitution de l'état du robot par sa différence avec un état de référence permet d'utiliser ce même contrôleur appris pour réaliser du suivi de trajectoire. Si cette première transformation ne permet plus d'intégrer un modèle des perturbations dans un environnement de type tuyau, une transformation supplémentaire permet d'utiliser le contrôleur appris en tenant compte du modèle de perturbations de l'environnement pour faire du suivi de trajectoire. Le contrôleur appris imite cependant de manière plus fidèle le contrôleur MPC pour les tâches de stabilisation que pour les tâches de suivi.



# Chapitre 5

## Estimation d'état d'un quadrotor volant à l'intérieur d'un tuyau

### 5.1 Introduction

Les contrôleurs présentés dans les sections 2.4.3.1, 2.4.3.2 et 4.4 sont des contrôleurs réactifs, pour calculer la commande ceux-ci ont besoin de connaître l'état du robot qui comprend sa position, sa vitesse, son orientation et sa vitesse angulaire. En simulation, il est aisément de déterminer l'état du robot pour le fournir au robot, mais pour faire tourner de tels contrôleurs à bord du robot il est impossible de mesurer directement cet état et il est nécessaire de l'estimer à l'aide de capteurs embarqués ou non.

Les risques de collision avec les parois dans un environnement confiné comme celui des tuyaux présentés précédemment renforcent encore l'importance qu'il y a à savoir estimer correctement l'état du robot puisque celui-ci doit se placer précisément pour éviter ces collisions. Mais les spécificités de cet environnement peuvent aussi être utilisées pour mieux estimer l'état du robot. La figure 5.1 illustre la situation dans laquelle est le Crazyflie et le rapport entre les dimensions du robot et celles du tuyau.

### Objectifs

L'objectif de ce chapitre est de tirer parti des spécificités géométriques de l'environnement pour estimer l'état du Crazyflie durant son vol. En particulier nous proposons des approches qui ont pour objectif d'utiliser des mesures de télémètres afin d'obtenir une information sur l'état du robot qui ne soit pas sujette à une dérive comme les sont les estimations à l'aide de mesures inertielles.

### Plan

Dans cette section je présente 3 approches pour l'estimation d'état à l'aide de mesures de télémètres. La première approche estime l'orientation du robot et sa distance aux parois par régression linéaire sous hypothèse d'une inclinaison horizontale. La seconde approche estime la position latérale et verticale du robot sous l'hypothèse que l'orientation du robot est connue. La dernière approche formule l'expression d'un filtre de Kalman étendu intégrant un modèle d'observation des télémètres laser dans un tuyau.

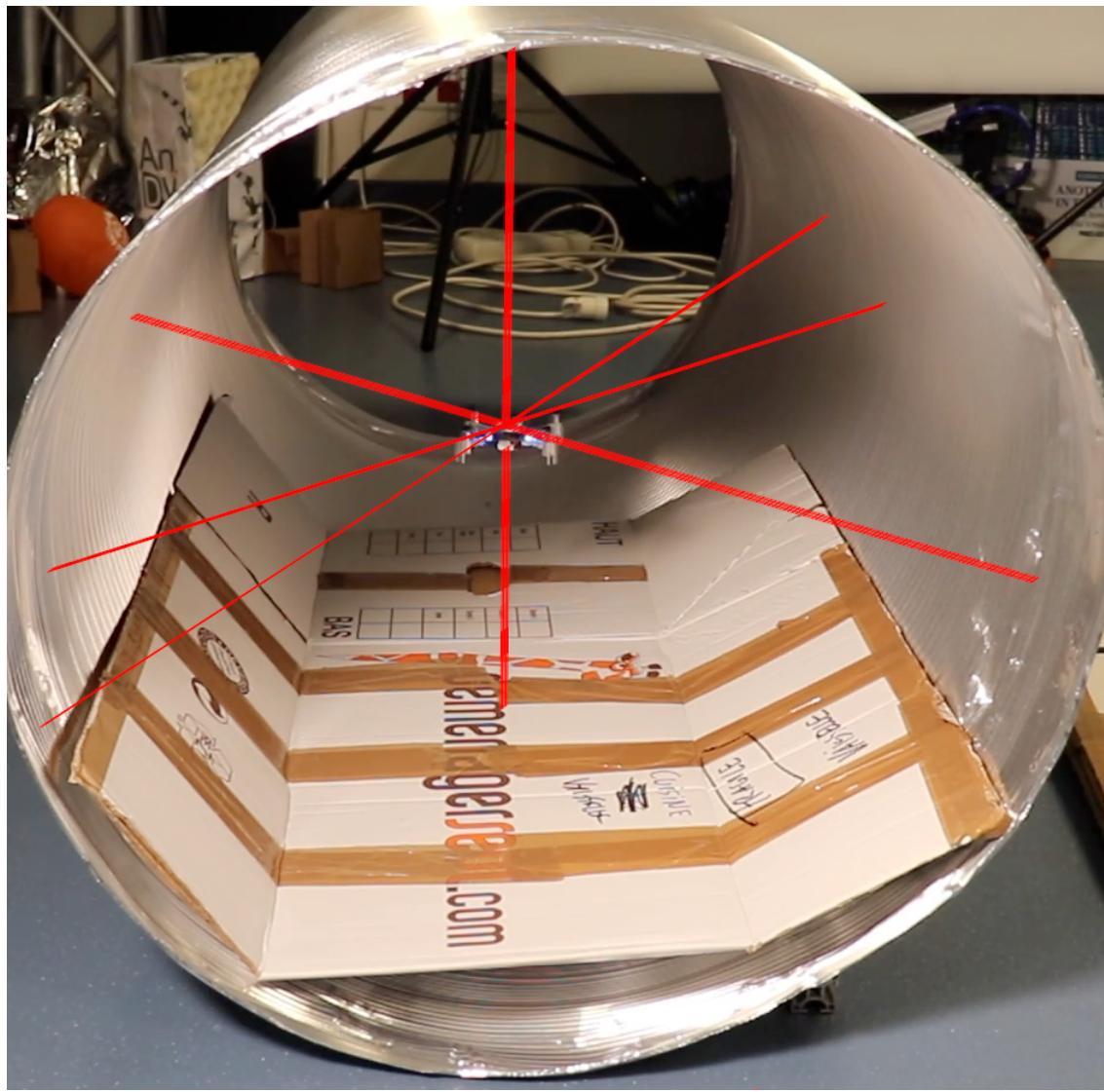


FIGURE 5.1 – Un Crazyflie dans un tuyau de 65cm de diamètre. La taille du robot par rapport aux dimensions de l'environnement rend l'estimation précise de la position cruciale. Les lignes rouges figurent des mesures de télémètres laser qui participent à l'estimation de la position du quadrotor dans le tuyau.

## 5.2 Travaux connexes

### Filtre de Kalman

Pour l'estimation de l'état de systèmes dynamiques, l'algorithme de référence est le filtre de Kalman. L'ouvrage de SIMON 2006 détaille le fonctionnement, les propriétés et les nombreuses variantes de ce filtre. La formulation originale du filtre de Kalman permet l'estimation de l'état de systèmes linéaires, mais il existe des variantes pour les systèmes non linéaires (EKF, UKF, filtres particulaires). GREIFF 2017 présente plusieurs approches pour l'estimation d'état d'un quadrotor comme le Crazyflie intégrant les mesures de capteurs embarqués (centrale inertuelle, capteur de flux optique, capteurs laser) et externes (système de capture de mouvements, ancre radio). MADGWICK, HARRISON et VAIDYANATHAN 2011 proposent un filtre ne réalisant que l'estimation de l'orientation à partir des données d'une centrale inertuelle, mais pour un coût en calculs réduit. Le firmware du Crazyflie implémente ce filtre.

### Extension des filtres de Kalman aux espaces non euclidiens

Le filtre de Kalman, est conçu pour estimer l'état de systèmes linéaires. Pour cette raison, l'estimation de grandeurs qui ne sont pas vectorielles (et donc pas linéaires) comme l'orientation pose problème. Ce problème n'est pas résolu par l'utilisation de variantes non linéaires du filtre de Kalman et nécessite une reformulation des représentations utilisées pour ces grandeurs, ou des versions du filtre de Kalman adaptées spécifiquement pour la gestion des grandeurs non euclidiennes. SOLÀ, DERAY et ATCHUTHAN 2021 présentent les différentes propriétés géométriques de l'espace des rotations dans ses différentes représentations. Ils présentent en particulier les formules de passage d'une représentation à l'autre, les systèmes de coordonnées locales, les espaces tangents et des méthodes pour intégrer et calculer des moyennes et des variances pour ces rotations. BERNAL-POLO et MARTÍNEZ-BARBERÁ 2019 comparent les différentes représentations des rotations ainsi que leurs mérites dans un contexte d'estimation par les extensions non linéaires d'un filtre de Kalman. MÜLLER 2016 détaillent les méthodes et usages pour l'estimation d'états contenant une orientation par un filtre de Kalman pour des quadrotors. GILL, MUELLER et D'ANDREA 2020 montrent que l'utilisation de coordonnées locales pour l'estimation de l'orientation, et le changement du point de référence local de ces coordonnées nécessite une adaptation de la covariance estimée par un filtre de Kalman. BROSSARD, BARRAU et BONNABEL 2020 décrivent les étapes nécessaires pour implémenter un filtre de Kalman sans parfum (UKF) pour estimer un état qui évolue dans une variété (qui sont des espaces dont la géométrie est courbe et donc non euclidienne, l'espace des rotations est par exemple une variété). Cette variante, contrairement au filtre de Kalman étendu (EKF), ne nécessite pas de calculer la dérivée des modèles de la dynamique et de l'observation. Ces dérivées peuvent être complexes à calculer pour les transformations d'états qui évoluent dans des variétés plutôt que des espaces vectoriels. D'autres approches d'estimation d'état utilisent les outils de la géométrie non euclidienne pour réaliser des filtres respectent cette géométrie. MAHONY et al. 2012 réalisent par exemple un filtre complémentaire sur le groupe  $SL(3)$  pour l'estimation de transformation d'images d'une scène. Il s'agit d'un autre exemple d'un espace dont la géométrie est non euclidienne et qui permet d'estimer la position d'un robot par rapport à un objet fixe à partir d'observations issues d'une caméra.

## Conclusion

De nombreux travaux ont traité de l'estimation d'état dynamique des quadrotors. L'outil le plus courant pour cela, le filtre de Kalman, ne permet cependant pas de réaliser cette approximation sans procéder à des approximations ou des adaptations spécifiques à la géométrie de l'espace des états. À notre connaissance aucune approche n'intègre d'hypothèses sur la géométrie d'un environnement cylindrique pour faciliter l'estimation de l'état à l'aide de télémètres. Nous proposons deux approches pour l'estimation instantanée de l'état et une formulation d'un filtre de Kalman étendu (EKF) qui intègrent les connaissances sur la géométrie de l'environnement et tirent parti de mesures de télémètres laser pour se localiser.

## 5.3 Capteurs

Le Crazyflie embarque plusieurs capteurs ainsi qu'un estimateur d'état (K. McGuire 2021). Pour estimer cet état, il embarque les capteurs suivants :

- un accéléromètre ;
- un gyromètre ;
- un capteur de flux optique estimant la vitesse relative au sol ;
- un capteur laser mesurant la distance au sol.

L'accéléromètre mesure la force spécifique qui s'exerce sur le robot, c'est-à-dire la résultante des forces extérieures, gravité exclue, divisée par la masse (MARTIN et SALAÜN 2010 ; GROVES 2013) ce qui correspond à l'accélération du robot exprimée dans le système de coordonnées du robot :  $[\ddot{\mathbf{p}} - \mathbf{g}]_{C_B}$ . Ceci ne signifie pas que l'accéléromètre est incapable d'obtenir des informations sur l'intensité de la gravité. En particulier, si le robot est immobile, un accéléromètre va mesurer une accélération de  $[-\mathbf{g}]_{C_B}$ . Un robot immobile dans le référentiel inertiel mesurera donc une accélération non nulle : le robot est immobile donc la résultante des forces (de réaction ou issue des rotors) compense exactement la gravité et l'accéléromètre mesure cette résultante. L'accéléromètre peut ainsi permettre d'obtenir des informations sur la direction de la gravité et donc sur l'orientation du robot dans le référentiel inertiel.

Le gyromètre permet de mesurer la vitesse angulaire du robot, exprimée dans le système de coordonnées du robot :  $[\omega_{WB}]_{C_B}$ . L'estimation d'état à l'aide de mesures inertielles (accéléromètre, gyromètre) est sujette à dérive à cause de l'accumulation des erreurs lors de l'intégration de ces mesures. Cette dérive rend l'estimation de la position peu fiable au bout d'un temps assez court. La proximité des parois et la variabilité des perturbations selon la position rendent cruciale l'estimation correcte de la position, ce que ne permet pas l'estimateur d'état déjà implémenté à bord du Crazyflie.

Afin de tirer parti de l'environnement qu'est le tunnel, nous avons choisi d'utiliser un ensemble de 9 télémètres laser supplémentaires embarqués sur le Crazyflie à l'aide d'un "deck" réalisé par Lucien Renaud, ingénieur dans l'équipe. 8 de ces 9 capteurs sont placés régulièrement dans le plan horizontal du robot avec un écart de 45° entre chaque capteur. Ils mesurent la distance entre le robot et les parois de l'environnement. Le 9e capteur est placé à la verticale vers le haut et mesure la distance entre le robot et le haut du tuyau. Le Crazyflie intégré aussi un 10e télémètre dirigé vers le bas qui lui permet d'estimer son altitude.

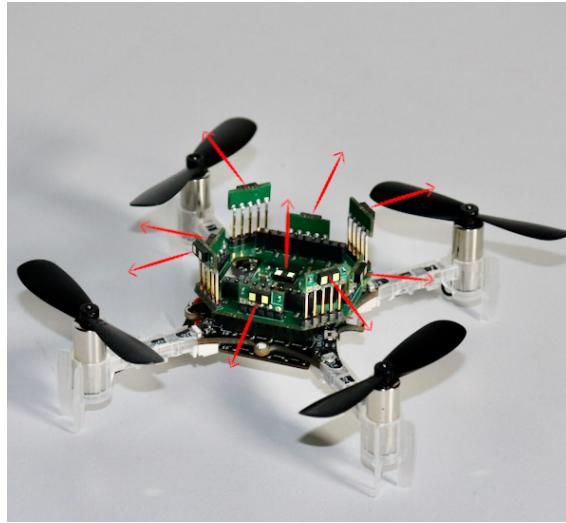


FIGURE 5.2 – Un Crazyflie sur lequel est monté une carte contenant 9 capteurs laser de distance. La direction de mesure de ces capteurs est indiquée par les flèches rouges. Ces capteurs sont orientés régulièrement et présentent un angle de  $45^\circ$  entre deux capteurs successifs.

La figure 5.2 résume le placement de ces 9 capteurs supplémentaires, alignés sur les axes horizontaux du système de coordonnées du robot.

Les mesures de ces télémètres et des capteurs de la centrale inertIELLE sont modélisées comme la somme de la distance mesurée et d'un bruit gaussien. Les niveaux de bruit pour l'accéléromètre et le gyromètre sont obtenus à partir de la variance de mesures réalisée sur un Crazyflie maintenu immobile, ses moteurs fonctionnant à un niveau permettant de compenser la gravité (70% de la pleine puissance).

La figure 5.3 présente la distribution des mesures pour la vitesse angulaire et la figure 5.4 présente celle des mesures d'accélération. On peut observer que l'hypothèse d'un bruit de mesure gaussien s'applique mal pour les mesures d'accélération. Il est possible d'intégrer des biais aux modèles de fonctionnement de ces capteurs et d'estimer la valeur de ces biais comme les autres grandeurs qui constituent l'état (GROVES 2013).

Les télémètres laser présentent un bruit de mesure qui varie avec la luminosité et la surface sur laquelle ils se réfléchissent. Ce niveau de bruit est de l'ordre du millimètre.

## 5.4 Approche par régression linéaire

Une première approche pour affiner l'estimation de la position consiste à utiliser les mesures de distance de télémètres placés dans le plan horizontal pour estimer la direction générale du tuyau ainsi que la distance à son axe central. Cette direction est déterminée à l'aide d'une régression linéaire qui estime l'équation de la droite passant par les points où les lasers des télémètres rencontrent la paroi de l'environnement. Cette approche a été formalisée sur une idée proposée par Lucien Renaud.

Elle prend comme hypothèse un environnement de type couloir, c'est-à-dire deux parois verticales parallèles. Un environnement constitué d'un tuyau ne vérifie bien entendu pas cette hypothèse dans le cas général, mais dans le cas particulier où le robot est presque à l'horizontale l'approche pourra convenir. Dans cet environnement supposé constitué de deux parois verticales,

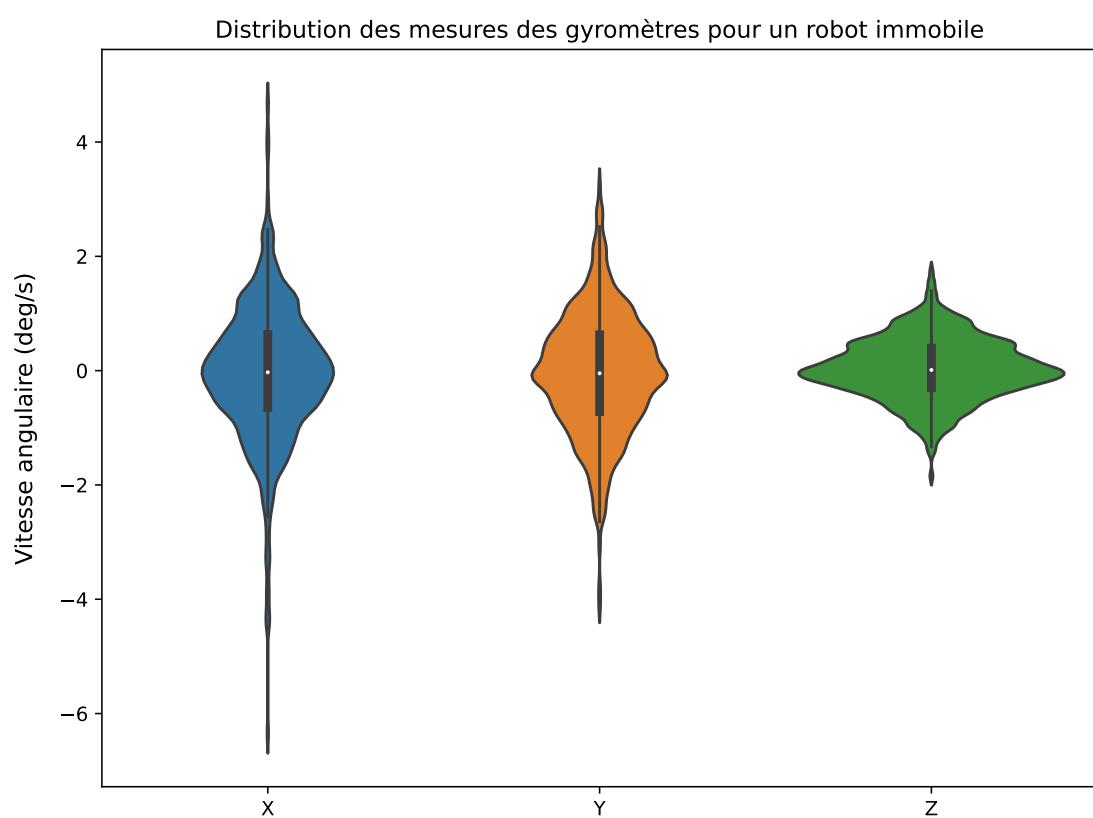


FIGURE 5.3 – Distribution des mesures des gyromètres sur les trois axes pour un robot immobile. La distribution des mesures ressemble à une gaussienne de moyenne nulle, et de variance de l'ordre de 1 deg/sec. Les données correspondent à 10 secondes de vol.

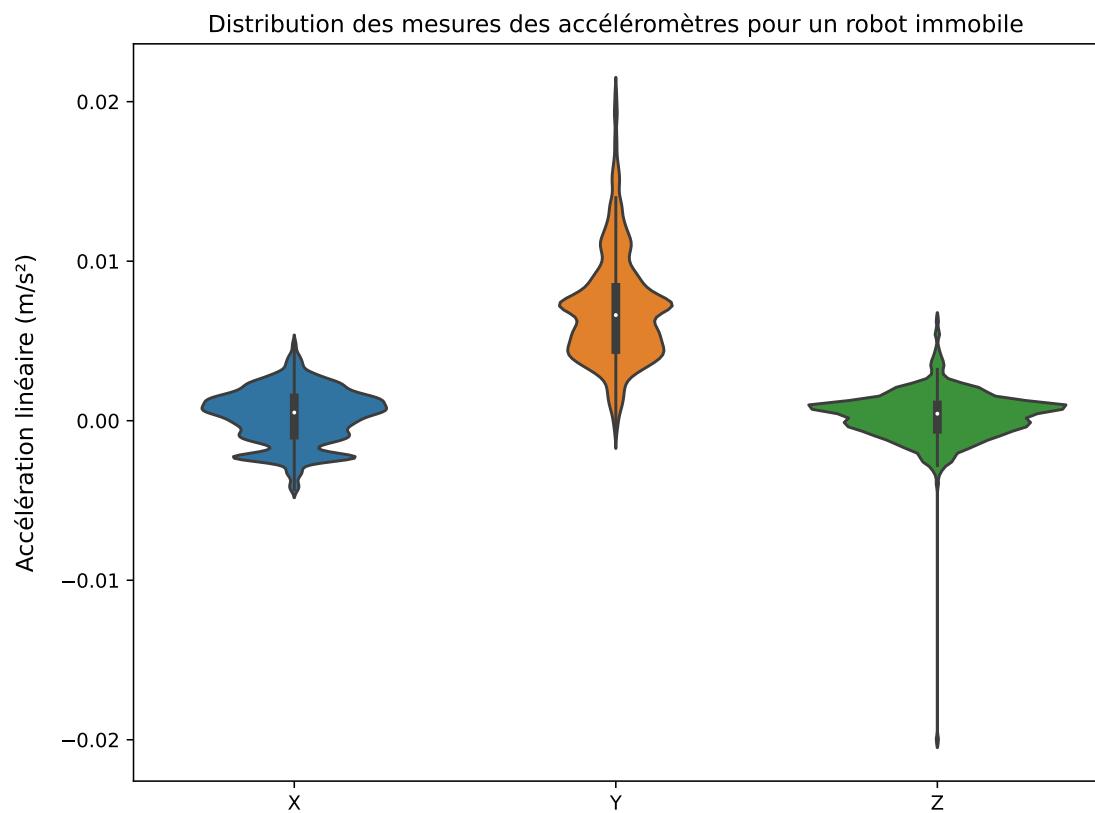


FIGURE 5.4 – Distribution des mesures des accéléromètres sur les trois axes pour un robot immobile. La distribution n'a pas une moyenne nulle pour l'accélération selon y et n'a pas la forme d'une gaussienne pour aucun des axes. L'accélération de gravité a été soustraite aux mesures sur l'axe z. Les données correspondent à 10 secondes de vol.

nous utilisons les mesures de distance pour estimer la distance du robot à ces deux parois ainsi que l'orientation du robot par rapport à ces deux parois (c'est-à-dire le lacet qui correspond à  $\theta$  sur la figure 5.5).

Pour estimer cette distance et cette orientation, on estime l'équation des deux droites (P1 et P2 sur la figure 5.5) qui passent par les points où les lasers des capteurs de distance atteignent les parois. L'expression de ces équations de droite dans le système de coordonnées du robot permet d'obtenir une estimation de l'angle entre la direction des parois et la direction "avant" dans le système de coordonnées du robot via la pente des droites, ainsi que la distance à ces parois via l'ordonnée à l'origine des droites. La figure 5.5 illustre la méthode utilisée, les grandeurs mesurées et les grandeurs estimées.

Selon l'orientation du robot, certains capteurs de distance peuvent avoir une direction parallèle aux parois, ceci a pour conséquence que la mesure réalisée par ces capteurs ne peut correspondre à la distance réelle le long de cette direction de mesure puisque la distance réelle est très importante (infinie dans le cas d'un alignement exact) et sature donc les capteurs.

Pour éviter d'utiliser des mesures qui correspondent à de telles situations, la paire de capteurs opposés mesurant la plus grande valeur n'est pas utilisée pour l'estimation des équations de droite. Cette estimation est donc réalisée à l'aide de 3 capteurs pour chaque paroi. Par ailleurs, afin d'éviter les instabilités numériques dues à l'estimation de droites dont la pente est très élevée, le système de coordonnées utilisé pour estimer les équations de droite subit une rotation quand la pente estimée est trop importante. L'algorithme 6 décrit le détail de l'approche. Cet algorithme ne permet donc d'estimer que la position latérale du robot et le lacet.

Pour un robot dont l'orientation est proche de l'horizontale, la manière dont est perçue un tuyau par les télémètres dans le plan horizontal est très proche d'une situation pour laquelle il y a deux parois verticales.

Vu la géométrie de l'environnement, il est impossible de distinguer les deux sens dans la direction du tuyau (i.e. l'avant de l'arrière). En conséquence, le lacet ne peut être estimé qu'à 180° près, et quand le robot fait un demi tour, la position le long de l'axe x ne peut plus être estimée : quand il est orienté vers l'arrière du tuyau, il estime l'opposé de la position latérale.

La figure 5.6 présente une trajectoire en simulation dans un tuyau de 30cm de rayon ainsi que l'estimation de la position latérale et du lacet par cet algorithme. Chaque télémètre subit un bruit gaussien de variance 1cm, et des prises de roulis et de tangage jusqu'à 1.5°. La position latérale et le lacet sont estimés avec une erreur respectivement de l'ordre d'1cm et de quelques degrés. L'estimation présente ponctuellement des écarts importants (en particulier pour le lacet). Ces estimations sont réalisées de manière instantanée, c'est-à-dire qu'elles ne nécessitent aucun historique des estimations ou des mesures passées. Il serait bien entendu possible de coupler ces estimations à un filtre ou un algorithme de régularisation afin de prévenir ces écarts ponctuels.

Une approche similaire à celle utilisée pour la position latérale pourrait être utilisée pour estimer l'altitude du robot par rapport à l'axe du tuyau. Une approche intégrant l'estimation simultanée de l'altitude et de la position latérale est décrite dans la section suivante.

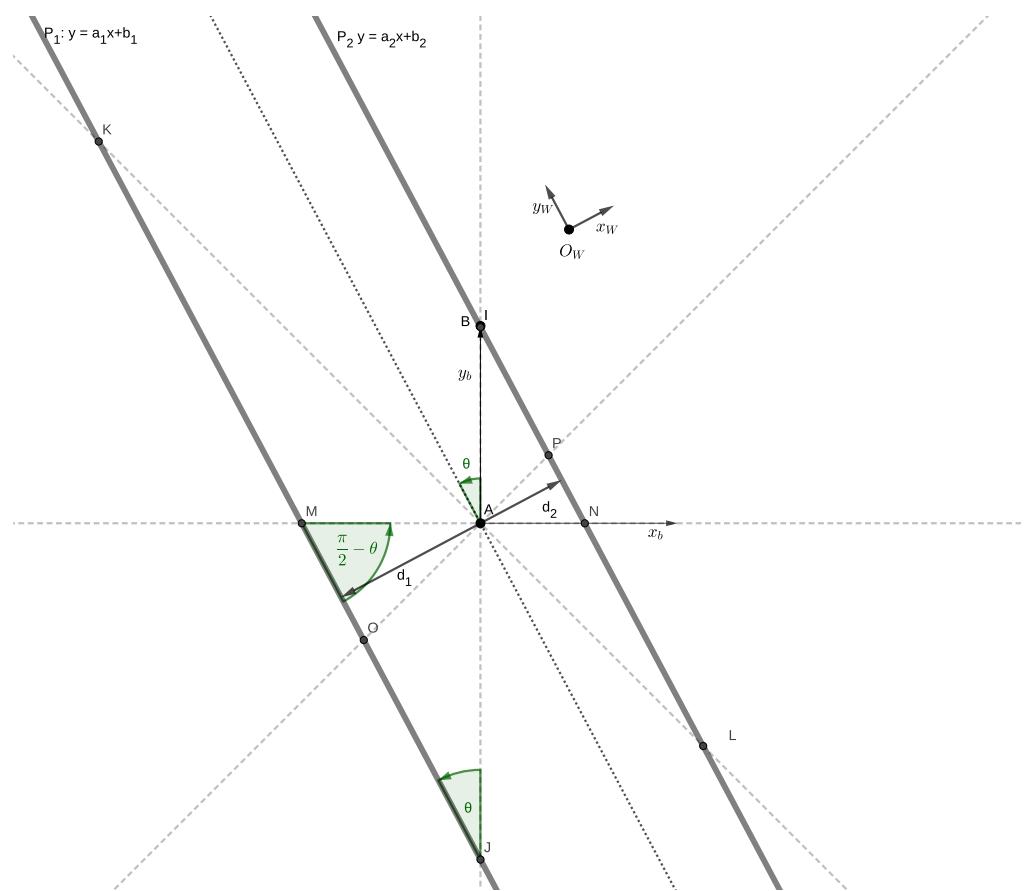


FIGURE 5.5 – Représentation d'un environnement avec deux parois parallèles dans le système de coordonnées du robot. Les points I, J, K, L, M, N, P sont les points où les lasers des capteurs atteignent la paroi. Les lignes grisées en pointillés représentent les directions de mesure de ces capteurs. Les coefficients des droites \$P\_1\$ et \$P\_2\$ permettent de calculer l'angle \$\frac{\pi}{2} - \theta\$ et donc l'angle theta ainsi que les distances \$d\_1\$ et \$d\_2\$.

---

**Algorithme 6** Algorithme d'estimation du lacet et de la distance à la paroi quand le drone est à l'horizontale

---

- 1: mesure des distances par chacun des 8 capteurs dans le plan horizontal du drone
  - 2: détermination du couple de capteurs indicé ( $i, i+4$ ) alignés sur la même droite mais de sens opposé tel que la somme des distances mesurées par ces capteurs soit la plus grande
  - 3: séparation des capteurs restants en deux groupes correspondants aux deux parois et séparés par la droite le long de laquelle le couple de capteurs de l'étape précédente
  - 4: pour chaque groupe, détermination des coordonnées des points où le laser du capteur de distance touche la paroi :  $(x_j, y_j) = d_j[s_j]_{C_B}$  avec  $d_j$  la distance mesurée et  $s_j$  la direction dans laquelle point le capteur  $j$
  - 5: pour chaque groupe, estimation de la pente de la droite  $a = \frac{3 \sum_{j=0}^2 x_j y_j - \left( \sum_{j=0}^2 x_j \sum_{j=0}^2 y_j \right)}{3 \sum_{j=0}^2 x_j^2 - \left( \sum_{j=0}^2 x_j \sum_{j=0}^2 y_j \right)}$
  - 6: **if**  $|a| > 5$  **then**
  - 7:     rotation du système de coordonnées de  $45^\circ$  et retour au point 5.
  - 8: **end if**
  - 9: calcul de l'ordonnée à l'origine  $b = \frac{1}{3} \sum_{j=0}^2 y_j - ax_j$
  - 10: calcul de l'angle  $\theta = \frac{\pi}{2} - \arctan(a)$  entre la direction des parois et la direction avant du robot
  - 11: rotation du système de coordonnées pour annuler les éventuelles rotations du point 6
  - 12: calcul de la distance à la paroi, c'est à dire la distance entre l'origine et la droite qui passe par la paroi  $d = \frac{|b|}{\sqrt{1+a^2}}$
  - 13: calcul de la position de long de l'axe  $x$ , c'est à dire l'axe horizontal perpendiculaire à la direction des parois  $x = r - d$  avec  $d$  la distance à la paroi de droite et  $r = d_1 + d_2$  la distance entre les deux parois, c'est à dire la somme de la distance à droite et à gauche
-

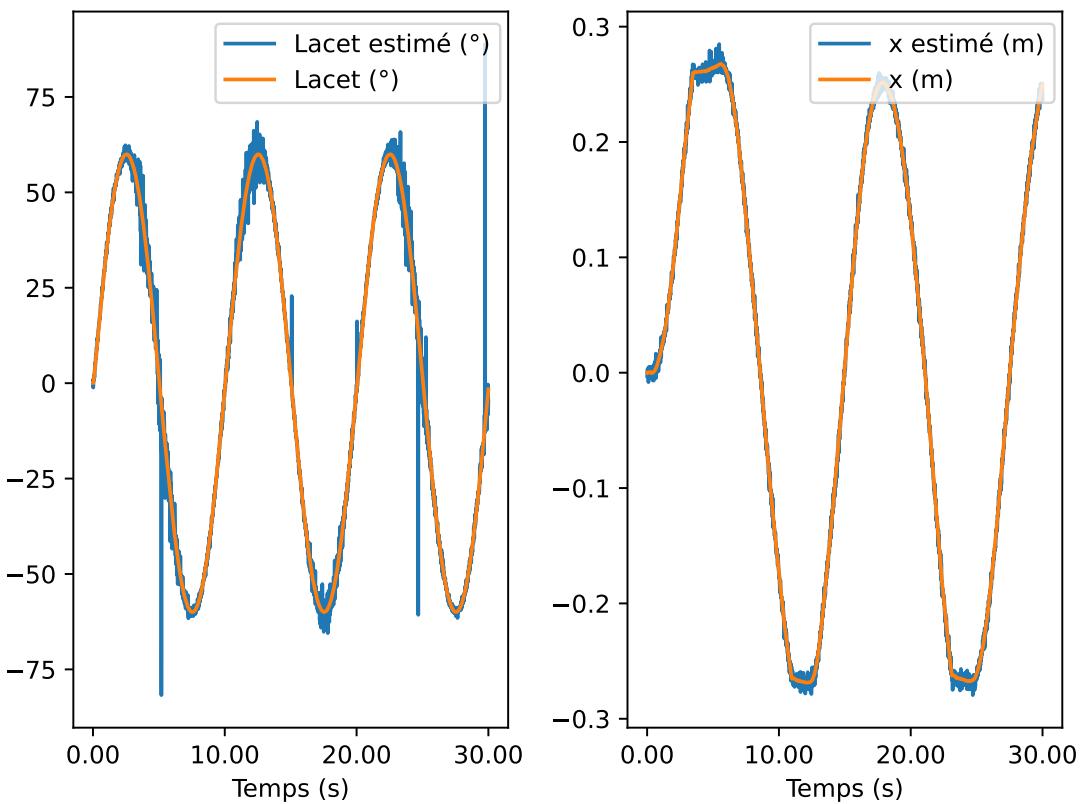


FIGURE 5.6 – Position latérale et lacet d'une trajectoire ainsi que son estimation par l'approche par l'algorithme 6. Le robot se déplace dans un tuyau de 30cm de rayon sans perturbations dues à l'environnement avec un modèle simplifié de la dynamique. Les mesures des capteurs et la simulation sont réalisées à 100Hz.

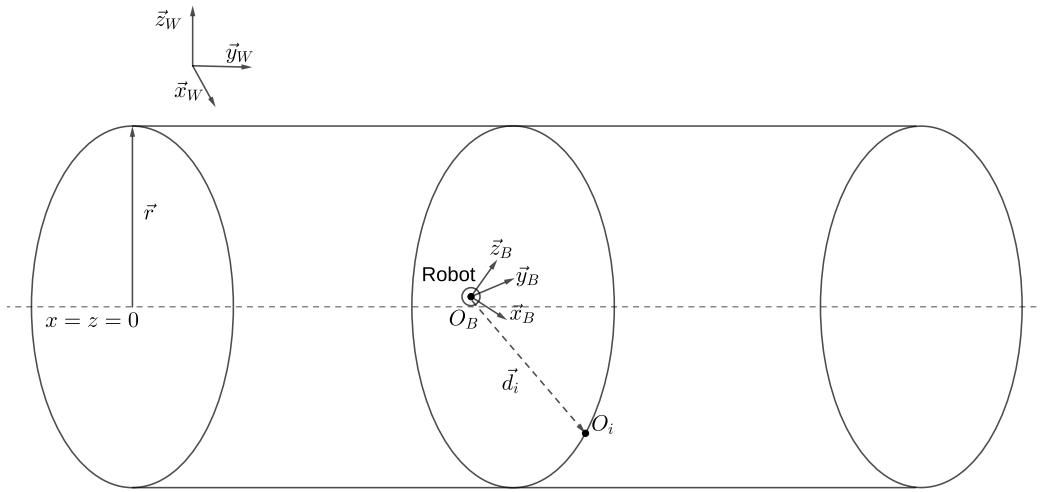


FIGURE 5.7 – Configuration géométrique de l'environnement et du robot. L'environnement consiste en un tuyau, un cylindre de rayon  $r$  et d'axe  $\mathbf{y}_W$ . Le robot est en  $O_B$ , centre de son système de coordonnées de base  $x_B, y_B, z_B$ . Le capteur de distance indicé  $i$  mesure la distance  $O_B O_i$  entre le robot et la paroi de l'environnement, c'est-à-dire la longueur du vecteur  $\mathbf{d}_i$ . Le point  $O_i$  est le point du cylindre où arrive le laser qu'utilise le capteur de distance  $i$ . Les ellipses représentent des sections transverses du tuyau.

## 5.5 Approche par modèle inverse via les mesures des télémètres

Cette seconde approche pour l'estimation de l'état du robot dans un environnement de type tuyau se base elle aussi quasi exclusivement sur les mesures des télémètres. Le principe de cette approche est de formuler les équations des distances mesurées par chacun des télémètres présents sur le robot en fonction de l'orientation du robot, de sa position dans l'environnement et du rayon du tuyau et d'inverser ce modèle des mesures pour exprimer la position du robot en fonction des mesures, de l'orientation du robot et du rayon du tuyau.

Le principal inconvénient de cette approche est qu'elle suppose que l'orientation du robot est connue. Il est possible d'utiliser l'estimation d'orientation réalisée par le contrôleur embarqué du Crazyflie et basée sur sa centrale inertuelle pour obtenir cette orientation. Cette estimation est cependant sujette à une dérive, en particulier sur le lacet et quand le robot reste immobile. En intérieur, l'estimation du lacet ne peut compter sur un capteur de type boussole comme elle le ferait en extérieur et elle repose exclusivement sur les mesures de la centrale inertuelle.

La figure 5.7 présente la géométrie du robot dans un environnement de tuyau et les notations utilisées pour cette approche.

Afin d'estimer la position, je commence par formuler un modèle direct de la distance mesurée en fonction de la position, de l'orientation et du rayon du cylindre  $r$ . Pour chaque télémètre d'indice  $i$ , le point  $O_i$ , qui est l'intersection entre l'axe de mesure du télémètre et la paroi, est sur le cylindre ses coordonnées vérifient l'équation du cylindre :

$$O_{i,x}^2 + O_{i,z}^2 = r^2 \quad (5.1)$$

Or le point  $O_i$  est obtenu par translation du point  $O_B$  selon le vecteur d'orientation du télémètre  $\mathbf{d}_i = d_i \mathbf{s}_i$ . Ses coordonnées dans le repère inertiel vérifient alors :

$$\begin{cases} O_{i,x} = O_{B,x} + d_i s_{i,x} \\ O_{i,y} = O_{B,y} + d_i s_{i,y} \\ O_{i,z} = O_{B,z} + d_i s_{i,z} \end{cases} \quad (5.2)$$

et on obtient donc

$$(O_{B,x} + d_i s_{i,x})^2 + (O_{B,z} + d_i s_{i,z})^2 = r^2 \quad (5.3)$$

On peut ensuite développer l'équation 5.3 pour obtenir l'équation du second ordre en la variable  $d_i$  suivante :

$$d_i^2(s_{i,x}^2 + s_{i,z}^2) + d_i 2(s_{i,x} O_{B,x} + s_{i,z} O_{B,z}) + O_{B,x}^2 + O_{B,z}^2 - r^2 = 0 \quad (5.4)$$

Ce polynôme de second degré a au plus deux solutions. Les conditions habituelles sur le déterminant sont ici vérifiées par des arguments géométriques : si  $O_B$  est strictement à l'intérieur du cylindre et que  $s_i$  n'est pas parallèle à l'axe du cylindre, il existe deux exactement solutions pour l'intersection de la droite portée par  $s_i$  avec la paroi du cylindre.

Une de ces solutions est positive (on rencontre le cylindre en suivant  $s_i$  dans sa direction) et une négative (même chose mais dans la direction opposée). La distance mesurée par le télémètre correspond à la solution positive et en conséquence, on peut exprimer la distance mesurée par le capteur  $i$ ,  $d_i$ , comme la plus grande des racines de l'équation 5.4

$$d_i = \frac{-(s_{i,x} O_{B,x} + s_{i,z} O_{B,z}) + \sqrt{r^2(s_{i,x}^2 + s_{i,z}^2) - (s_{i,x} O_{B,z} - s_{i,z} O_{B,x})^2}}{s_{i,x}^2 + s_{i,z}^2} \quad (5.5)$$

Dans cette expression de la distance mesurée  $d_i$ , l'orientation du robot apparaît indirectement via les coordonnées de  $s_i$  dans le repère inertiel. Les coordonnées de  $s_i$  dans le repère du robot sont connues et leur expression dans le repère inertiel dépend de l'orientation du robot :

$$[s_i]_{C_W} = R_{WB}[s_i]_{C_B} \quad (5.6)$$

L'équation 5.5 permet de modéliser les capteurs en simulation.

À partir de l'équation 5.3 associée au capteur  $i$  et celle associée à un capteur distinct  $j$  on obtient :

$$(O_{B,x} + d_i s_{i,x})^2 + (O_{B,z} + d_i s_{i,z})^2 = (O_{B,x} + d_j s_{j,x})^2 + (O_{B,z} + d_j s_{j,z})^2 = r^2 \quad (5.7)$$

On regroupe ensuite les termes pour isoler les coordonnées de la position :

$$O_{B,x}^2(1-1) + O_{B,z}^2(1-1) + O_{B,x}2(d_i s_{i,x} - d_j s_{j,x}) + O_{B,z}2(d_i s_{i,z} - d_j s_{j,z}) = d_j^2(s_{j,x}^2 + s_{j,z}^2) - d_i^2(s_{i,x}^2 + s_{i,z}^2) \quad (5.8)$$

On peut ensuite indiquer les couples de capteurs distincts et définir les coefficients suivants :

$$\begin{aligned} a_k &= a_{i,j} = & d_j^2(s_{j,x}^2 + s_{j,z}^2) - d_i^2(s_{i,x}^2 + s_{i,z}^2) \\ b_k &= b_{i,j} = & 2(d_i s_{i,x} - d_j s_{j,x}) \\ c_k &= c_{i,j} = & 2(d_i s_{i,z} - d_j s_{j,z}) \end{aligned} \quad (5.9)$$

et obtenir l'équation suivante :

$$\begin{aligned} a_k &= b_k O_{B,x} + c_k O_{B,z} \\ A &= (B|C) \begin{pmatrix} O_{B,x} \\ O_{B,z} \end{pmatrix} \end{aligned} \quad (5.10)$$

A, B et C sont des vecteurs colonnes constitués des coefficients  $a_k$ ,  $b_k$  et  $c_k$ . Si  $k > 2$  le problème est sur-constraint et on peut estimer la position comme la solution à un problème de moindres carrés décrit par l'équation 5.10. On peut obtenir la solution de ce problème à l'aide d'un pseudo inverse(QUARTERONI 2000) de la matrice  $M = (B|C)$  pour calculer

$$\begin{pmatrix} O_{B,x} \\ O_{B,z} \end{pmatrix} = (M^T M)^{-1} M^T A \quad (5.11)$$

Pour obtenir  $k > 2$  il suffit de 3 télémètres distinct, mais l'utilisation d'un plus grand nombre de mesures permet une plus grande fiabilité vis-à-vis du bruit de mesure. Par ailleurs, certains capteurs peuvent se retrouver alignés avec l'axe du tuyau. Le nombre important de ces capteurs permet d'écartier les mesures des télémètres alignés avec l'axe du tuyau. La géométrie de l'environnement ne permet pas à ce filtre d'estimer la position selon l'axe du cylindre,  $y$ .

La figure 5.8 présente l'estimation d'une trajectoire en simulation dans un tuyau de 30cm de rayon. La trajectoire présente des oscillations sinusoïdales en roulis, lacet et tangage d'amplitude 20°, avec un bruit gaussien de variance 1cm sur les mesures des télémètres. Pour améliorer la stabilité numérique des estimations, les mesures de capteurs qui dépassent 10 fois le rayon du tuyau sont ignorées. L'estimation de la position obtenue présente des erreurs de l'ordre de 1cm, ce qui est comparable à l'estimateur de position latérale précédent.

La trajectoire choisie pour illustrer cet estimateur a été conçue avec des variations de vitesse importante pour placer le robot dans un régime pour lequel les écarts d'orientation par rapport à l'horizontale sont importants. Sur cette trajectoire, l'estimateur de la section 5.4 présente des erreurs sur la position latérale de l'ordre de 5-7cm et des erreurs de lacet de l'ordre de plusieurs dizaines de degrés.

Cette approche utilisant un modèle inverse de l'équation de mesure et la donnée de l'orientation du robot permet d'estimer la position du robot dans des situations plus variées que l'estimateur de la section 5.4. Cette approche tire parti d'une estimation de l'orientation du robot, réalisée par un estimateur d'état intégré au contrôleur embarqué grâce aux mesures de la centrale inertuelle.

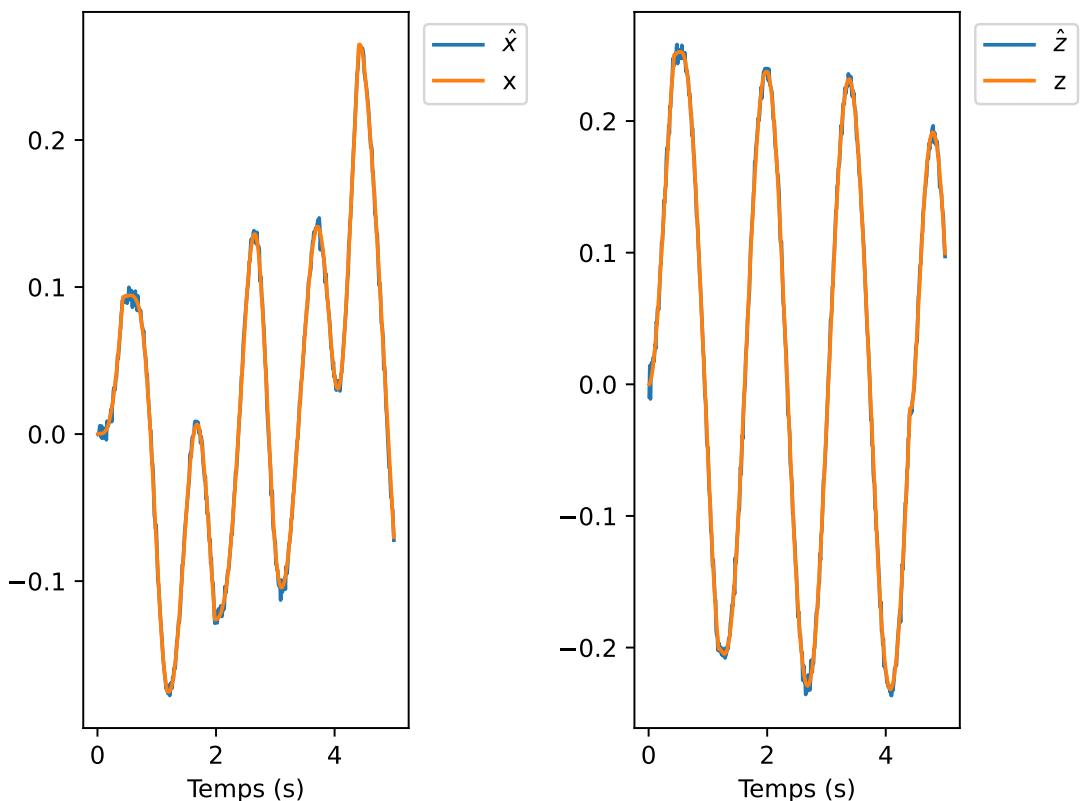


FIGURE 5.8 – Estimation de la position sur les axes verticaux et latéraux dans un tuyau de 30cm de rayon en simulation par l'approche basée sur le modèle inverse de l'équation de mesure des capteurs de distance. La position estimée (en bleu) présente une erreur de l'ordre de 1cm par rapport à la position réelle (en orange). Les mesures des capteurs et la simulation sont réalisées à 100Hz.

## 5.6 Approche par filtre de Kalman

Pour tenter de réaliser une estimation de la position n'utilisant que les mesures réalisables à bord du Crazyflie, je présente une 3e approche utilisant un filtre de Kalman étendu, un modèle de la dynamique du robot et les mesures de la centrale inertuelle en supplément des mesures des télémètres.

L'utilisation d'un filtre de Kalman et d'un modèle dynamique ont l'avantage de permettre l'utilisation d'un historique de mesures, par opposition à l'estimation instantanée réalisée dans les deux approches précédentes qui n'utilisaient que les mesures de l'instant courant pour réaliser l'estimation de l'état courant. Cette approche fait donc l'hypothèse qu'un modèle de la dynamique du robot est connu, que des mesures de distance, d'accélérations linéaires et de vitesses angulaires sont réalisées et estime la position et l'orientation du robot dans un environnement cylindrique de rayon connu.

Pour ne réaliser l'estimation qu'à partir des mesures de la centrale inertuelle et des capteurs de distance, il est nécessaire d'estimer les vitesses linéaire et angulaire ainsi que l'accélération linéaire en plus de l'orientation et de la position. L'état estimé comprendra donc les grandeurs suivantes :  $x = ([O_B]_{C_W}, [\mathbf{v}]_{C_W}, [\mathbf{a}]_{C_W}, [R_B^W]_{C_W}, [\boldsymbol{\omega}_{WB}]_{C_B})$  respectivement les positions, vitesses et accélérations linéaires puis les positions et vitesses angulaires. Les équations de la dynamique sont adaptées des équations de la section 2.3.3 :

$$f(x) = f([O_B]_{C_W}, \mathbf{v}_{C_W}, \mathbf{a}_{C_W}, [R_B^W]_{C_W}, [\boldsymbol{\omega}_{WB}]_{C_B}) = \begin{pmatrix} [O_B]_{C_W} + [\mathbf{v}]_{C_W} dt \\ [\mathbf{v}]_{C_W} + [\mathbf{a}]_{C_W} dt \\ [\mathbf{a}]_{C_W} \\ [R_B^W]_{C_W} (I_3 + [[\boldsymbol{\omega}_{WB}]_{C_B} dt] \wedge) \\ [\boldsymbol{\omega}_{WB}]_{C_B} \end{pmatrix} \quad (5.12)$$

Ce modèle de la dynamique suppose que l'accélération linéaire et la vitesse angulaire restent constantes. L'estimation de ces grandeurs n'est mise à jour que si des observations constatent une évolution de celles-ci.

Les mesures sont celles des capteurs de distance et de la centrale inertuelle, c'est-à-dire les distances  $d_i$  définies par l'équation (5.5), la vitesse angulaire et l'accélération linéaires exprimées dans le repère du robot. Comme précisé dans la section 5.3, l'accélération mesurée par la centrale inertuelle ne mesure pas la gravité, il est donc nécessaire de soustraire celle-ci de l'accélération dans le modèle des mesures.

$$h(x) = h([O_B]_{C_W}, \mathbf{v}_{C_W}, \mathbf{a}_{C_B}, [R_B^W]_{C_W}, [\boldsymbol{\omega}_{WB}]_{C_B}) = \begin{pmatrix} d_i ([O_B]_{C_W}, [R_B^W]_{C_W}) \\ [\boldsymbol{\omega}_{WB}]_{C_B} \\ [R_B^W]_{C_W}^T ([\mathbf{a}]_{C_W} - [\mathbf{g}]_{C_W}) \end{pmatrix} \quad (5.13)$$

L'estimateur que je vais décrire dans la suite de cette section est un filtre de Kalman.

Un filtre de Kalman réalise l'estimation d'un état à partir d'une séquence de mesures et d'un a priori sur l'état initial, représenté par une distribution de probabilités sur celui-ci. Contrairement aux estimateurs détaillés précédemment, un filtre de Kalman utilise les mesures passées pour réaliser l'estimation de l'état et peut ainsi atténuer le bruit de mesure. Un filtre de Kalman fonctionne de manière récursive : l'estimation de l'état du système à l'instant  $t$  se calcule à partir de l'estimation à l'instant précédent  $t - 1$  et de la mesure courante, l'information relative aux mesures des instants antérieurs à l'instant courant et celle concernant l'état initial sont contenues dans l'estimation de l'état à l'instant  $t - 1$ .

Plutôt que d'estimer uniquement l'état, un filtre de Kalman estime une distribution l'état. Cette distribution est supposée suivre une loi normale et l'estimation de l'état consiste en l'estimation des paramètres de cette loi (moyenne et variance). Ainsi, en parallèle de l'estimation de l'état du système, un filtre de Kalman maintient une estimation de la variance de la distribution qui peut être interprétée comme une incertitude. Les propriétés mathématiques du filtre de Kalman sont décrites et prouvées dans l'ouvrage de SIMON 2006.

L'estimation de l'état d'un système via un filtre de Kalman procède en deux étapes principales :

- la prédition qui consiste à mettre à jour l'estimation de l'instant précédent en tenant compte du modèle dynamique ;
- la correction qui consiste à prendre en compte les mesures de l'instant courant pour corriger l'estimation faite à l'étape de prédition.

Contrairement aux hypothèses d'application classiques du filtre de Kalman, l'état du robot présente une dynamique et un modèle de mesure non linéaires. J'utilise ici une approximation de ces modèles au premier ordre pour réaliser l'estimation comme avec un filtre de Kalman classique. Cette approche est appelée filtre de Kalman étendu (EKF) (SIMON 2006).

On note :

- $x_{t|t-1} \in \mathbb{R}^n$  l'estimation de l'état à l'instant  $t$  tenant compte des mesures jusqu'à l'instant  $t$  exclus (estimation pré-correction) ;
- $x_{t|t} \in \mathbb{R}^n$  l'estimation de l'état à l'instant  $t$  tenant compte des mesures jusqu'à l'instant  $t$  inclus ;
- $P_{t|t-1} \in \mathbb{R}^{n \times n}$  la variance de l'estimation à l'instant  $t$  tenant compte des mesures jusqu'à l'instant  $t$  exclus (variance pré-correction) ;
- $P_{t|t} \in \mathbb{R}^{n \times n}$  la variance à l'instant  $t$  tenant compte des mesures jusqu'à l'instant  $t$  inclus ;
- $z_t \in \mathbb{R}^m$  la mesure réalisée à l'instant  $t$ .

On supposera ainsi que le système vérifie les équations suivantes :

$$\begin{aligned} x_t &= f(x_{t-1}) + w_t \\ z_t &= h(x_t) + v_t \end{aligned} \tag{5.14}$$

Où  $w_t \sim \mathcal{N}(0, Q_t)$  et  $v_t \sim \mathcal{N}(0, R_t)$  sont des bruits gaussiens rendant respectivement la dynamique du système incertaine et les mesures sur celui-ci imprécises. Les équations de l'étape de prédition sont les suivantes :

$$\begin{aligned} x_{t|t-1} &= f(x_{t-1|t-1}) \\ P_{t|t-1} &= F_t P_{t-1|t-1} F_t^T + Q_t \end{aligned} \tag{5.15}$$

Ici  $F_t$  est l'approximation au premier ordre de la dynamique  $f$  au voisinage de  $x_{t-1|t-1}$  :

$$F_t = \left. \frac{\partial f(x)}{\partial x} \right|_{x_{t-1|t-1}} \in \mathbb{R}^{n \times n} \tag{5.16}$$

Ensuite vient l'étape de correction où est calculé un gain de Kalman  $K_t$  pour mettre à jour l'état estimé à partir de l'erreur entre la mesure issue des capteurs  $z_t$  et la mesure issue du modèle  $h(x_{t|t-1})$ .

$$\begin{aligned} S_t &= H_t P_{t|t-1} H_t^T + R_t \in \mathbb{R}^{m \times m} \\ K_t &= P_{t|t-1} H_t^T S_t^{-1} \in \mathbb{R}^{n \times m} \\ y_t &= z_t - h(x_{t|t-1}) \in \mathbb{R}^m \\ x_{t|t} &= x_{t|t-1} + K_t y_t \\ P_{t|t} &= (I_n - K_t H_t) P_{t|t-1} \end{aligned} \tag{5.17}$$

On appelle  $y_t$  l'innovation, et de manière similaire à  $F_t$ ,  $H_t$  est l'approximation au premier ordre de la mesure  $h$  au voisinage de  $x_{t|t-1}$

$$H_t = \left. \frac{\partial h(x)}{\partial x} \right|_{x_{t|t-1}} \in \mathbb{R}^{m \times n} \quad (5.18)$$

Pour réaliser le filtrage décrit, il est nécessaire de dériver l'approximation au premier ordre de  $f$  et  $h$  tels qu'exprimés par les équations (5.12) et (5.13).

### 5.6.1 Jacobienne de la dynamique $f$

Pour alléger l'expression de l'approximation au premier ordre de  $f$  on note dans cette partie uniquement :

$$\begin{aligned} [O_B]_{C_W} &= p \in \mathbb{R}^3 \\ [\mathbf{v}]_{C_W} &= v \in \mathbb{R}^3 \\ [\mathbf{a}]_{C_W} &= a \in \mathbb{R}^3 \\ [R_B^W]_{C_W} &= R \in \mathbb{R}^{3 \times 3} \\ [\boldsymbol{\omega}_{WB}]_{C_B} &= \omega \in \mathbb{R}^3 \end{aligned} \quad (5.19)$$

De plus, on utilisera la notation  $R$  pour une matrice et pour le vecteur colonne de ses coordonnées triées de gauche à droite puis de haut en bas pour pouvoir utiliser la dérivée partielle par rapport à  $R$ . On procédera de même pour la quantité  $R(I_3 + [\omega]_\wedge)$  qui sera traitée comme un vecteur colonne.

On a la matrice par blocs suivante détaillant les différentes composantes de l'approximation au premier ordre de  $f$  :

$$\left. \frac{\partial f}{\partial x} \right|_x = \begin{pmatrix} \left. \frac{\partial p+vdt}{\partial p} \right|_x & \left. \frac{\partial p+vdt}{\partial v} \right|_x & \left. \frac{\partial p+vdt}{\partial a} \right|_x & \left. \frac{\partial p+vdt}{\partial R} \right|_x & \left. \frac{\partial p+vdt}{\partial \omega} \right|_x \\ \left. \frac{\partial v+adt}{\partial p} \right|_x & \left. \frac{\partial v+adt}{\partial v} \right|_x & \left. \frac{\partial v+adt}{\partial a} \right|_x & \left. \frac{\partial v+adt}{\partial R} \right|_x & \left. \frac{\partial v+adt}{\partial \omega} \right|_x \\ \left. \frac{\partial a}{\partial p} \right|_x & \left. \frac{\partial a}{\partial v} \right|_x & \left. \frac{\partial a}{\partial a} \right|_x & \left. \frac{\partial a}{\partial R} \right|_x & \left. \frac{\partial a}{\partial \omega} \right|_x \\ \left. \frac{\partial R(I_3 + [\omega]_\wedge dt)}{\partial p} \right|_x & \left. \frac{\partial R(I_3 + [\omega]_\wedge dt)}{\partial v} \right|_x & \left. \frac{\partial R(I_3 + [\omega]_\wedge dt)}{\partial a} \right|_x & \left. \frac{\partial R(I_3 + [\omega]_\wedge dt)}{\partial R} \right|_x & \left. \frac{\partial R(I_3 + [\omega]_\wedge dt)}{\partial \omega} \right|_x \\ \left. \frac{\partial \omega}{\partial p} \right|_x & \left. \frac{\partial \omega}{\partial v} \right|_x & \left. \frac{\partial \omega}{\partial a} \right|_x & \left. \frac{\partial \omega}{\partial R} \right|_x & \left. \frac{\partial \omega}{\partial \omega} \right|_x \end{pmatrix} \quad (5.20)$$

Or

$$\begin{aligned} \left. \frac{\partial p+vdt}{\partial a} \right|_x &= \left. \frac{\partial p+vdt}{\partial R} \right|_x = \left. \frac{\partial p+vdt}{\partial \omega} \right|_x \\ &= \left. \frac{\partial v+adt}{\partial p} \right|_x = \left. \frac{\partial v+adt}{\partial R} \right|_x = \left. \frac{\partial v+adt}{\partial \omega} \right|_x \\ &= \left. \frac{\partial a}{\partial p} \right|_x = \left. \frac{\partial a}{\partial v} \right|_x = \left. \frac{\partial a}{\partial R} \right|_x = \left. \frac{\partial a}{\partial \omega} \right|_x \\ &= \left. \frac{\partial R(I_3 + [\omega]_\wedge dt)}{\partial p} \right|_x = \left. \frac{\partial R(I_3 + [\omega]_\wedge dt)}{\partial v} \right|_x = \left. \frac{\partial R(I_3 + [\omega]_\wedge dt)}{\partial a} \right|_x \\ &= \left. \frac{\partial \omega}{\partial p} \right|_x = \left. \frac{\partial \omega}{\partial v} \right|_x = \left. \frac{\partial \omega}{\partial a} \right|_x = \left. \frac{\partial \omega}{\partial R} \right|_x \\ &= 0 \end{aligned} \quad (5.21)$$

Et :

$$\begin{aligned} \frac{\partial p + vdt}{\partial p} \Big|_x &= \frac{\partial v + adt}{\partial v} \Big|_x = \frac{\partial a}{\partial a} \Big|_x = \frac{\partial \omega}{\partial \omega} \Big|_x = I_3 \\ \frac{\partial p + vdt}{\partial v} \Big|_x &= \frac{\partial v + adt}{\partial a} \Big|_x = dtI_3 \end{aligned} \quad (5.22)$$

Ce qui donne :

$$\frac{\partial f}{\partial x} \Big|_x = \begin{pmatrix} I_3 & dtI_3 & 0 & 0 & 0 \\ 0 & I_3 & dtI_3 & 0 & 0 \\ 0 & 0 & I_3 & 0 & 0 \\ 0 & 0 & 0 & I_9 + dt \frac{\partial R([\omega]_{\wedge})}{\partial R} \Big|_x & dt \frac{\partial R([\omega]_{\wedge})}{\partial \omega} \Big|_x \\ 0 & 0 & 0 & 0 & I_3 \end{pmatrix} \quad (5.23)$$

De plus :

$$R[\omega]_{\wedge} = \begin{pmatrix} R_{0,1}\omega_z - R_{0,2}\omega_y & -R_{0,0}\omega_z + R_{0,2}\omega_x & R_{0,0}\omega_y - R_{0,1}\omega_x \\ R_{1,1}\omega_z - R_{1,2}\omega_y & -R_{1,0}\omega_z + R_{1,2}\omega_x & R_{1,0}\omega_y - R_{1,1}\omega_x \\ R_{2,1}\omega_z - R_{2,2}\omega_y & -R_{2,0}\omega_z + R_{2,2}\omega_x & R_{2,0}\omega_y - R_{2,1}\omega_x \end{pmatrix} \quad (5.24)$$

donc

$$\frac{\partial R[\omega]_{\wedge}}{\partial R} \Big|_x = \begin{pmatrix} [\omega]_{\wedge}^T & 0 & 0 \\ 0 & [\omega]_{\wedge}^T & 0 \\ 0 & 0 & [\omega]_{\wedge}^T \end{pmatrix} \quad (5.25)$$

et

$$\frac{\partial R[\omega]_{\wedge}}{\partial \omega} \Big|_x = \begin{pmatrix} 0 & -R_{0,2} & R_{0,1} \\ R_{0,2} & 0 & -R_{0,0} \\ -R_{0,1} & R_{0,0} & 0 \\ 0 & -R_{1,2} & R_{1,1} \\ R_{1,2} & 0 & -R_{1,0} \\ -R_{1,1} & R_{1,0} & 0 \\ 0 & -R_{2,2} & R_{2,1} \\ R_{2,2} & 0 & -R_{2,0} \\ -R_{2,1} & R_{2,0} & 0 \end{pmatrix} \quad (5.26)$$

### 5.6.2 Jacobienne du modèle de mesure $h$

On utilisera les mêmes notations simplifiées pour les variables d'état et pour la représentation de matrices en vecteurs colonnes que dans la section précédente. On y ajoute le vecteur des mesures de distance  $d = (d_i)_i$  et  $g = -9.81\mathbf{z}_W$  le vecteur d'accélération de gravité.

En partant de l'équation (5.13) pour obtenir l'approximation au premier ordre de  $h$  on obtient la matrice par blocs suivante :

$$\frac{\partial h}{\partial x} \Big|_x = \begin{pmatrix} \frac{\partial d}{\partial p} \Big|_x & \frac{\partial d}{\partial v} \Big|_x & \frac{\partial d}{\partial a} \Big|_x & \frac{\partial d}{\partial R} \Big|_x & \frac{\partial d}{\partial \omega} \Big|_x \\ \frac{\partial \omega}{\partial p} \Big|_x & \frac{\partial \omega}{\partial v} \Big|_x & \frac{\partial \omega}{\partial a} \Big|_x & \frac{\partial \omega}{\partial R} \Big|_x & \frac{\partial \omega}{\partial \omega} \Big|_x \\ \frac{\partial R^T(a-g)}{\partial p} \Big|_x & \frac{\partial R^T(a-g)}{\partial v} \Big|_x & \frac{\partial R^T(a-g)}{\partial a} \Big|_x & \frac{\partial R^T(a-g)}{\partial R} \Big|_x & \frac{\partial R^T(a-g)}{\partial \omega} \Big|_x \end{pmatrix} \quad (5.27)$$

Avec

$$\begin{aligned}
\frac{\partial d}{\partial v} \Big|_x &= \frac{\partial d}{\partial a} \Big|_x = \frac{\partial d}{\partial \omega} \Big|_x \\
&= \frac{\partial \omega}{\partial p} \Big|_x = \frac{\partial \omega}{\partial v} \Big|_x = \frac{\partial \omega}{\partial a} \Big|_x = \frac{\partial \omega}{\partial R} \Big|_x \\
&= \frac{\partial R^T(a-g)}{\partial p} \Big|_x = \frac{\partial R^T(a-g)}{\partial v} \Big|_x = \frac{\partial R^T(a-g)}{\partial \omega} \Big|_x \\
&= 0
\end{aligned} \tag{5.28}$$

Ce qui donne :

$$\frac{\partial h}{\partial x} \Big|_x = \begin{pmatrix} \frac{\partial d}{\partial p} \Big|_x & 0 & 0 & \frac{\partial d}{\partial R} \Big|_x & 0 \\ 0 & 0 & 0 & 0 & I_3 \\ 0 & 0 & R^T & \frac{\partial R^T(a-g)}{\partial R} \Big|_x & 0 \end{pmatrix} \tag{5.29}$$

Avec

$$\frac{\partial R^T(a-g)}{\partial R} \Big|_x = \begin{pmatrix} a_0 - g_0 & 0 & 0 & a_1 - g_1 & 0 & 0 & a_2 - g_2 & 0 & 0 \\ 0 & a_0 - g_0 & 0 & 0 & a_1 - g_1 & 0 & 0 & a_2 - g_2 & 0 \\ 0 & 0 & a_0 - g_0 & 0 & 0 & a_1 - g_1 & 0 & 0 & a_2 - g_2 \end{pmatrix} \tag{5.30}$$

En posant les notations suivantes, où l'indice  $i$  réfère à l'indice d'un des 10 télémètres et  $s$  est la direction de mesure de ce capteur :

$$\begin{aligned}
[s_i]_{C_W} &= (s_{i,0}, s_{i,1}, s_{i,2}) \\
[s_i]_{C_B} &= (s_{i,0,b}, s_{i,1,b}, s_{i,2,b}) \\
p &= (O_{B,x}, O_{B,y}, O_{B,z}) = (p_0, p_1, p_2) \\
\alpha_i &= s_{i,0}p_0 + s_{i,2}p_2 \\
\beta_i &= s_{i,0}^2 + s_{i,2}^2 \\
\gamma_i &= s_{i,0}p_2 - s_{i,2}p_0
\end{aligned} \tag{5.31}$$

$\alpha_i, \beta_i$  et  $\gamma_i$  sont des termes partiels de l'expression de la distance mesurée par le capteur  $i$ . Ces termes sont introduits ici pour simplifier l'expression de la mesure des télémètres qui devient d'après l'équation 5.5 :

$$d_i = \frac{\sqrt{r^2\beta_i - \gamma_i^2} - \alpha_i}{\beta_i} \tag{5.32}$$

et ainsi pour une variable générique  $X$  de l'état on a :

$$\frac{\partial d_i}{\partial X} \Big|_x = \frac{1}{\beta_i^2} \left( \beta_i \left( \frac{r^2 \frac{\partial \beta_i}{\partial X} \Big|_x - 2\gamma_i \frac{\partial \gamma_i}{\partial X} \Big|_x}{2\sqrt{r^2\beta_i - \gamma_i^2}} - \frac{\partial \alpha_i}{\partial X} \Big|_x \right) - \left( \sqrt{r^2\beta_i - \gamma_i^2} - \alpha_i \right) \frac{\partial \beta_i}{\partial X} \Big|_x \right) \tag{5.33}$$

$$\begin{aligned}
\frac{\partial \alpha_i}{\partial p} \Big|_x &= \begin{pmatrix} s_{i,0} \\ 0 \\ s_{i,2} \end{pmatrix} \\
\frac{\partial \beta_i}{\partial p} \Big|_x &= 0 \\
\frac{\partial \gamma_i}{\partial p} \Big|_x &= \begin{pmatrix} -s_{i,2} \\ 0 \\ s_{i,0} \end{pmatrix}
\end{aligned} \tag{5.34}$$

De plus, les coordonnées de  $s_i$  dans le repère inertiel vérifient

$$s_i = R s_{i,b} \quad (5.35)$$

On obtient alors

$$\begin{aligned} \frac{\partial \alpha_i}{\partial R} \Big|_x &= \begin{pmatrix} p_0 s_{i,0,b} \\ p_0 s_{i,1,b} \\ p_0 s_{i,2,b} \\ 0 \\ 0 \\ 0 \\ p_2 s_{i,0,b} \\ p_2 s_{i,1,b} \\ p_2 s_{i,2,b} \end{pmatrix} \\ \frac{\partial \beta_i}{\partial R} \Big|_x &= \begin{pmatrix} 2s_{i,0}s_{i,0,b} \\ 2s_{i,0}s_{i,1,b} \\ 2s_{i,0}s_{i,2,b} \\ 0 \\ 0 \\ 0 \\ 2s_{i,2}s_{i,0,b} \\ 2s_{i,2}s_{i,1,b} \\ 2s_{i,2}s_{i,2,b} \end{pmatrix} \\ \frac{\partial \gamma_i}{\partial R} \Big|_x &= \begin{pmatrix} p_2 s_{i,0,b} \\ p_2 s_{i,1,b} \\ p_2 s_{i,2,b} \\ 0 \\ 0 \\ 0 \\ -p_0 s_{i,0,b} \\ -p_0 s_{i,1,b} \\ -p_0 s_{i,2,b} \end{pmatrix} \end{aligned} \quad (5.36)$$

En combinant les équations (5.33), (5.34), (5.36), avec (5.29) et (5.30) on obtient  $H$ .

Comme pour les deux estimateurs précédents, quand les télémètres sont proches d'être alignés avec l'axe  $y$  cylindre, les mesures théoriques (telles que données par l'équation 5.5) tendent vers l'infini, et les mesures pratiques réalisées par les capteurs embarqués saturent. En conséquence, pour éviter les instabilités numériques dues et l'usage de mesures incorrectes, certaines mesures peuvent être préalablement écartées si celles-ci correspondent à des distances trop importantes (10 fois le rayon du cylindre), faisant ainsi varier la taille  $m$  du vecteur de mesure et des matrices associées ( $H_t, R_t, K_t$ ).

Par ailleurs, vu la nature récursive du filtre de Kalman discret, l'état estimé est utilisé comme argument de la fonction de mesure, ceci peut poser problème si l'état estimé comprend une position en dehors du cylindre. En effet l'expression de la distance en fonction de l'état a pour hypothèse importante que la position estimée est à l'intérieur du cylindre. L'expression de la distance peut ne pas être définie si cette hypothèse n'est pas vérifiée. Dans le cas où l'état obtenu à l'étape de prédiction est à l'extérieur du cylindre, il faut rajouter une étape de correction de cet état au filtre qui ramène la position estimée du robot à l'intérieur du cylindre.

Le modèle dynamique de la variation de la matrice de rotation  $R$  ne tient pas compte du fait que les matrices de rotation ne sont pas stables par somme. Ainsi, l'accumulation d'erreurs peut éloigner la matrice estimée pour l'orientation de l'espace des matrices de rotation et poser ainsi des difficultés supplémentaires dans l'estimation. Il est possible de re-normaliser régulièrement l'estimation de l'orientation afin de prévenir l'accumulation d'erreurs, mais il est aussi possible de formuler l'estimateur d'état de manière à exprimer la variation de l'orientation dans un espace local tangent à l'espace des matrices de rotation et d'utiliser une carte locale décrivant la correspondance entre l'espace tangent et l'espace des matrices de rotation. SOLÀ, DERAY et ATCHUTHAN 2021 détaillent les équations décrivant de telles cartes locales pour les rotations et d'autres espaces utiles pour la robotique et BROSSARD, BARRAU et BONNABEL 2020 décrivent l'utilisation de telles coordonnées locales dans l'implémentation d'un filtre de Kalman sans parfum (UKF).

L'implémentation de ce filtre de Kalman étendu n'a pas pu être achevée, mais cet estimateur pourrait permettre l'exploitation conjointe des mesures de distances réalisées par les télémètres et celles de vitesses angulaires et d'accélération réalisées par la centrale inertuelle embarquée.

## 5.7 Évaluation des estimations sur des données réelles

Les méthodes présentées dans ce chapitre n'ont pour le moment été évaluées que sur des données en simulation. Cette évaluation en simulation a nécessité de faire des hypothèses sur plusieurs aspects :

- la distribution et les paramètre du bruit de mesure des différents capteurs, en particulier les télémètres ;
- les modalités de mesure des télémètres, ceux-ci sont modélisés comme mesurant la distance le long d'une droite alors que les capteurs ont un angle d'ouverture important ( $30^\circ$ ) ;
- la dynamique du Crazyflie.

Pour tester une partie de ces hypothèses, nous avons réalisé des mesures avec un Crazyflie volant dans un tuyau de 56cm de diamètre. Le Crazyflie est piloté manuellement afin d'éviter les collisions au maximum. Un dispositif externe de mesure de la position est utilisé dans le but de fournir un point de comparaison avec l'état estimé par les deux approches présentées plus tôt. Ce dispositif externe de mesure de la position consiste en deux boîtiers externe dit *lighthouse* qui balaien l'espace avec des lasers, comme des phares. Ces faisceaux balayant l'espace sont reçus à bord du Crazyflie par un deck conçu à cet effet contenant plusieurs capteurs. La différence de temps entre la réception d'un même faisceau les différents capteurs du deck permet d'estimer la distance angulaire entre les capteurs par rapport au boîtier externe. Ces mesures angulaires permettent ensuite d'estimer la position du quadrotor par rapport au boîtier. Cette estimation se fait directement à bord du Crazyflie.

Nous avons placé deux boîtiers lighthouse de manière à ce que l'intérieur du tuyau soit balayé correctement. Durant le vol, à fréquence fixée, le firmware embarqué du Crazyflie enregistre les valeurs mesurées par les capteurs sur une carte SD embarquée. Ceci permet d'utiliser les implémentations des approches par régression linéaire et par modèle inverse utilisées en simulation sur des données réelles. Il serait bien entendu possible de réaliser directement une implémentation de ces approches directement à bord du Crazyflie.

Les mesures collectées durant ces vols sont les suivantes :

- la distance mesurée par chacun des 10 télémètres ;
- la position mesurée à l'aide des boîtiers lighthouse ;
- l'estimation de l'orientation réalisée par le contrôle embarqué.

La position mesurée à l'aide des boîtiers lighthouse sert ainsi de point de comparaison pour évaluer la précision des estimations réalisée par les deux approches précédentes. L'approche par régression linéaire estime la position latérale et le lacet. L'estimation de l'orientation est utilisée comme point de comparaison de l'estimation du lacet.

L'approche par modèle inverse demande au préalable une estimation de l'orientation du Crazyflie. L'estimation que nous décidons d'utiliser est aussi celle réalisée par le contrôleur embarqué à l'aide des données inertielles.

A l'aide de ces données réelles et avec chaque je réalise deux estimations d'état :

- la première est faite directement à partir des données des télémètres et de l'estimation d'orientation et est appelée *estimation mesures réelles* ;
- pour la seconde j'utilise les mesures de position et d'orientation pour obtenir des mesures dites *théoriques* des télémètres via le modèle décrit par l'équation (5.5). Ces mesures de télémètres obtenues à l'aide du modèle permettent de réaliser une seconde estimation appelée *estimation mesures théoriques*.

L'estimation mesures réelles correspond à une estimation telle qu'elle serait réalisée directement à bord du Crazyflie. L'estimation théorique correspond à une estimation à la manière de la simulation, mais pour des positions et orientations qui sont celles mesurées pour une trajectoire réelle. La figure 5.9 présente une comparaison, sur deux trajectoires, des mesures réelles des télémètres (en orange), utilisées pour réaliser l'estimation mesures réelles, aux mesures des télémètres obtenues à l'aide du modèle (en bleu) et utilisées pour l'estimation mesures théoriques.

L'écart entre ces mesures issues du modèle et les mesures réelles est la conséquence de trois phénomènes :

- l'erreur de mesure et de modélisation des télémètres (placement pas exactement au centre du Crazyflie, biais sur la mesure, etc) ;
- l'erreur sur la mesure de la position et l'estimation de l'orientation qui servent de base pour déduire les mesures théoriques du télémètre ;
- le tuyau est de longueur finie, et certains télémètres mesurent des distances le long d'axes qui sortent du tuyau si le Crazyflie est proche des extrémités du tuyau.

Malgré ces trois phénomènes, les mesures issues du modèle et les mesures réelles coïncident dans de très nombreuses situations. Le lissage des mesures des télémètres ne semble pas apporter beaucoup de régularité puisque les mesures non lissées sont déjà très régulières sur les segments où les distances mesurées par les télémètres sont courtes.

Les figures 5.10 et 5.11 présentent respectivement les estimations par régression linéaire et par modèle inverse. Ces figures comparent trois grandeurs :

- les mesures à l'aide des lighthouses et l'estimation interne de l'orientation, appelées *mesures lighthouse* ou *estimations Kalman* (bleu) et qui servent de point de comparaison pour les estimations ;
- les estimations réalisées à partir des mesures réelles des télémètres appelées *estimations mesures réelles* (vert) ;
- les estimations réalisées à partir des mesures issues du modèle des télémètres appelées *estimations mesures théoriques* (orange).

Ces trois catégories ont pour objectif de faire la distinction entre les différents phénomènes qui sont la cause des erreurs d'estimation. Les estimations mesures réelles permettent par exemple d'ignorer les erreurs dues aux biais et bruits de mesure des télémètres et de valider ou d'invalider la pertinence des hypothèses de l'approche par régression linéaire. Sur chacune de ces figures, les deux graphiques du haut correspondent à une trajectoire pour laquelle les mesures sont réalisées

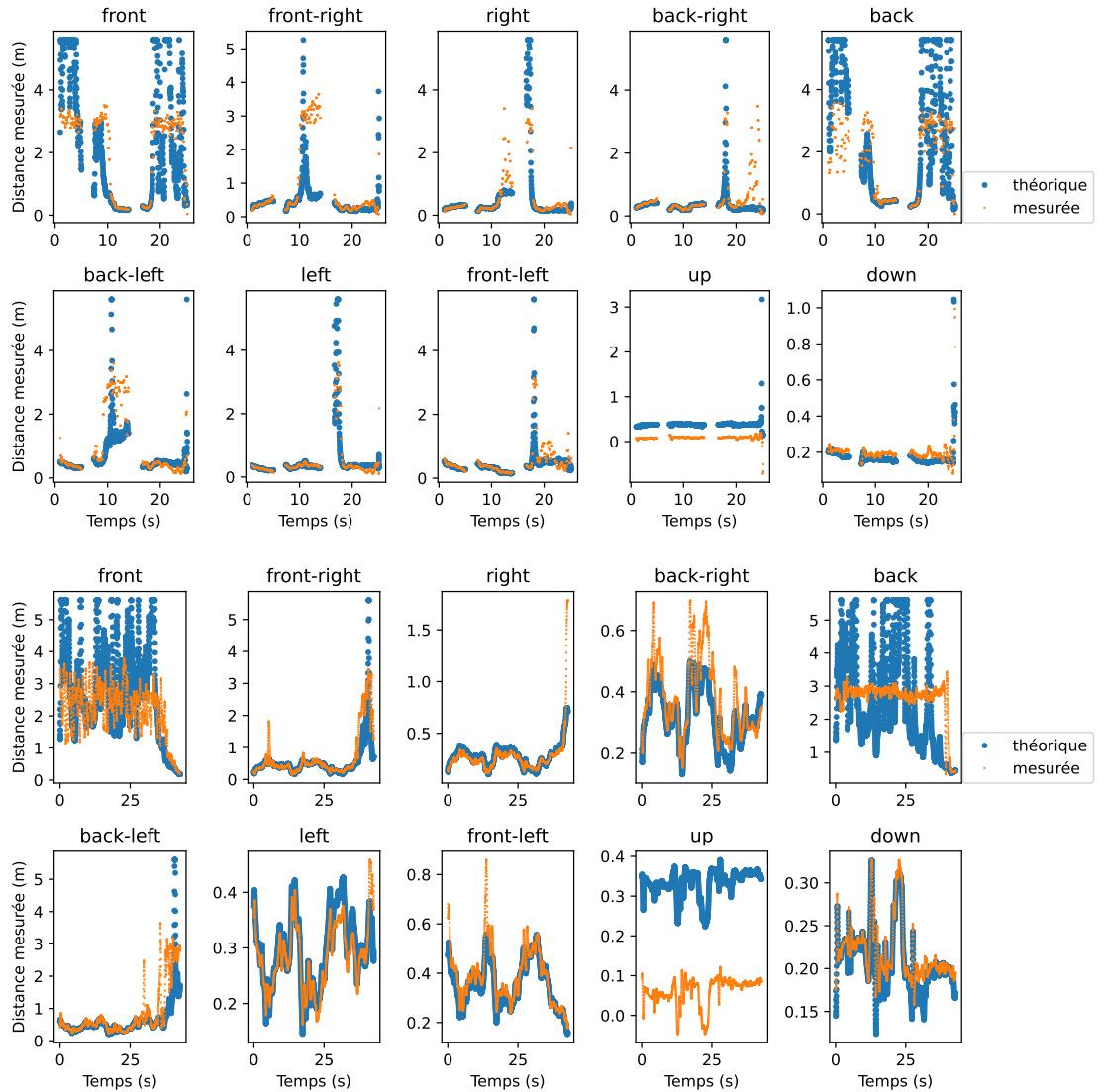


FIGURE 5.9 – Comparaison des mesures effectives des télémètres (orange) avec les mesures attendues pour les télémètres d'après le modèle, les mesures de position et d'orientation (bleu). Les deux rangées supérieures correspondent à des mesures collectées à 50Hz. Les deux rangées inférieures correspondent à des mesures réalisées à 500Hz, lissée avec un filtre de Savitzky-Golay et sous échantillonnées à 50Hz. La mesure correspondant au télémètre nommé "up" ne représente pas les véritables mesures de celui-ci, cachées par le deck lighthouse qui recouvre le télémètre. Elles sont remplacées par la différence entre le rayon du tuyau et la mesure qui correspond eu télémètre complémentaire ("down"). Ceci explique en partie le biais important observé sur ce capteur. Sur de nombreux segments les mesures réelles et issues du modèle coïncident. Les segments où les mesures réelles et issues du modèle sont étalées correspondent à des distances pour lesquelles les télémètres saturent (1m-3m). Les interruptions de trajectoire sur les deux premières rangées supérieures (à 5s et 15s) correspondent à des moments du vol où le Crazyflie est sorti du tuyau et pour lesquelles le modèle ne peut définir formellement une mesure attendue des télémètres.

à 50Hz et ne sont pas lissées. Les deux graphiques du bas correspondent à une trajectoire pour laquelle les mesures sont réalisées à 500Hz, lissées à l'aide d'un filtre de Savitzky-Golay (SAVITZKY et GOLAY 1964) puis sous échantillonnées à 50Hz.

Sur la figure 5.10 on observe par exemple que les mesures lighthouse et les mesures théoriques coïncident de manière presque systématique. Le segment entre 17s et la fin (sur les deux graphiques du haut) on observe que les estimations mesures théoriques et les mesures lighthouse ne correspondent pas du tout. Les deux sont symétriques par rapport à 0. Ceci est dû à un demi tour du Crazyflie. Vu la géométrie du tuyau, il n'est pas possible de faire la distinction entre deux orientations du Crazyflie qui diffèrent de 180°. Ces deux estimations coïncident donc sur ce segment. Sur ce segment on peut aussi observer que les mesures lighthouse sont constantes par morceaux. Ceci est en fait dû à une situation de perte de visibilité sur les balises lighthouse. Dans une telle situation il est possible que les estimations mesures réelles soient plus pertinentes que les mesures lighthouse. On peut déduire de la coïncidence entre les estimations théoriques et les mesures lighthouse que l'hypothèse utilisée par l'approche par régression linéaire est bien vérifiée. Cette hypothèse selon laquelle l'inclinaison du Crazyflie est proche de l'horizontale n'est pas exactement vérifiée, mais semble l'être suffisamment pour que l'approche par régression linéaire permettent une estimation d'état précise dans sa version théorique (c'est-à-dire à partir de mesures issues du modèle).

Les estimations mesures réelles de lacet sur les deux graphes du haut de la figure 5.10 présentent une dispersion importante sur le segment entre 0s et 5s. C'est cette dispersion qui a motivé l'ajout d'une étape de lissage pour la seconde trajectoire (deux graphiques du bas). Cependant, on peut observer que l'estimation du lacet pour la trajectoire dont les mesures sont lissées présente aussi une dispersion importante que l'étape de lissage n'a pas corrigé. On observe aussi une dispersion moins importante sur l'estimation de la position latérale bien que cette estimation mesures réelles suive d'assez près les mesures lighthouse et les mesures théoriques.

Sur la figure 5.11 on observe aussi que les mesures théoriques et les mesures lighthouse coïncident de manière systématique. Les estimations mesures réelles de la position latérale et verticale sur la trajectoire aux mesures non lissées (les deux graphiques du haut) présentent une dispersion importante. Cette dispersion semble un peu moins importante sur la trajectoire aux mesures lissées (deux graphiques du bas). Outre cette dispersion, sur la trajectoire aux données filtrées l'estimation de position verticale et latérale à partir des mesures réelles suit les mesures lighthouse et les estimations mesures théoriques. Cependant cette estimation de position à partir de mesures réelles présente des erreurs importantes (de l'ordre de 10cm) ce qui est trop important vu les dimensions de l'environnement dans lequel le Crazyflie évolue.

Les estimations réalisées à partir de mesures réelles présentent des erreurs importantes (10cm en position, plusieurs dizaines de degrés en lacet). Ces erreurs semblent trop importantes pour utiliser directement ces mesures pour voler dans un tuyau de petit diamètre. Les estimations sur mesures théoriques valident cependant les deux approches utilisées (par régression linéaire et modèle inverse) et invitent à affiner la modélisation des mesures des télémètres.

La première piste pour améliorer le modèle de mesure des télémètres consiste à tenir compte de la position des télémètres qui n'est pas exactement confondue avec celle d'un drone considéré comme ponctuel dans le modèle des mesures. La prise en compte de l'ouverture des faisceaux des télémètres lors de la mesure constitue une seconde piste, mais demanderait un travail important pour adapter les approches formulées dans ce chapitre. Une dernière piste consiste à réaliser ces mesures dans un tuyau de longueur plus importante pour s'assurer que tous les télémètres puissent mesurer la distance jusqu'aux parois plutôt que de sortir du tuyau.

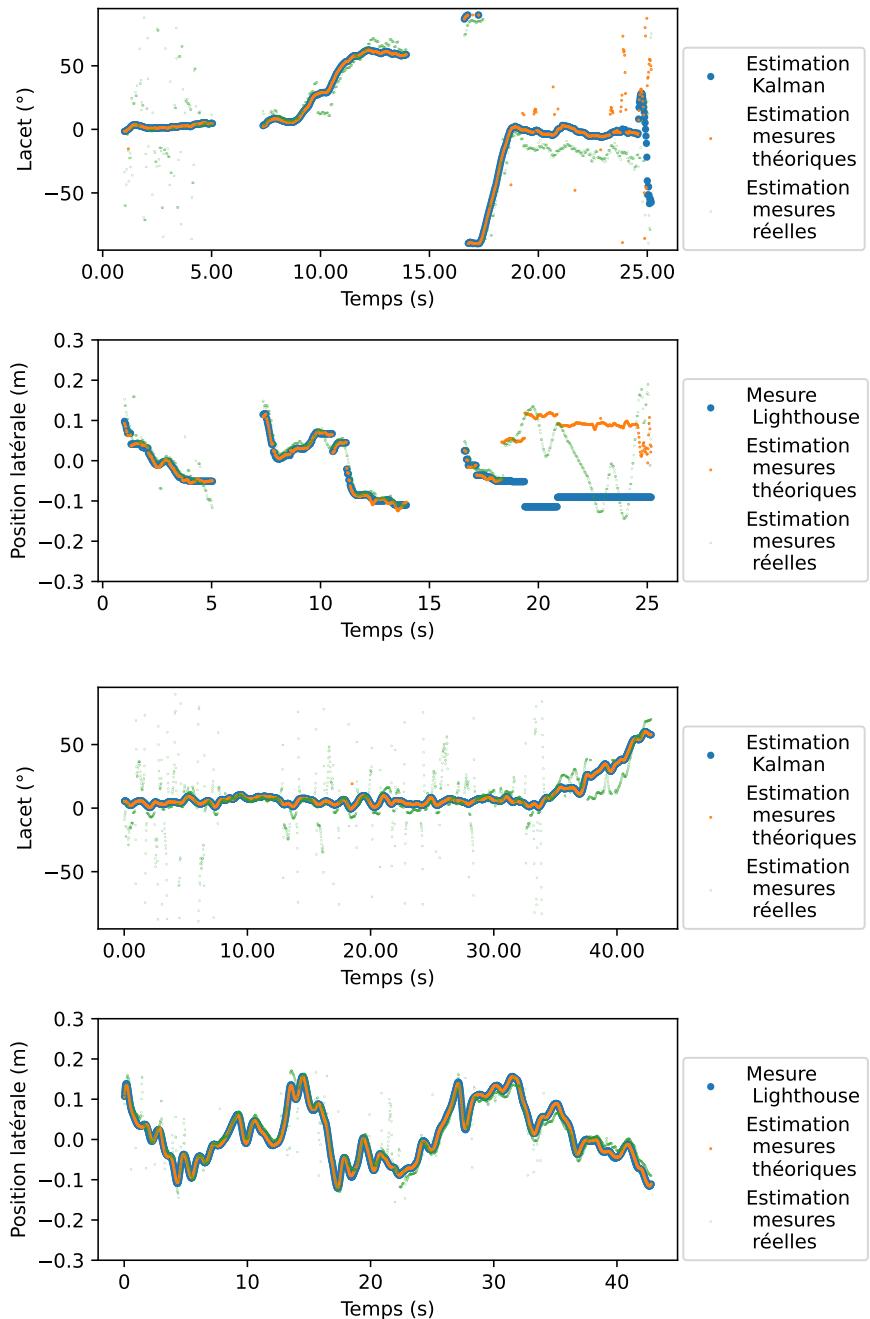


FIGURE 5.10 – Comparaison des estimations par régression linéaire à partir des mesures réelles des télémètres (vert) avec les estimations par régression linéaire issues des mesures attendues pour les télémètres d'après le modèle (orange) et avec les mesures de position du lighthouse et les estimations d'orientation du contrôleur embarqué (bleu). Les deux graphiques supérieurs correspondent à des estimations à partir de mesures collectées à 50Hz. Les deux graphiques inférieurs correspondent à des estimations à partir de mesures réalisées à 500Hz, lissée avec un filtre de Savitzky-Golay et sous échantillonnées à 50Hz. Les estimations ne sont pas elles-mêmes lissées. Les segments où les mesures sont constantes correspondent à des positions pour lesquelles le Crazyflie n'était pas dans le champ de vision des boîtiers lighthouse. Les interruptions de trajectoire sur les deux premiers graphiques supérieurs (à 5s et 15s) correspondent à des moments du vol où le Crazyflie est sorti du tuyau et pour lesquelles le modèle ne peut définir formellement une mesure attendue des télémètres.

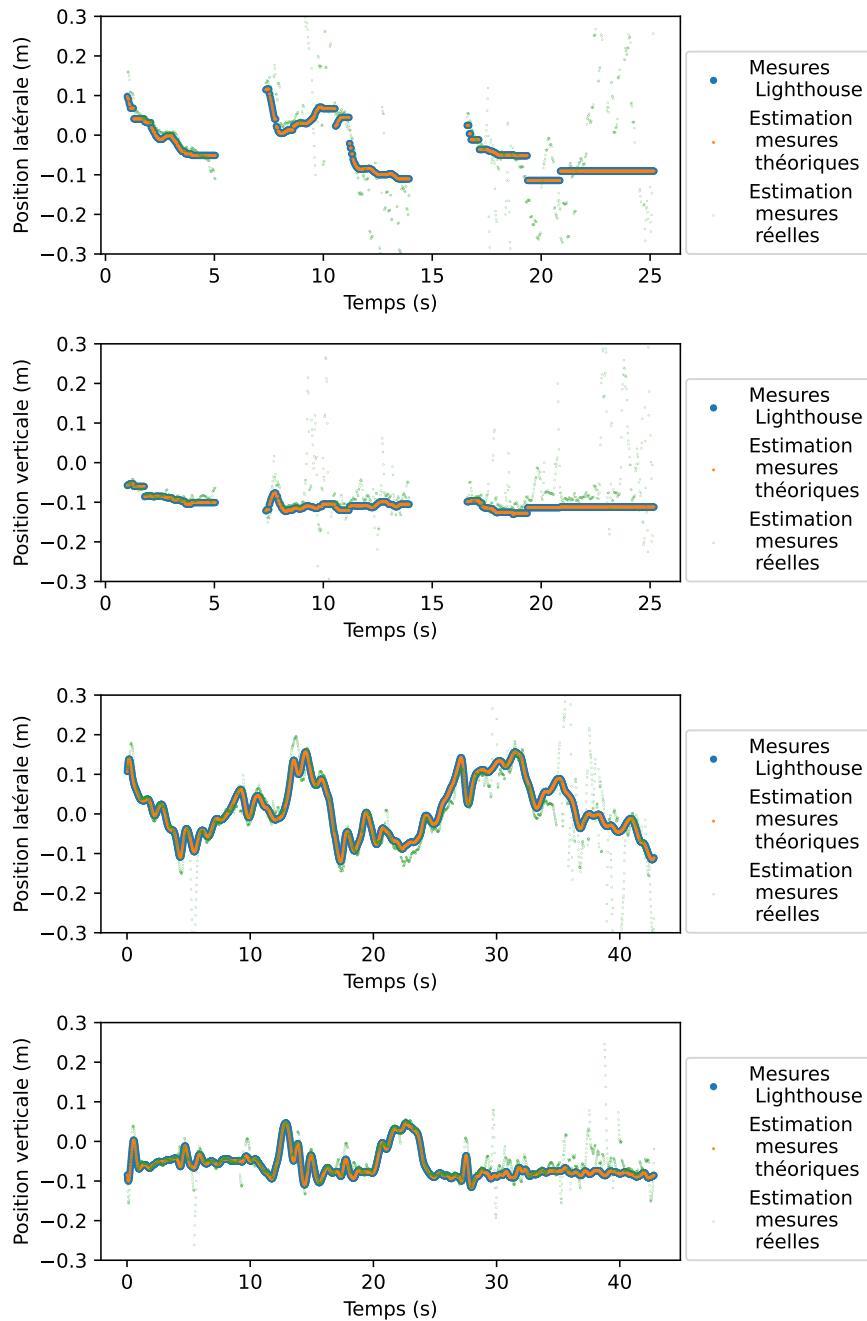


FIGURE 5.11 – Comparaison des estimations par modèle inverse à partir des mesures réelles des télémètres (vert) avec les estimations par modèle inverse issues des mesures attendues pour les télémètres d'après le modèle (orange) et avec les mesures de position du lighthouse (bleu). Les deux graphiques supérieurs correspondent à des estimations à partir de mesures collectées à 50Hz. Les deux graphiques inférieurs correspondent à des estimations à partir de mesures réalisées à 500Hz, lissée avec un filtre de Savitzky-Golay et sous échantillonnées à 50Hz. Les estimations ne sont pas elles-mêmes lissées. Les segments où les mesures sont constantes correspondent à des positions pour lesquelles le Crazyflie n'était pas dans le champ de vision des boîtiers lighthouse. Les interruptions de trajectoire sur les deux premiers graphiques supérieurs (à 5s et 15s) correspondent à des moments du vol où le Crazyflie est sorti du tuyau et pour lesquelles le modèle ne peut définir formellement une mesure attendue des télémètres.

Méthode	Entrées	Grandeurs estimées	Hypothèses	Commentaire
Régression linéaire (sec. 5.4)	Télémètres	Position latérale et lacet	Inclinaison du drone proche de l'horizontale	Hypothèse validée sur un vol réel, mais estimation à partir de mesures réelles trop imprécise
Modèle inverse (sec. 5.5)	Télémètres, estimation interne de l'orientation	Positions latérale et verticale	Connaissance de la géométrie du tuyau (rayon)	Estimation à partir de mesures réelles trop imprécise
Filtre de Kalman (sec 5.6)	Télémètres, données inertielles	Positions latérale et verticale, orientation	Connaissance d'un modèle dynamique du drone et de la géométrie du tuyau (rayon)	Implémentation non achevée

FIGURE 5.12 – Comparaison des méthodes d'estimation de l'état du robot dans un tuyau

## 5.8 Conclusion

La figure 5.12 récapitule les différentes méthodes présentées dans ce chapitre et leurs caractéristiques.

L'utilisation de la centrale inertuelle d'un côté pour estimer l'orientation du robot et du modèle inverse des mesures de distance de l'autre pour estimer la position ne permet pas une prise en compte aussi fine des corrélations entre les mesures qu'un filtre de Kalman intégrant d'un seul coup l'ensemble de ces mesures. C'est cependant une approche pratique puisque, même en dehors de la navigation dans des environnements spécifiques comme ceux présentés dans cette section, un estimateur d'orientation est nécessaire pour le contrôle d'un quadrotor. Un estimateur d'état basé sur un filtre de Kalman est en particulier déjà embarqué dans les Crazyflie. Une partie de cette estimation d'état est réalisée à partir des résultats décrits dans les travaux de MUELLER, HEHN et D'ANDREA 2017 ; MUELLER, HAMER et D'ANDREA 2015.

Les deux premières approches évaluées en simulation montrent une capacité satisfaisante à estimer une partie de la position ou de l'orientation du robot sans présenter de dérive de l'état, ce qui est attendu puisque ces approches n'utilisent pas les informations passées pour estimer l'état du robot, mais uniquement les mesures courantes. Cependant ces deux approches ne parviennent pas à estimer la correction de l'état du Crazyflie à l'aide de mesures réalisées en conditions réelles, mais plusieurs pistes existent pour les affiner.

# Chapitre 6

## Auto-organisation d'une flotte de drones pour explorer un tunnel

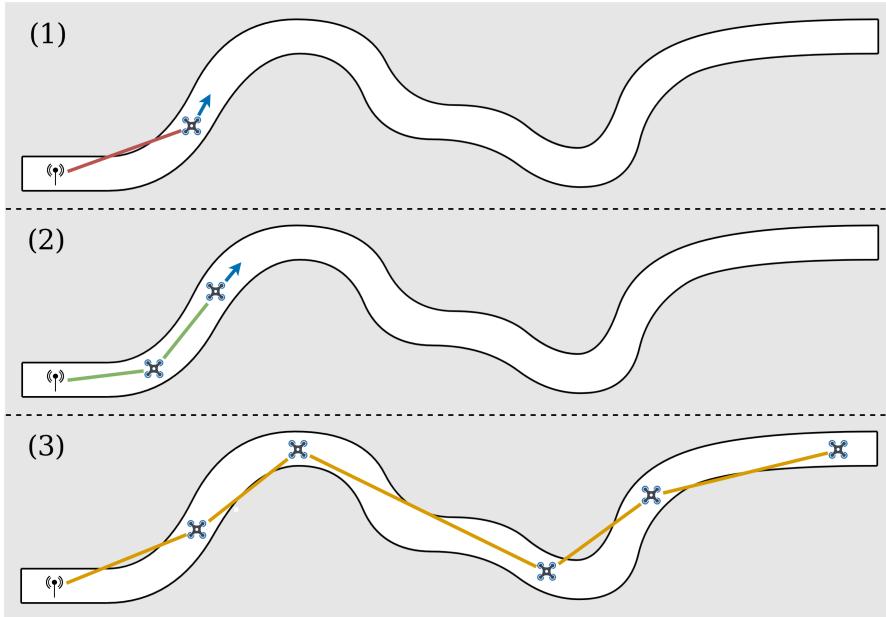
### 6.1 Introduction

Dans les chapitres précédents, nous avons présenté des travaux permettant le vol autonome de quadrotors dans un tuyau. Ces travaux décrivent des approches qui permettent au robot de naviguer dans de tels environnements, mais n'abordent pas la manière de réaliser l'exploration de ces environnements.

Dans ce chapitre, nous nous intéressons à un algorithme décentralisé qui organise l'exploration d'environnements comme des tuyaux ou des tunnels par une flotte de quadrotors. Cette exploration pourrait se faire par des robots autonomes et indépendants, mais nous avons choisi d'opter pour une exploration organisée par une flotte de robots qui permet de garder un lien de communication avec chacun des robots participant à l'exploration. Cela permet par exemple à un opérateur d'être informé en quasi temps réel des observations réalisées par les robots. L'algorithme présenté détermine des instructions de haut niveau pour l'exploration et pourrait s'appuyer sur ce qui est présenté dans les chapitres précédents, c'est-à-dire d'un modèle de ces perturbations aérodynamiques causées par la présence du drone dans un tuyau et aidant au contrôle de ce drone.

À l'air libre les communications par radio sont efficaces, mais dans les environnements intérieurs, les robots ont plus de difficultés pour établir un lien radio de bonne qualité sur des distances importantes en raison de l'incapacité du signal à traverser les murs, parois, plafonds et autres obstacles de ces environnements (KHUWAJA et al. 2018). Pour le Crazyflie en particulier, le signal utilisé (radio à 2.4GHz) est très perturbé par des parois métalliques de tuyaux. Ceci signifie que pour garder un lien radio entre les robots et l'opérateur, les robots doivent maintenir une ligne de vue entre eux et avec l'opérateur.

L'algorithme que nous présentons dans ce chapitre, appelé *U-Chain* (Underground Chain) organise de manière distribuée le positionnement des drones pour assurer le maintien d'un lien radio de bonne qualité entre les robots et un opérateur placé à l'entrée d'un environnement de type tunnel. La figure 6.1 illustre un usage de cet algorithme.



**FIGURE 6.1 – Illustration de l'algorithme U-Chain maintenant le lien radio dans un tunnel.**  
 Un opérateur explore le tunnel avec un drone (le drone le plus à droite ici). Quand le lien radio entre l'opérateur et le drone est trop faible (1), un nouveau drone décolle et se place automatiquement de manière à servir de relai radio (2). Le positionnement de ces robots comme relais permet l'exploration de l'environnement en assurant une bonne qualité radio. L'algorithme calcule la position des relais de manière à assurer une qualité de signal optimale (3).

U-Chain prend comme hypothèse que les robots naviguent dans un environnement comme un tuyau ou un tunnel, mais ne s'appuient pas sur un système de localisation pour estimer leur position le long de ce tuyau. La navigation demande bien entendu une estimation de la position au niveau local, en particulier de la distance aux parois, mais en dehors de cette navigation, notre algorithme ne s'appuie que sur le signal radio entre les différents robots pour déterminer comment positionner des robots qui servent de relais. Ceci permet que l'algorithme U-Chain s'adapte de manière naturelle à la topographie de l'environnement et qu'il puisse être embarqué à bord de petits robots avec de faibles capacités de calcul, comme c'est le cas pour le Crazyflie. Notre algorithme permet en particulier de placer plus de robots relais dans une zone où la propagation du signal radio serait perturbée pour une raison inconnue.

Je démontre que la maximisation de la qualité du réseau de communications entre les robots équivaut à une égalisation du signal radio entre deux drones consécutifs. Ceci permet de déduire un algorithme de positionnement régulant la qualité de signal entre les robots. Afin que cet algorithme puisse fonctionner efficacement nous présentons un filtre de Kalman qui permet de réduire le bruit de mesure de la qualité de signal en exploitant des mesures de la vitesse du drone par un capteur de flux optique. Nous présentons ensuite les résultats de cet algorithme en simulation et sur une chaîne de 3 Crazyflies dans un environnement réel.

Ce chapitre correspond à une traduction des travaux publiés dans LACLAU et al. 2021. Ma contribution personnelle à cet article consiste en la formulation précise du problème d'optimisation ainsi que la réalisation de la preuve de la correction et de la convergence de l'algorithme U-Chain.

Pierre Laclau a implémenté un simulateur permettant d'évaluer l'algorithme, le filtre estimant la qualité de signal et des capacités de communication pair à pair embarquées à bord du Crazyflie permettant d'estimer la qualité de signal radio nécessaire à l'algorithme. Il a aussi réalisé les expériences sur les robots réels. Jean-Baptiste Mouret, Franck Ruffier et Enrico Natalizio ont participé à la rédaction du papier.

## 6.2 Travaux connexes

La plupart des travaux concernant l'exploration d'environnements souterrains se sont concentrées sur des robots à roues ou à chenilles (MORRIS et al. 2006) couplés à un algorithme de SLAM construisant une carte de l'environnement (THRUN et al. 2004).

Cependant, les robots volants sont de plus en plus utilisés à l'intérieur de mines et montrent des résultats prometteurs (FREIRE et COTA 2017; PRESTON et ROY 2017; MANSOURI, KANELAKIS et al. 2020; JONES et al. 2019; MANSOURI, CASTAÑO et al. 2019).

Indépendamment des travaux spécifiques aux environnements souterrains, de nombreux algorithmes pour l'exploration par un groupe de robots ont été proposés (DORIGO et al. 2013; AMIGONI, BANFI et BASILICO 2017). Ces travaux traitent aussi bien d'algorithmes centralisés que décentralisés.

La plus grande partie de ces approches fait intervenir une estimation absolue ou relative de la position des robots (AMIGONI, BANFI et BASILICO 2017; YUANTENG PEI, MUTKA et NING XI 2010; CESARE et al. 2015; STUMP et al. 2011; NESTMEYER et al. 2015) qui est partagée entre les robots qui ont parfois déjà connaissance de l'environnement (STUMP et al. 2011; HSIEH et al. 2008).

L'utilisation d'un système de positionnement est bien plus facile pour des usages en extérieurs pour lesquels l'accès aux systèmes de géolocalisation type GPS est possible. En intérieur l'utilisation d'algorithmes de SLAM par chacun des robots d'un groupe permet aussi une localisation efficace.

Pour petits drones volants dans des environnements souterrains il est impossible d'utiliser les systèmes de géolocalisation, et les faibles capacités de calculs embarquées ne permettent pas l'utilisation d'algorithmes de SLAM.

Contrairement à la majorité des travaux sur le sujet, HAUERT, ZUFFEREY et FLOREANO 2009 présentent une approche s'appuyant sur un algorithme évolutionniste qui permet le maintien d'un réseau de communication sans utiliser d'informations de positionnement.

L'estimation de la qualité des liens radio est bien souvent trop peu fiable pour guider seule l'organisation d'un groupe de robots (AMIGONI, BANFI et BASILICO 2017). Cette faible fiabilité est la conséquence de l'influence d'un grand nombre de phénomènes inconnus et non modélisés sur la qualité du lien radio entre deux robots (AMIGONI, BANFI et BASILICO 2017; KHUWAJA et al. 2018).

Malgré ce manque de fiabilité, K. N. MCGUIRE et al. 2019 utilisent cette modalité pour guider l'exploration d'un bâtiment par des quadrotors Crazyflie. L'estimation de la qualité de signal est utilisée pour plusieurs tâches : le retour à la base via le suivi du gradient de la qualité du signal, l'évitement des autres robots et le choix des directions générales d'exploration. Ces travaux montrent que la qualité de signal peut être utilisée avec succès pour explorer, mais ils n'ont pas

pour objectif de maintenir la communication : il est supposé que chaque robot sera toujours en mesure de communiquer avec une station de base.

Dans ce chapitre, l'environnement sera supposé être à une dimension. Cela signifie que les robots n'auront de déplacement possible que dans une seule direction : vers l'avant ou vers l'arrière. Cette hypothèse simplifie la formulation du problème que doit résoudre l'algorithme chargé de maintenir les communications entre les robots et d'optimiser le placement de ceux-ci pour obtenir la meilleure qualité de signal possible. Cette hypothèse ne signifie pas que les robots sont réellement limités à des déplacements sur une dimension, mais plutôt que la navigation sur les axes autres que l'axe avant-arrière n'est pas gérée directement par l'algorithme U-Chain.

Une approche similaire de coordination d'un groupe de robot roulant basée sur la qualité du signal radio dans un environnement souterrain est présentée par RIZZO et al. 2013. Les auteurs conduisent une analyse théorique et expérimentale de la propagation du signal et ses caractéristiques. Ils décrivent une approche utilisant les caractéristiques identifiées pour organiser un groupe de robots. Cette approche parvient à résoudre le problème présenté dans l'article, mais nécessite une identification préalable de paramètres de l'environnement où sont déployés les robots. Cette identification est impossible en cas de déploiement dans un environnement inconnu. Par ailleurs, cette approche ne tente pas d'optimiser le placement des robots de manière à avoir la meilleure qualité de signal, mais seulement à assurer que le lien radio est d'une qualité suffisante.

### 6.3 Formulation du problème de placement des robots

On considère une chaîne de drones dans un tunnel. Ces drones ne possèdent pas de système capable de déterminer leur position le long de ce tunnel, mais peuvent se positionner localement par rapport aux parois du tunnel. L'objectif est de maintenir un lien radio entre chacun des drones de la chaîne de manière à ce que, par l'intermédiaire des autres robots, le premier drone de la séquence puisse communiquer avec le dernier.

Une première hypothèse importante sur l'environnement consiste à dire que seuls les déplacements vers l'avant et l'arrière sont considérés pour le problème de placement des robots. Les mouvements latéraux et verticaux sont gérés par un autre algorithme qui ne sera pas présenté dans ce chapitre, mais est détaillé dans LACLAU et al. 2021. Le contrôle de ces mouvements vers l'avant ou l'arrière est réalisé uniquement sur la base de l'estimation de la qualité de signal radio entre deux drones successifs. La métrique de qualité de signal utilisée est le RSSI (Received Signal Strength Indicator (SAUTER 2011)).

Le groupe de robots est organisé de manière séquentielle, en conséquence nous avons choisi de quantifier la qualité de l'ensemble des liens radio entre les robots par la qualité du lien le plus faible entre deux robots successifs dans cette séquence.

Ce lien le plus faible entre deux drones est le goulot d'étranglement des communications pour l'ensemble de la chaîne. Notre objectif est de maximiser la qualité de ce lien le plus faible pour optimiser la qualité d'une communication de bout en bout.

#### 6.3.1 Notations utiles

Pour formuler et prouver un algorithme optimisant la qualité du signal les notations suivantes seront utilisées :

- $\mathcal{C}$  : une courbe en deux dimensions décrivant la géométrie du tunnel dans lequel évoluent les robots ;
- $A = \{a_0, \dots, a_n\}$  : l'ensemble des drones ;
- $R = \{a_1, \dots, a_{n-1}\}$  : l'ensemble des drones servant de relais entre le drone de tête et une station de base ;
- $s_{\min}$  : une qualité de signal caractérisant le seuil à partir duquel un lien radio entre deux drones successifs n'est plus considéré comme suffisamment fiable (ici 60 pour les Crazyflie) ;
- $P = \{x_0, \dots, x_n\}$  : l'ensemble des abscisses curvilignes qui caractérisent la position des drones le long de la courbe du tunnel  $\mathcal{C}$  ;
- $s : (x_1, x_2) \mapsto s(x_1, x_2)$  : la fonction décrivant la qualité de signal entre deux drones placés aux positions  $x_1$  et  $x_2$ .

### 6.3.2 Hypothèses

Pour conduire la preuve de la convergence et de la correction de l'algorithme U-Chain nous posons plusieurs hypothèses sur l'environnement et les robots :

- chaque robot se positionne au centre du tunnel, équidistant de chaque paroi à l'aide d'un algorithme décrit dans LACLAU et al. 2021 ;
- $\forall i, j \in \{0, \dots, n\}^2, i > j \implies x_i \leq x_j$  : la position des drones est ordonnée dans l'ordre inverse des indices de ceux-ci, le drone de tête est d'indice 0 et le drone servant de station de base à l'entrée du tunnel est d'indice n ;
- la fonction  $s$  est continue et définie uniquement pour  $x_1 \leq x_2$ , ceci signifie que par convention la position la plus proche de l'entrée du tunnel est le premier argument de la fonction donnant la qualité de signal ;
- $s$  est une fonction strictement décroissante pour une comparaison au sens de l'inclusion :

$$[x_1, x_2] \subset [\tilde{x}_1, \tilde{x}_2] \implies s(x_1, x_2) > s(\tilde{x}_1, \tilde{x}_2)$$

(la qualité de signal décroît quand deux robots s'éloignent l'un de l'autre) ;

- $s$  est k-lipschitzienne :  $\exists k > 0$  tel que  $\forall x_1 \leq x_2, \forall \epsilon \in [0, x_2 - x_1]$  :
  - $s(x_1 + \epsilon, x_2) \leq s(x_1, x_2) + k \cdot \epsilon$
  - $s(x_1, x_2 - \epsilon) \leq s(x_1, x_2) + k \cdot \epsilon$
- les temps de communication entre drones sont négligés (le temps de transmission des paquets (0.25ms) est petit devant la vitesse des Crazyflies).

### 6.3.3 Configurations optimales

Une configuration optimale  $x^*$  est un ensemble de positions qui maximisent la qualité des liens, c'est-à-dire vérifiant :

$$x^* = \operatorname{argmax}_{x_i} \left( \min_i (s(x_{i+1}, x_i)) \right) \quad (6.1)$$

ces positions maximisent la qualité du lien le plus faible de la chaîne. Ceci correspond à la classique règle de décision “max-min” (JAFFE 1981).

Proposition 6.1: Si la fonction  $s$  est continue alors la configuration optimale  $x^*$  vérifie

$$\forall (i, j) \in \{0, \dots, n-1\} \quad s(x_{i+1}^*, x_i^*) = s(x_{j+1}^*, x_j^*) \quad (6.2)$$

Démonstration 6.1: L'idée qui structure cette preuve est qu'une configuration avec des liens de qualité inégale peut être améliorée en déplaçant d'une distance  $\epsilon$  un drone à la jonction de deux liens de qualité inégale de manière à ce que le lien de plus mauvaise qualité soit un peu meilleur, mais que le lien de meilleure qualité reste d'une qualité suffisante. C'est la continuité de  $s$  qui assure qu'il existe une distance  $\epsilon$  assez petite pour assurer que le lien le plus fort ne soit pas trop dégradé. La configuration obtenue après déplacement est meilleure que la précédente. Une configuration optimale ne peut être améliorée, elle doit donc posséder des liens de qualité égale.

Plus formellement, procédons par l'absurde en supposant l'existence d'une configuration optimale  $x^*$  dont les liens ne sont pas de qualité égale, c'est à dire qui ne vérifie pas 6.2 et ainsi

$$\exists j, s(x_{j+1}^*, x_j^*) < s(x_j^*, x_{j-1}^*), \text{ ou } s(x_{j+1}^*, x_j^*) < s(x_{j+2}^*, x_{j+1}^*) \quad (6.3)$$

Sans perte de généralité on peut considérer que le premier cas s'applique. Montrons par récurrence descendante sur le nombre de liens de  $x^*$  qui sont de qualité minimale qu'il existe une meilleure configuration.

S'il existe un seul lien de qualité minimale (il en existe au moins un), notons les indices des drones bordant ce lien  $j$  et  $j + 1$ . Par continuité de  $s$ ,  $\exists \epsilon > 0$ ,

$$s(x_{j+1}^*, x_j^*) < s(x_{j+1}^*, x_j^* - \epsilon) < s(x_j^* - \epsilon, x_{j-1}^*) < s(x_j^*, x_{j-1}^*) \quad (6.4)$$

Si  $j$  est l'indice du drone d'une des extrémités il est possible de déplacer  $j + 1$  à la place pour obtenir un résultat similaire. Comme le lien entre  $j$  et  $j + 1$  était le seul lien de qualité minimale, la nouvelle configuration dont les positions sont données par  $x_i = x_i^*$  si  $i \neq j$  et  $x_j = x_j^* - \epsilon$  est une meilleure configuration que  $x^*$  puisque le lien de qualité minimale est maintenant meilleur et que le seul autre lien qui a été dégradé (entre  $x_j$  et  $x_{j-1}$ ) reste meilleur que le lien entre  $x_j$  et  $x_{j+1}$  et n'est donc pas le pire lien.  $x^*$  étant censée être une configuration optimale, il est absurde de pouvoir l'améliorer. Ceci conclut l'étape d'initialisation.

S'il existe plus d'un lien de qualité minimale, il est possible de réaliser la même transformation que dans le cas à un seul lien minimal. Ceci réduit de 1 le nombre de liens de qualité minimale et il est possible d'appliquer de nouveau la transformation jusqu'à ce que la configuration ne contienne plus qu'un seul lien de qualité minimale.

Soit  $j$  et  $j + 1$  les indices de deux robots formant un de ces liens de qualité minimale. On peut supposer que l'un des liens voisins de  $(j, j + 1)$  n'est pas de qualité minimale. En effet, si tous les liens de qualité minimale étaient bordés par d'autres liens de qualité minimale, tous les liens seraient de qualité minimale et seraient donc égaux. Or  $x^*$  contient des liens de qualité inégale.

On peut donc appliquer sur le lien  $(j, j + 1)$  la transformation décrite dans l'étape d'initialisation qui réduit de 1 le nombre de liens de qualité minimale.

On a ainsi prouvé par récurrence qu'il était possible de construire une configuration meilleure que la configuration optimale  $x^*$ . Ceci permet de déduire que toute configuration optimale a ses liens égaux.  $\square$

Ce résultat prouve une condition nécessaire sur l'égalité des liens d'une configuration optimale. Je prouverais plus loin que cette condition est aussi suffisante.

Il est important de noter que l'égalité des qualités de lien ne signifie absolument pas que les distances entre les drones sont elles aussi égales. La qualité des liens peut en effet être affectée par la géométrie ou la composition locale de l'environnement.

Comme la fonction  $s$  est continue, la fonction qui à une configuration associe la qualité de son lien de qualité minimale est elle aussi continue. L'ensemble de toutes les configurations est un ensemble compact (fermé et borné et dimension finie), il existe donc une configuration optimale, par continuité de  $s$  sur un ensemble compact.

Notons  $s_{eq}$  la qualité de signal des liens d'une configuration dont tous les liens sont de qualité égale. Une telle configuration existe puisque la configuration optimale en particulier a tous ses liens égaux et celle-ci existe.

Proposition 6.2: Étant donné une position initiale et une position finale fixée dans le tunnel et un nombre de drones fixé, si  $s$  est une fonction décroissante (i.e.  $[x_1, x_2] \subset [\tilde{x}_1, \tilde{x}_2] \Rightarrow s(x_1, x_2) > s(\tilde{x}_1, \tilde{x}_2)$ ) alors toute configuration dont les liens sont de qualité égale est une configuration optimale, solution de l'équation 6.1.

La preuve de ce résultat n'est pas évidente car peu d'hypothèses sont faites sur la fonction de qualité de signal  $s$ . Pour montrer que toutes les configurations avec des liens de qualité égale sont optimales, je vais prouver qu'il n'existe qu'une seule configuration dont les liens sont de qualité égale.

Démonstration 6.2: (Par l'absurde.) Supposons qu'il existe deux configurations distinctes  $C$  et  $\tilde{C}$  dont tous les liens sont de qualité égale :

$$\begin{cases} \forall i < n \quad s(x_{i+1}, x_i) = s_{eq} \\ x_0 = x_t \\ x_n = 0 \end{cases} \quad (6.5) \qquad \begin{cases} \forall i < n \quad s(\tilde{x}_{i+1}, \tilde{x}_i) = \tilde{s}_{eq} \\ \tilde{x}_0 = x_t \\ \tilde{x}_n = 0 \end{cases} \quad (6.6)$$

Montrons d'abord que la qualité de lien de ces deux configurations est différente. Soit  $i$  le premier indice tel que  $x_i \neq \tilde{x}_i$ . Cet indice existe sinon  $C$  et  $\tilde{C}$  seraient identiques. On peut en particulier supposer que  $x_i < \tilde{x}_i$ . Comme  $i$  est le premier indice pour lequel les positions diffèrent,  $x_{i-1} = \tilde{x}_{i-1}$ . On a ainsi :

$$s_{eq} = s(x_i, x_{i-1}) = s(x_i, \tilde{x}_{i-1}) > s(\tilde{x}_i, \tilde{x}_{i-1}) = \tilde{s}_{eq} \quad (6.7)$$

Et en particulier :

$$s_{eq} > \tilde{s}_{eq} \quad (6.8)$$

Prouvons maintenant par récurrence descendante sur les indices des robots  $i$  que toutes les positions des robots de  $\tilde{C}$  sont plus éloignées de l'origine que celles de  $C$  :

$$\forall i \in [0, n[, \quad x_i < \tilde{x}_i \quad (6.9)$$

En particulier on aurait  $x_0 < \tilde{x}_0$  ce qui contredit les hypothèses de positions extrêmes fixées et permettrait de conclure qu'il n'existe pas deux configurations dont les liens sont de qualité égale.

Comme les positions extrêmes sont fixées, on a  $x_n = \tilde{x}_n$ . Si  $x_{n-1} \geq \tilde{x}_{n-1}$  alors  $[\tilde{x}_n, \tilde{x}_{n-1}] \subseteq [x_n, x_{n-1}]$  et comme  $s$  est décroissante :

$$\tilde{s}_{eq} = s(\tilde{x}_n, \tilde{x}_{n-1}) \geq s(x_n, x_{n-1}) = s_{eq} \quad (6.10)$$

ce qui contredit l'équation 6.8 et permet de déduire que

$$x_{n-1} < \tilde{x}_{n-1} \quad (6.11)$$

et conclut l'étape d'initialisation.

Supposons maintenant  $x_i < \tilde{x}_i$  pour un  $i > 0$ . Si on a  $x_{i-1} \geq \tilde{x}_{i-1}$  alors  $[\tilde{x}_i, \tilde{x}_{i-1}] \subset [x_i, x_{i-1}]$  et donc

$$\tilde{s}_{eq} = s(\tilde{x}_i, \tilde{x}_{i-1}) \geq s(x_i, x_{i-1}) = s_{eq} \quad (6.12)$$

ce qui contredit l'équation 6.8 et permet de déduire que

$$x_{i-1} < \tilde{x}_{i-1} \quad (6.13)$$

Le principe de récurrence permet de conclure sur la propriété 6.9 et de conclure la preuve de l'unicité d'une configuration dont les liens sont de qualité égale.

En plus de prouver que l'égalité de la qualité des liens est une condition suffisante pour qu'une configuration soit optimale, cette preuve montre que le nombre de robots et les positions des robots aux extrémités déterminent directement la position de chacun des drones intermédiaires ainsi que la qualité de signal des liens.  $\square$

Le problème d'optimisation était pour le moment formulé avec un nombre fixe de drones. Pour explorer un environnement, nous reformulons ce problème d'optimisation afin d'utiliser un nombre minimal de drones tout en assurant une qualité de signal la meilleure possible et au dessus du seuil  $s_{min}$  pour un environnement et des positions des drones extrêmes données :

$$n^* = \operatorname{argmin} n$$

satisfaisant :

$$\begin{cases} \forall i, j \in \{0, \dots, n-1\}^2 \quad s(x_{i+1}, x_i) = s(x_{j+1}, x_j) \\ \forall i \in \{0, \dots, n-1\} \quad s(x_{i+1}, x_i) \geq s_{min} \\ x_n = 0 \\ x_0 = x_t \end{cases} \quad (6.14)$$

Imposer la propriété d'égalité des qualités de signal assure que la qualité du lien le plus faible est toujours optimale. La contrainte d'une qualité meilleure que le seuil  $s_{min}$  est imposée pour éviter les pertes de signal le long de la chaîne de drones. La démonstration 6.2 montre le compromis entre la qualité de lien et le nombre de robots pour des positions extrêmes fixées. Cette nouvelle formulation ne fixe plus le nombre de robots et tente de déterminer la quantité minimale de robots de manière à assurer une qualité de lien à même de garantir la stabilité des communications entre les robots.

## 6.4 Algorithme de positionnement le long du tunnel

Notre objectif est d'obtenir un algorithme qui minimise le nombre de drones comme formulé par l'équation 6.14.

Nous montrons que :

- étant donné un nombre fixe de drones, égaliser les qualités de signal de manière décentralisée sur chaque robot est en fait un algorithme qui optimise la qualité de lien (6.4.1) ;
- une estimation de la qualité du signal peut être obtenue par le filtrage du RSSI par un filtre de Kalman tirant parti de mesures de flux optique ;
- les robots supplémentaires doivent décoller quand la qualité de signal passe sous le seuil minimal afin de garantir que le nombre minimal de robot soit utilisé.

### 6.4.1 Convergence vers la configuration optimale

Nous avons posé l'hypothèse que l'algorithme U-Chain ne contrôle les robots que sur un degré de liberté, le déplacement vers l'avant ou l'arrière. Nous considérons avec cette hypothèse que la stabilisation sur les autres axes est gérée par un autre algorithme réactif décrit dans LACLAU et al. 2021. Cet algorithme maintient et stabilise les robots au centre du tunnel, à égale distance des parois.

Supposons que les robots sont positionnés dans une configuration sous optimale du point de vue de la qualité des liens radio. Pour avoir une configuration optimale il faut et il suffit que chaque robot ait un lien radio de qualité égale avec les robots qui le précédent et le suivent. Pour atteindre cette configuration initiale, chaque drone va comparer la qualité des liens avec ses deux voisins et se déplacer en avant ou en arrière de manière à renforcer la qualité du lien le plus faible sans trop dégrader la qualité du lien le plus fort. L'algorithme 7 détaille la manière dont chaque robot agit. L'amplitude du déplacement pour ajuster la qualité des liens dépend de la qualité des liens à ajuster. Les robots doivent éviter de bouger trop vite afin de ne pas créer un phénomène d'oscillations. Dans la suite de cette sous section,  $k$  représente le coefficient de Lipschitz de la fonction de qualité de signal  $s$  (cf. 6.3).

---

**Algorithme 7** Version simplifiée de l'algorithme U-Chain pour le robot d'indice  $i$

---

```
while Les liens ne sont pas de qualité égale do
     $s_d = s(x_{i+1}, x_i) - s(x_i, x_{i-1})$ 
    Déplacement du robot d'une distance  $\epsilon_i = \frac{s_d}{3k}$ 
end while
```

---

Proposition 6.3: L'algorithme 7 fait converger les positions des robots vers une configuration dont tous les liens sont égaux.

Démonstration 6.3: Nous avons montré avec la démonstration 6.1 qu'une configuration non optimale peut être améliorée en bougeant le drone d'une petite distance  $\epsilon$ . L'algorithme 7 réalise ceci de manière distribuée : chaque agent dont les liens ne sont pas de qualité égale se déplace pour égaliser la qualité de ces liens. Ces déplacements sont réalisés à bord de chaque drone de manière indépendante, contrairement à la démonstration 6.1 qui procède de manière séquentielle. Il est nécessaire de s'assurer que l'ensemble des contributions individuelles simultanées des drones améliore bien la configuration du point de vue global.

Pour vérifier ceci, il suffit de vérifier qu'un lien de bonne qualité qui serait affaibli par le mouvement des deux robots qui le composent demeure de meilleure qualité que le lien de qualité

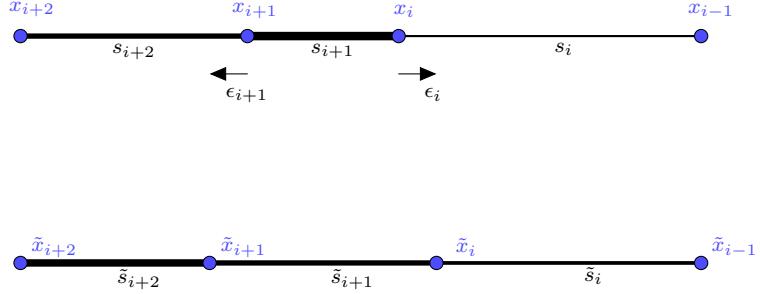


FIGURE 6.2 – Illustration d'une étape de l'algorithme 7 dans le cas où un lien qui n'est pas minimal est voisin de deux liens de qualité plus faible. La qualité d'un lien est représentée par l'épaisseur du trait entre deux positions.

la plus faible avant le mouvement, et que le lien de qualité la plus faible devient plus fort.

On note  $\epsilon_i$  le déplacement du robot d'indice  $i$ ,  $s_i = s(x_{i+1}, x_i)$  la qualité du lien entre les robots d'indice  $i$  et  $i + 1$  avant le mouvement de ceux-ci et  $\tilde{x}_i = x_i + \epsilon_i$  la position du robot  $i$  après son mouvement et  $\tilde{s}_i = s(x_{i+1} + \epsilon_{i+1}, x_i + \epsilon_i)$  la qualité du lien après les mouvements. La figure 6.2 illustre ces notations.

Je commence par prouver qu'un lien qui n'est pas un des liens les plus faibles n'est pas trop diminué après que l'ensemble des robots se soient déplacés. Soient  $s_{i-1}, s_i, s_{i+1}$  trois liens tels que  $s_i$  soit le lien le plus fort des trois avant le déplacement des robots :  $s_i > s_{i-1}, s_{i+1}$ . On a :

$$\epsilon_i = \frac{s_i - s_{i-1}}{3k}, \quad \epsilon_{i+1} = \frac{s_{i+1} - s_i}{3k} \quad (6.15)$$

et grâce au fait que  $s$  est  $k$ -lipschitzienne (croissance bornée par  $k$ ) :

$$\begin{aligned} s_i &= s(x_{i+1}, x_i) \\ &= s(x_{i+1} + \epsilon_{i+1} - \epsilon_{i+1}, x_i + \epsilon_i - \epsilon_i) \\ &= s(\tilde{x}_{i+1} + (-\epsilon_{i+1}), \tilde{x}_i - \epsilon_i) \\ &\leq \tilde{s}_i + k(-\epsilon_{i+1} + \epsilon_i) \\ &\leq \tilde{s}_i + k \left( \frac{s_i - s_{i+1} + s_i - s_{i-1}}{3k} \right) \end{aligned} \quad (6.16)$$

On en déduit :

$$\begin{aligned} \tilde{s}_i &\geq s_i - k \left( \frac{s_i - s_{i+1} + s_i - s_{i-1}}{3k} \right) \\ &\geq \frac{s_{i+1} + s_{i-1} + s_i}{3} \\ &> \min(s_{i+1}, s_{i-1}) \end{aligned} \quad (6.17)$$

Cette inégalité montre que le lien fort n'a pas été assez affaibli pour que sa qualité devienne inférieure à la qualité de chacun des liens voisins avant le déplacement. Cela signifie en particulier que la qualité de ce lien après le déplacement est meilleure que la qualité du lien le plus faible avant le déplacement.

Montrons ensuite que la qualité du lien le plus faible a été améliorée.

Soit  $s_i$  la qualité d'un des liens les plus faibles de la configuration (il peut exister plusieurs liens qui ont tous la qualité la plus faible). Comme  $s_i$  est minimale, cette qualité de lien est plus

faible ou égale à celle des liens voisins :  $s_i \leq s_{i-1}, s_{i+1}$ . Les deux robots qui composent ce lien ne peuvent donc se déplacer que de manière à renforcer ce lien. Aucun lien avec la qualité la plus faible n'est donc affaibli.

Si un des voisins de  $s_i$  n'est pas de qualité minimale, alors  $s_i$  sera strictement améliorée par le mouvement d'un des drones qui composent le lien correspondant. La qualité des liens de qualité minimale qui sont entourés de liens eux aussi de qualité minimale ne change pas. Si la configuration est non optimale, au moins un des liens de qualité minimale est voisin d'un lien de qualité non minimale. En conséquence, le nombre de liens de qualité minimale diminue à chaque étape ou alors la qualité du lien le plus faible augmente (quand chaque lien de qualité minimale est voisin d'au moins un lien de qualité non minimale).

La suite des configurations des positions lors de l'exécution de l'algorithme est donc une suite croissante de configurations. Toute suite de configurations est bornée par la configuration optimale (puisque celle-ci est optimale). On peut en déduire que la suite des configurations converge, et la seule configuration qui reste stable après une étape de l'algorithme 7 est la configuration optimale. Ceci permet de conclure que la suite des configurations converge vers la configuration optimale.  $\square$

Pour assurer la convergence, il est nécessaire de calculer pour chaque robot  $a_i$  la distance de déplacement  $\epsilon_i$ . Cette distance dépend de la mesure de la qualité de signal sur les liens auxquels participe un robot ainsi que du coefficient de Lipschitz  $k$  de  $s$ .

Cependant, la preuve de convergence de l'algorithme prouve que si le robot  $a_i$  se déplace d'une distance d'au plus  $\epsilon_i$  dans la bonne direction, il y a quand même convergence. Ceci permet de se passer d'une mesure exacte de la qualité de signal et de  $k$  et de se contenter d'une borne supérieure sur la distance de déplacement.

C'est parce que la configuration qui maximise la qualité du lien le plus faible est la configuration pour laquelle tous les liens sont de qualité égale et que cette configuration est unique que les drones peuvent parvenir à un consensus. Prouver ce consensus en faisant intervenir la dynamique du système (WEI, BEARD et ATKINS 2007) nécessiterait la formulation d'une équation exacte d'évolution de  $s_i$  à chaque étape. À défaut d'une expression exacte de l'évolution de  $s_i$  suite à un mouvement  $\epsilon_i$ , supposer que  $s$  est bi-lipschitzienne (croissance comprise entre  $0 < k_1\epsilon$  et  $k_2\epsilon$ ) permettrait de formuler deux dynamiques qui décriraient l'évolution de  $s_i$  à vitesse maximale et minimale. Ces deux dynamiques pourraient permettre de borner l'évolution véritable de  $s_i$  et d'adapter la preuve de convergence en utilisant les outils de la théorie du consensus (WEI, BEARD et ATKINS 2007).

#### 6.4.2 Estimation du RSSI par un filtre de Kalman

L'algorithme 7 prend ses décisions et calcule les distances de déplacement de chaque robot sur la base de mesures de la qualité du signal entre deux robots. Cette qualité de signal est mesurée à l'aide du RSSI (Received Signal Strength Indicator (SAUTER 2011)). Ces mesures sont malheureusement entachées de bruit et peu fiables. La qualité du signal est impactée par de nombreux facteurs outre la distance entre l'émetteur et le récepteur, en particulier l'orientation du drone, l'activité de ses rotors, l'activité radio des autres drones, les rebonds sur l'environnement, etc. (KHUWAJA et al. 2018; K. N. MCGUIRE et al. 2019).

Il est donc nécessaire de filtrer le RSSI pour enlever les bruits à haute fréquence. L'utilisation d'un filtre à moyenne glissante n'est pas satisfaisante en raison de l'introduction de retards dans l'estimation de la qualité de signal qui pourraient empêcher la convergence de l'algorithme.

Il est important d'observer que nous essayons de réduire autant que possible la durée entre une mesure de la qualité de signal (changement de la valeur du RSSI) et la décision de faire se déplacer un des robots afin d'augmenter la vitesse de convergence des robots vers la configuration optimale.

Les délais dus aux communications entre robots sont négligés pour plusieurs raisons : peu de messages sont échangés entre les robots relativement à la bande passante (jusqu'à 1Mbps pour un Crazyflie), il faut moins de 0.25ms pour envoyer un paquet et recevoir une réponse. Cette durée est très faible par rapport à la dynamique individuelle des drones (inférieure à 0.3m/s pour les expériences de ce chapitre) et celle de l'ensemble. Il serait possible d'obtenir de meilleures performances pour les communications en optimisant le nombre de paquets, leur contenu et la fréquence des échanges et en utilisant des canaux différents pour chaque communication.

Les hypothèses que nous utilisons sont que la qualité de signal baisse quand deux robots s'éloignent l'un de l'autre et que la vitesse relative entre deux drones voisins peut être mesurée pour un coût en calcul faible à l'aide un capteur de flux optique de très petite taille (RUFFIER et al. 2003). Par ailleurs, les télémètres laser embarqués sont utilisés pour assurer un vol à une attitude constante, ce qui permet d'exploiter les mesures de flux optique pour estimer la vitesse par rapport au sol. Les capteurs de flux optique peuvent aussi fonctionner dans des environnements peu éclairés (MAFRICA, SERVEL et RUFFIER 2016). Les capteurs de flux optique récents sont développés à partir des capteurs des souris optiques et ne pèsent que quelques milligrammes.

Nous intégrons ces hypothèses pour formuler un modèle de dynamique et d'observation d'un filtre de Kalman à une dimension estimant la qualité de signal. Les travaux de BULTEN, VAN ROSSUM et HASELAGER 2016 utilisent des filtres de Kalman pour la localisation d'individus ou de téléphones avec l'hypothèse que les individus ne se déplacent pas entre deux mesures. PAUL et WAN 2009 réalisent une tâche similaire en combinant des capteurs externes aux mesures de RSSI.

Le filtre de Kalman suppose les modèles de dynamique et d'observations suivants :

$$r_t = r_{t-1} + Au_t + \nu_t \quad (6.18)$$

$$z_t = r_t + \delta_t \quad (6.19)$$

où  $r_t$  est la quantité estimée (le RSSI) et  $u_t$  est la vitesse relative entre les deux robots dont le RSSI est estimé à l'instant  $t$ .  $\nu_t \sim \mathcal{N}(0, Q)$  est un bruit intrinsèque du RSSI dont on suppose connue la distribution.  $A$  représente l'influence de la vitesse relative sur l'évolution du RSSI.  $z_t$  est l'observation du RSSI, c'est-à-dire la mesure du RSSI qui est la somme du véritable RSSI et d'un bruit de mesure  $\delta_t \sim \mathcal{N}(0, R)$ . Les équations de mise à jour et de correction du filtre de Kalman sont présentées dans le chapitre 5.6.

Dans la simulation comme pour les tests réels, la valeur de  $A$  a été ajustée de manière à minimiser les retards dans l'estimation et à réduire la variance de l'estimation.

L'algorithme 8 détaille le comportement suivi par chaque robot de la chaîne et combine l'égalisation (section 6.4.1) des liens à l'estimation du RSSI (section 5.6). Cet algorithme se résume à

obtenir une mesure de la qualité des liens avec les drones en amont et en aval, mettre à jour une estimation de cette qualité à l'aide de la mesure et d'un filtre de Kalman, puis à commander au drone de se déplacer vers l'avant, l'arrière ou de rester sur place en fonction de la différence entre les deux RSSI estimés.

La fonction  $Kalman_{A,R}$  est le filtre de Kalman,  $r_b$ ,  $r_f$  sont les valeurs estimées pour les RSSI des liens avec les drones précédent et suivant et  $Q_b$ ,  $Q_f$  les incertitudes associées à ces estimations. Les estimations  $r_b$  et  $r_f$  sont initialisées avec les premières mesures et  $Q_b$ ,  $Q_f$  par Q.

---

**Algorithme 8** Algorithme de positionnement de chaque drone  $i \in R$  (les drones des extrémités sont exclus).

---

```

while True do
     $z_b = \text{mesureRSSI}(i + 1, i)$ 
     $z_f = \text{mesureRSSI}(i, i - 1)$ 
     $r_b, Q_b = \text{Kalman}_{A,R}(r_b, z_b, \dot{x}_{i+1} - \dot{x}_i, Q_b)$ 
     $r_f, Q_f = \text{Kalman}_{A,R}(r_f, z_f, \dot{x}_i - \dot{x}_{i-1}, Q_f)$ 
     $r_{\text{diff}} = r_b - r_f$ 
    if  $|r_{\text{diff}}| > T$  then
         $\epsilon_i = \frac{r_{\text{diff}}}{3k}$ 
    else
         $\epsilon_i = 0$ 
    end if
    déplacer le drone de  $\epsilon_i$ 
end while
```

- La valeur de  $k$  peut être déduite en inversant une expression de  $r$  de manière à ce qu'un déplacement de  $\epsilon_i$  du robot  $a_i$  change  $r_{i+1,i}$  d'au plus  $\frac{r_{i,i-1} - r_{i+1,i}}{3}$  comme décrit dans 6.3.
  - Le seuil  $T$  (tolérance) est introduit pour éviter les instabilités numériques dans les calculs de différences de RSSI.
- 

#### 6.4.3 Exploration avec une chaîne de drones

Une utilisation immédiate de l'algorithme 8 de positionnement consiste à déplacer le premier drone (le plus avancé dans le tunnel) et à attendre que la chaîne de drones s'auto-organise pour maximiser la qualité de signal. L'avancée du drone de tête étire la chaîne et pourrait dégrader la qualité de signal de la chaîne sous le seuil  $s_{\min}$ . Pour éviter que la qualité de signal ne passe sous ce seuil, si la station de base (à l'entrée du tunnel) détecte une qualité de lien inférieure à  $s_{\min}$  elle commande le décollage d'un nouveau drone qui vient s'ajouter à la chaîne et permet à la qualité de signal de s'améliorer.

Nous supposons ici que la vitesse de convergence de l'algorithme qui égalise les qualités de signal est suffisante pour que ces qualités de signal soient égales au moment du décollage du nouveau drone. Dans ce cas, si la qualité de signal entre la station de base et le drone suivant est inférieure à  $s_{\min}$  cela signifie que toutes les qualités de signal sont aussi inférieures au seuil. En cas de convergence rapide de l'algorithme de positionnement il y a donc un décollage d'un nouveau drone si et seulement si la qualité minimale de signal ne peut être maintenue dans la chaîne. Ceci assure qu'un nombre minimal de drones est utilisé pour l'exploration.

S'il ne reste plus de drones disponibles pour servir de relais, l'algorithme maintient toujours une qualité de signal égale entre les différents drones, mais cette qualité de signal peut devenir inférieure au seuil minimal de fiabilité des communications. De plus, si le drone de tête se déplace trop rapidement la connexion peut être rompue durant son déplacement (par exemple en cas d'entrée dans une zone où la propagation du signal est plus difficile). Dans le cas d'une perte de connexion, tous les drones se déplacent vers l'entrée du tunnel tant que leur signal avec le drone précédent est trop faible ou rompu.

## 6.5 Résultats

### 6.5.1 En simulation

#### 6.5.1.1 Hypothèses de travail

- La fréquence d'exécution de l'algorithme de positionnement et de l'estimation du RSSI est de 5Hz pour assurer une implémentation assez légère pour fonctionner à bord du Crazyflie.
- L'environnement de simulation est en deux dimensions et la gestion de la distance aux parois et de l'orientation est gérée par un contrôleur réactif décrit dans LA CLAU et al. 2021.
- Les communications sont instantanées.

#### 6.5.1.2 Modèle de propagation du signal

Nous avons choisi de modéliser le RSSI selon la formulation de VINOGRADOV et POLLIN 2019 :

$$RSSI(x_1, x_2) = 10 \times \alpha(x_1, x_2) \times \log(|x_1 - x_2|) + \epsilon \quad (6.20)$$

$\alpha$  est un facteur d'atténuation de l'environnement et indique une difficulté de propagation locale, qui varie habituellement entre 2 et 6 selon l'environnement.  $\epsilon \sim \mathcal{N}(0, 3)$  représente le bruit de mesure gaussien. Nous avons ici utilisé une expression du RSSI qui est croissante contrairement à l'hypothèse faite sur la qualité de signal. Pour y adapter les résultats présentés précédemment il suffit de prendre l'opposé de cette expression.

Sur la figure 6.3 on observe que le RSSI varie avec le temps. Cette variation dans le temps provient de deux sources, la première est le bruit de mesure qui varie avec le temps. La seconde est la variation de la position des drones avec le temps. Nous avons estimé que ces causes de variation justifiaient de ne pas modéliser une dépendance directe au temps dans la fonction de qualité de signal. La simulation comprend une probabilité de 20% de perdre un paquet, en accord avec ce qui est observé pour des Crazyflies.

Nous avons observé expérimentalement que la qualité de signal décroissait quand la ligne de jonction de deux drones passait à travers leurs moteurs, dont l'activité interfère avec le signal radio. Le RSSI dépend donc de l'orientation relative des drones en plus de la distance entre ceux-ci. Cet effet est mal documenté et pourrait être spécifique à la position des antennes et au modèle des moteurs, nous avons donc décidé d'ignorer cet effet dans la simulation. De futurs travaux pourraient étendre le modèle de propagation du signal décrit par l'équation 6.20 pour tenir compte de l'orientation relative des robots, mais nécessiteraient des efforts importants pour identifier les paramètres d'un tel modèle.

### 6.5.1.3 Résultats

Nous avons d'abord vérifié la convergence vers des liens de qualité égale pour un nombre fixé de drones dans des environnements différents. Les robots sont placés à des positions aléatoires dans trois environnements. La position des deux drones aux extrémités de la chaîne est fixée. La figure 6.3-A-C montre que les drones convergent vers une configuration à liens de qualité égale malgré le bruit de mesure. Nous avons confirmé ces résultats par 30 expériences indépendantes dans chacun des trois environnements. Dans chacune de ces 30 expériences les drones ont convergé vers une position égalisant les qualités de signal.

Nous avons ensuite évalué l'utilité de l'estimation du RSSI par un filtre de Kalman dans un contexte où le drone de tête est mobile. Nous avons simulé une exploration en plaçant initialement tous les drones à l'entrée du tunnel et en faisant décoller le premier drone depuis ce point. Ce premier drone se déplace à vitesse constante ( $0.2m.s^{-1}$ ) en s'éloignant de l'entrée du tunnel pendant 50s.

L'utilisation du signal brut, sans filtration du bruit de mesure, comme entrée de l'algorithme de positionnement permet aux drones de converger vers une position qui égalise les qualités de signal (cf. figure 6.3-E). Une mesure anormalement haute (à cause du bruit) peut cependant déclencher le décollage d'un drone non nécessaire (cf. figure 6.3-D à 80s pour la figure  $T_0$ ).

Par ailleurs, après que les drones aient convergé vers une position égalisant les qualités de signal, ceux-ci continuent à osciller autour de cette position avec une amplitude importante (cf. figure 6.3-F). Les figures E et F ne tiennent compte que des 3 premiers liens (suffisant pour maintenir les communications) pour calculer la variance afin d'éviter de prendre en compte les lancements superflus dans cette variance. Ces oscillations pourraient poser des problèmes de stabilité sur les robots réels.

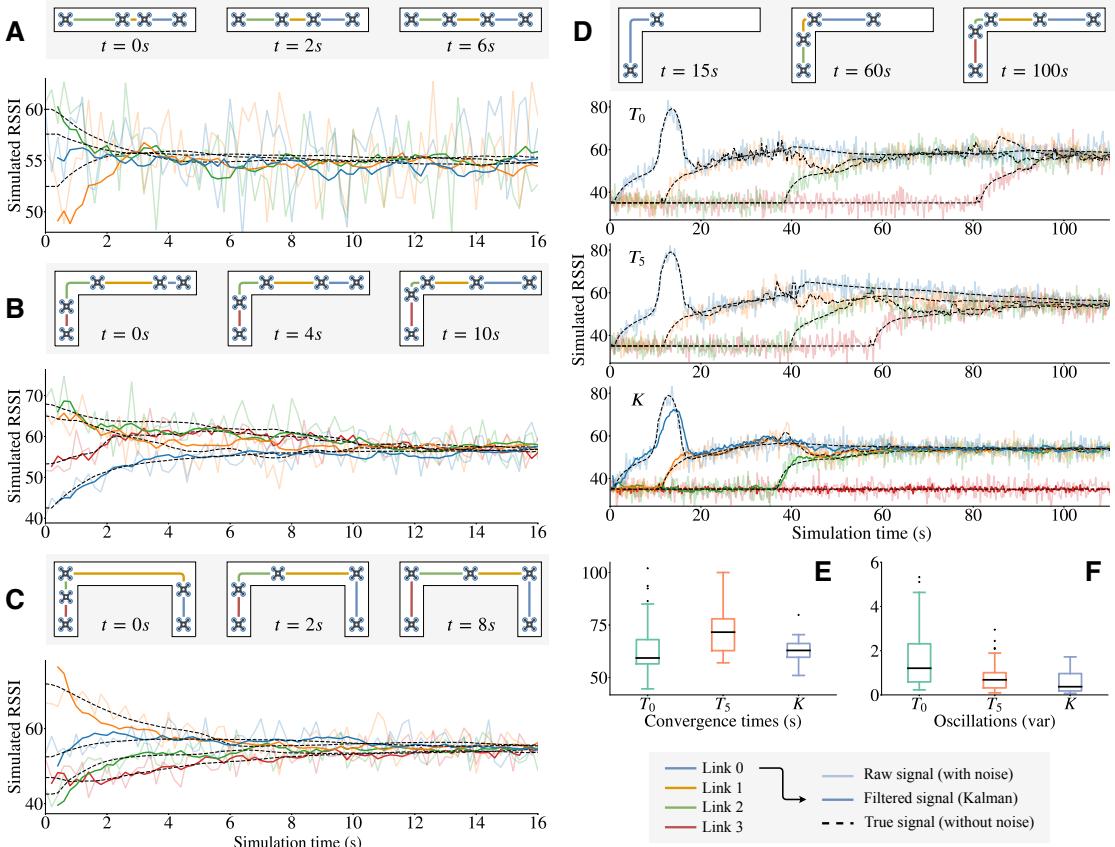
Pour prévenir ces oscillations, nous avons évalué l'introduction d'un seuil de tolérance sur la différence de signal au dessous duquel il n'y a pas de mouvement (méthode  $T_5$ ). La figure 6.3-F montre que l'introduction de ce seuil réduit effectivement les oscillations autour des positions qui égalisent les liens. La figure 6.3-E montre cependant que cette approche diminue la vitesse de convergence de manière importante. Ceci s'explique par le fait que les drones attendent que la différence de qualité de signal dépasse le seuil  $T$  avant d'initier un déplacement.

L'utilisation d'un filtre de Kalman (méthode  $K$  sur la figure 6.3-D-E-F) améliore aussi bien la vitesse de convergence et la stabilité autour des positions qui égalisent les qualités de signal.

### 6.5.2 Dans un environnement réel

Nous avons conçu un circuit qui ressemble à un tunnel et qui est présenté par la figure 6.4-A. Les parois sont constituées de carton pour une installation simplifiée. Ces parois ne bloquent que peu la propagation du signal, nous avons dû choisir avec précaution la forme du tunnel pour assurer la propriété de monotonicité.

Les parois de l'environnement empêchent les drones de progresser en ligne droite les uns vers les autres pour améliorer la qualité de signal et simulent ainsi une propagation du signal non linéaire et complexe. Si cet environnement ne permet pas de reproduire toutes les propriétés d'un environnement souterrain, il permet de valider l'algorithme de positionnement, les décisions de décollage, la stabilité globale de la chaîne ainsi que le contrôleur qui stabilise le drone à égale distance des parois.



**FIGURE 6.3 – Résultats en simulation.** Pour chaque figure, la véritable valeur du RSSI obtenue à l'aide de l'équation 6.20 et sans le bruit de mesure est présentée en ligne pointillée. Le signal brut (ligne pleine pâle) est l'entrée de l'algorithme U-Chain. Le signal filtré (ligne pleine saturée) est utilisé pour prendre les décisions d'avancer ou de reculer. **A-C.** Convergence de l'algorithme 8 dans différents environnements présentés sous la figure. Les drones se placent à des positions adaptées pour servir de relais. **D.** comparaison entre différentes méthodes d'estimation du RSSI :  $T_0$  correspond à une utilisation du signal brut,  $T_5$  pour une tolérance de 5 et  $K$  pour l'utilisation du filtre de Kalman. **E.** Comparaison des temps de convergence pour 30 répliques indépendantes de ces trois méthodes de traitement du signal. **F.** Variance des positions des drones après que la chaîne soit stabilisée. *On observe que  $K$  permet à la fois une convergence rapide (par rapport à  $T_5$ ) ainsi qu'une faible variance dans les positions (par rapport à  $T_0$ ).*

Dans cette expérience, tous les drones sont initialement placés à l'entrée de l'environnement et prêts à décoller, à l'exception d'un Crazyflie qui sert de station de base. Le drone de tête décolle et est contrôlé manuellement, seuls les déplacements vers l'avant et l'arrière sont contrôlables, la stabilisation à égale distance est gérée automatiquement. Les drones suivants suivent l'algorithme 8 de positionnement et décollent si la qualité de signal avec le drone suivant passe sous le seuil.

Les drones présentent un comportement similaire à celui de la simulation. Comme on peut le voir sur la figure 6.4, ils décollent quand la qualité de signal dépasse le seuil et égalisent la qualité des liens jusqu'à la fin de l'expérience. Une fois les liens égalisés, les robots ne sont pas équidistants les uns des autres, ce qui est attendu car l'algorithme égalise les qualités de signal et non les distances entre les drones. C'est cette qualité de signal qui importe plutôt que la distance inter-drones : c'est de cette qualité de signal dont dépend le maintien des communications le long de la chaîne des drones.

Cette qualité de signal ne varie pas linéairement avec la distance inter-drones et il est donc normal que ces distances ne soient pas nécessairement égales en fin d'expérience. Dans cet environnement réel, les drones ne se stabilisent jamais tout à fait en une position. La qualité de signal, même filtrée, est toujours bruitée.

## 6.6 Discussion

La principale hypothèse utilisée pour concevoir l'algorithme U-Chain est que la qualité du signal décroît de manière monotone avec la distance entre deux drones. Cette hypothèse semble raisonnable pour des environnements souterrains pour lesquels les communications radios ne peuvent traverser les parois de l'environnement. Ce n'est cependant pas le cas dans tous les environnements intérieurs. Dans les bâtiments modernes en particulier, il est courant que des signaux radios puissent traverser les parois. Dans ces cas là, la qualité de lien entre deux robots peut s'améliorer dès lors que ceux-ci s'éloignent l'un de l'autre le long d'un environnement à une dimension (par exemple avec un virage en U pour lequel les robots ont un meilleur signal s'ils sont tous deux à une extrémité).

Si l'algorithme est étendu pour gérer la mémorisation des directions prises par le drone de tête aux intersections, il est nécessaire d'utiliser un contrôleur pour gérer la distance aux parois qui soit capable de gérer le passage de ces intersections. Cela peut se faire en décidant de suivre la paroi de gauche ou de droite. Si les intersections présentent plus de deux branches, cette approche est à revoir. La gestion des intersections nécessiterait aussi une capacité à identifier que le robot se trouve à une intersection. Cette identification pourrait être réalisée à l'aide d'une estimation de la position relative entre les robots et donc entre les intersections, ou bien à l'aide d'un système de reconnaissance visuelle. Mais ces deux solutions sont difficiles à embarquer sur de petits drones comme le Crazyflie.

Une des limitations principales des drones aux dimensions similaires au Crazyflie est leur autonomie très limitée (entre 5 et 15 min pour le Crazyflie). Une extension de l'algorithme U-Chain pourrait consister à gérer le remplacement des drones de la chaîne dont la batterie est trop basse pour poursuivre le vol.

Ceci permettrait de maintenir la chaîne de robots de manière prolongée, par exemple observer un environnement sur une longue durée. Le maintien d'un lien radio avec un humain bloqué dans

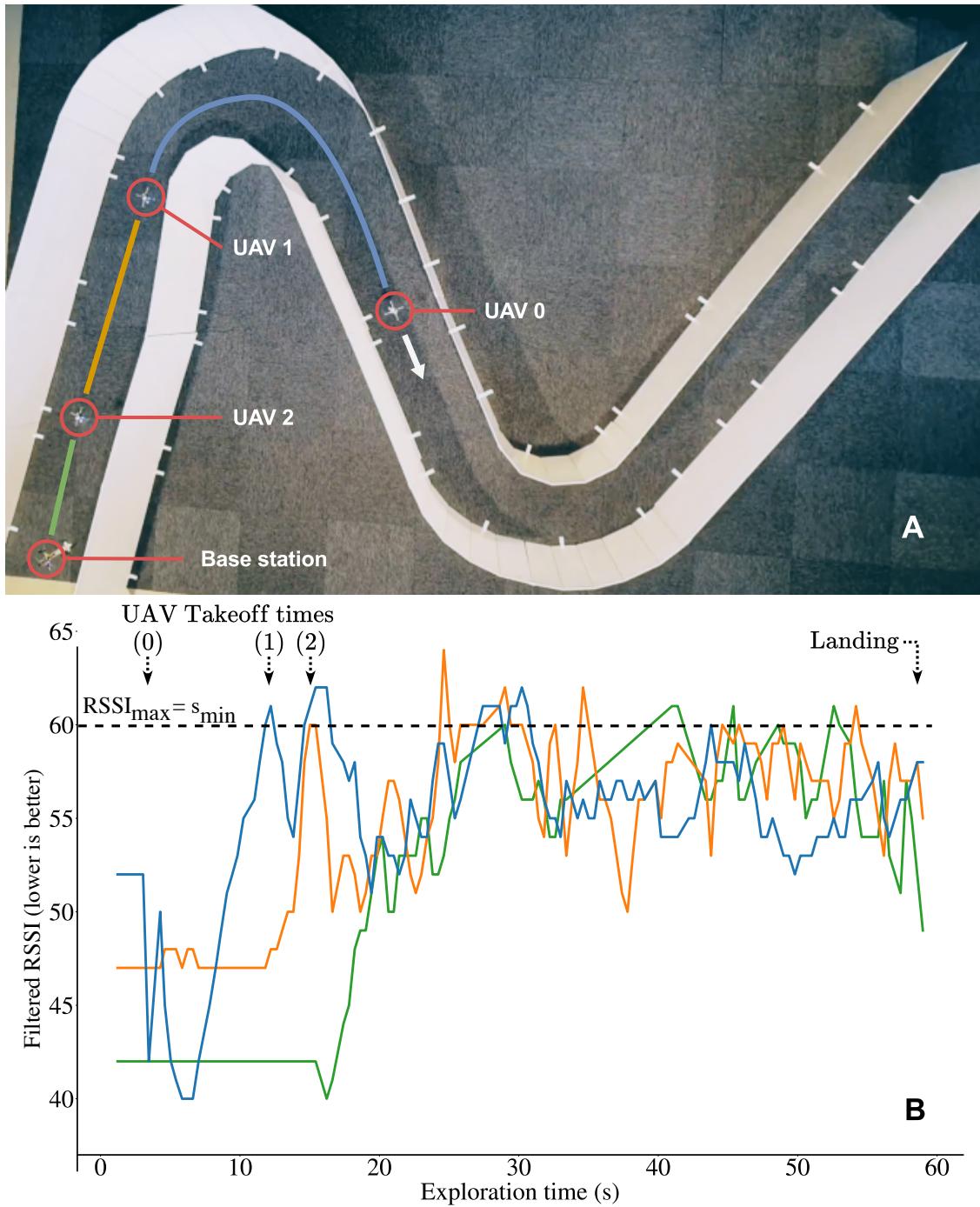


FIGURE 6.4 – Expérience avec 3 Crazyflies plus une station de base Les robots utilisent un capteur de flux optique ainsi que 5 télémètres laser (1 pour l'altitude et 4 pour le positionnement par rapport aux parois). La communication se fait avec un protocole pair à pair personnalisé. **A.** Environnement utilisé pour les expériences. **B.** Qualités de signal (RSSI) pendant l'exploration dans l'environnement réel : après quelques secondes les liens sont de qualité égale et au-dessus du seuil minimal. L'algorithme de positionnement égalise les qualités de signal et non les distances entre robots.

une mine ou un autre bâtiment souterrain pourrait être un exemple de contexte d'usage pour une présence prolongée dans l'environnement.

Cela pourrait être réalisé de manière distribuée en appliquant un tri à bulle le long de la chaîne : chaque drone compare le niveau de charge de sa batterie avec le drone précédent et le drone suivant et échange de position avec le drone plus avancé si la batterie de celui-ci est à un niveau de charge plus bas. Ceci ferait "remonter" les drones dont la batterie est basse vers la station de base. La difficulté principale de cette approche réside dans la réalisation de l'échange de positions entre deux drones consécutifs dans un environnement supposé assez étroit tout en évitant les collisions des robots entre eux ou avec les parois. Une seconde difficulté réside dans la modification de l'algorithme pour maintenir la connectivité de la chaîne au moment de l'échange de position de deux robots. Ceci pourrait demander une contraction temporaire de la chaîne ou bien le décollage de nouveaux drones pour compenser.

## 6.7 Conclusion

En combinant un algorithme de positionnement distribué et réactif avec un filtre de Kalman estimant la qualité du signal radio entre deux drones, l'algorithme U-Chain coordonne une chaîne de drones dans un environnement type tunnel. À l'aide de cet algorithme des drones maintiennent une bonne qualité de lien radio entre eux durant l'exploration d'un tunnel. Le coût en calcul est assez réduit pour pouvoir être embarqué à bord de petits drones comme les Crazyflie. L'algorithme U-Chain s'appuie directement sur la régulation de la qualité du signal radio et peut ainsi s'adapter à des environnements variés et des régimes imprévus de propagation de ce signal.

Dans ce travail nous n'avons pas considéré le cas d'environnements présentant des intersections. Il serait cependant possible de garder en mémoire et transmettre le long de la chaîne la direction prise par le pilote du drone de tête quand celui-ci arrive à une intersection. Les drones suivant dans la chaîne pourraient ainsi suivre le drone de tête à cette intersection.



# Chapitre 7

## Discussion

Nous avons détaillé des approches pour parvenir à stabiliser le vol d'un quadrotor dans un tuyau. Pour la formulation de ces approches nous avons fait des choix, que ce soit dans la nature des mesures pour les perturbations, le type de contrôleur ou bien le fait de se concentrer sur le contrôleur pour stabiliser le robot. Je vais discuter dans ce chapitre des limites et hypothèses qui sont conséquences de ces choix et les éventuelles manières de remédier aux difficultés rencontrées.

### 7.1 Mesure des perturbations dans un tuyau à l'aide d'un bras robotisé

Une première difficulté majeure concerne la manière dont sont considérées les perturbations causées par l'interaction des flux d'air des rotors avec les parois du tuyau. À cause de l'existence de ces perturbations, il est difficile de faire voler un drone dans un tuyau et en conséquence il est difficile de récolter des données sur un drone en vol. C'est la raison pour laquelle nous avons choisi de fixer le multirotor durant la récolte des mesures.

Une des conséquences de ce choix est que les mesures réalisées le sont pour un robot statique. En particulier, contrairement à un multirotor en vol, le robot garde constant son régime moteur et son orientation durant la collecte des données. La circulation de l'air dans un tel environnement est éminemment dynamique, et la cause de cette circulation est l'activité des différents rotors. Les régimes de circulation de l'air dans ces conditions pourraient être très différent du régime de circulation dans des conditions de vol libre. Les variations de la vitesse de rotation des rotors pourraient aussi changer ce régime. Cette différence dans les régimes de circulation pourrait avoir comme conséquence une différence entre les perturbations qui s'appliquent avec le dispositif de mesure que nous avons proposé et celles qui s'appliqueraient sur un robot réel.

Plusieurs expériences pourraient tenter de déterminer s'il existe ou non une différence de régime. Une première approche consiste dans la conduite de mesures similaires, mais avec un régime moteur variable, calqué sur les variations de commande pour les micro-corrections d'orientation d'un multirotor se stabilisant en air libre. Cette approche nécessite de soustraire aux mesures les forces et couples créés par le changement de régime moteur, soit par l'utilisation d'un modèle du robot prédisant ces forces, soit en mesurant ces couples et forces en extérieur puis en rejouant la même séquence de commandes à l'intérieur du tuyau et en soustrayant les forces mesurées en extérieur. Ceci nécessite de pouvoir aligner précisément les deux séquences de commandes et donc

de synchroniser les séquences de mesure.

Une visualisation du régime d'écoulement d'air par des techniques de vélocimétrie particulaire (PIV (SHUKLA et KOMERATH 2018)) permettrait de comparer qualitativement un régime d'écoulement de l'air pour un robot immobile à régime moteur constant avec un régime d'écoulement pour un robot à régime moteur variable et éventuellement une orientation présentant de petit écart à l'horizontale. Le fait de visualiser le régime d'écoulement de l'air à l'aide de la vélocimétrie particulaire permet de ne pas avoir besoin d'aligner les séquences de commande et de mouvement, mais peut conclure à une différence de régime d'écoulement alors que cette différence n'a pas d'impact notable sur les perturbations observées.

Ces expériences pourraient aussi permettre de savoir si les perturbations pour un robot en mouvement, par exemple une vitesse latérale importante, sont identiques à celles pour un robot stationnaire. Une hypothèse serait qu'un multirotor avec un déplacement assez rapide laisse derrière lui les perturbations qu'il crée et n'est donc que peu affecté par celles-ci.

Une seconde difficulté réside dans le fait que le multirotor est un système soumis à d'importantes vibrations. La rotation des rotors, fixés au bout de bras dont la rigidité n'est pas parfaite, cause ces vibrations qui ajoutent un bruit très important aux mesures des perturbations par le capteur de forces six axes sur lequel est posé le robot.

L'existence de ces vibrations nous a décidé à appliquer un filtre aux mesures afin d'en tirer un signal moins bruité. En plus de l'application de ce filtre, nous avons décidé de résumer une perturbation à l'intensité moyenne mesurée sur chaque axe pendant toute la durée de la mesure, associée à une variance. Ce choix de représentation d'une perturbation statique cache une éventuelle dynamique de la perturbation.

Les mesures réalisées selon le protocole présenté permettent de réaliser un traitement plus fin de ces perturbations pour en étudier la dynamique et tenter de la séparer du bruit de mesure, mais nous n'avons pas réalisé ce traitement.

Le dispositif utilisé pour conduire les mesures de perturbations présenté sur la figure 3.3 a des dimensions non négligeables devant celles du robot. Le capteur lui-même et l'extrémité du bras sont placés sous le Crazyflie. Ce dispositif se trouve directement dans les flux d'air générés par les rotors. Sa présence est donc susceptible de modifier le comportement de ces flux. Comme l'illustre la figure 3.3, nous avons tenté de minimiser l'impact du bras robotique par l'usage d'une extension en plastique qui permet de fixer le capteur et le bras à 20cm de l'extrémité du bras lui-même. Cette extension est plus fine que le bras entend limiter l'influence du dispositif de mesure tout en garantissant une rigidité suffisante au dispositif.

Placer le dispositif de mesure au dessus du robot plutôt qu'en dessous pourrait aussi limiter son influence sur les régimes d'écoulement du flux d'air.

Les mesures sont réalisées dans un tuyau, mais contrairement à un probable cas d'utilisation, les mesures sont réalisées avec un drone positionné près des extrémités du tuyau. Les tuyaux que nous utilisons sont assez courts et le bras qui positionne le Crazyflie dans ces tuyaux n'est pas capable de le placer profondément dans le tuyau. Cette configuration pourrait avoir des effets sur les perturbations mesurées. En particulier, il est possible que les flux d'air "s'échappent" par les extrémités du tuyau, les perturbations mesurées pourraient donc être atténuées par rapport à un vol plus loin des extrémités du tuyau.

Corriger cet hypothétique biais nécessiterait de concevoir un nouveau dispositif de déplacement du Crazyflie dans le tuyau.

Une dernière limite dans le protocole utilisé pour collecter des données sur les perturbations concerne l'inclinaison du Crazyflie. Celle-ci est fixée à l'horizontale. La direction des flux d'air créés par les rotors dépend de l'inclinaison du robot. Les perturbations dans le tuyau dépendent donc certainement de l'inclinaison du robot. Pour des petits écarts à l'horizontalité, il est possible que cela ne change pas les perturbations.

Pour trancher la question, il faudrait réaliser des mesures à différentes inclinaisons. Il serait possible, pour chaque position dans de mesure, de réaliser une mesure pour quelques inclinaisons différentes. Les inclinaisons pertinentes concernent probablement une rotation "vers l'avant" et "vers l'arrière" (tangage) pour représenter les déplacements le long de l'axe du tuyau, et une rotation autour de l'axe longitudinal (roulis) pour le déplacement latéral dans le tuyau. Cela multiplierait par 5 le nombre de mesures à réaliser. Le dispositif de mesure actuel est automatisé (placement du robot, déclenchement des mesures), mais l'interaction des différents systèmes n'est pas sans heurts et il est nécessaire de surveiller la collecte de données pour éviter qu'en cas de problème le bras n'abîme le tuyau, le capteur de force ou le Crazyflie fixé dessus.

## 7.2 Choix du type de contrôleur

Nous avons choisi de nous concentrer sur un contrôleur basé sur l'utilisation d'un modèle des perturbations pour planifier une commande qui tiendrait compte des perturbations. Le calcul de cette commande prévoit et rejette ces perturbations via la résolution d'un problème de contrôle optimal.

Cette décision de calculer une commande qui permet de réaliser un vol qui rejette les perturbations n'est pas la seule solution envisageable pour réaliser des vols dans un tuyau. Le modèle des perturbations pourrait tout à fait être utilisé pour calculer une trajectoire pour laquelle les perturbations sont les plus faibles, voire les moins turbulentes. La zone qui apparaît stable et ne pas présenter de perturbations (cf section 3.4) serait par exemple une candidate intéressante pour la conception d'un contrôleur qui ne tente pas de lutter contre les perturbations, mais plutôt de les éviter.

Nous avons fait le choix d'un réseau de neurones classique (perceptron multicouches) comme architecture pour apprendre un contrôleur par imitation. Ce choix a été motivé par le caractère générique et très répandu de ce type d'architectures, mais d'autres choix auraient pu être faits. Le premier concerne l'architecture du réseau elle-même. L'architecture que nous avons choisie avec deux couches cachées de 64 neurones chacune n'est pas le fruit d'un long processus de recherche de la meilleure architecture. Elle a été choisie par imitation d'autres architectures identiques ou similaires dans des contextes d'usage similaire et après vérification de sa capacité à faire voler un Crazyflie en simulation.

Aucune tentative avec d'autres architectures n'a été menée, que ce soit pour des architectures plus petites, et donc plus facilement embarquables, ou bien des architectures intégrant des a priori de structure comme des réseaux convolutionnels à une dimension ou des réseaux récurrents capables de traiter un historique des états passés du robot. Si les résultats présentés dans cette thèse n'indiquent pas spécialement une nécessité de chercher une architecture plus riche et capable de modéliser des fonctions complexes, le fait d'embarquer le contrôleur appris pourrait justifier de consacrer un travail à la détermination de l'architecture minimale capable d'imiter la commande optimale.

Il serait aussi possible de ne pas se limiter aux réseaux de neurones et d'apprendre les paramètres d'un contrôleur plus classique. Apprendre par imitation les paramètres d'un PID par exemple pourrait apporter au contrôleur des propriétés de stabilité ou de symétrie que ne possède pas un réseau de neurones.

Les paramètres utilisés pour le contrôleur MPC se décomposent en ceux du coût, de l'horizon, du pas de temps et du nombre d'itérations pour la résolution du problème d'optimisation. L'obtention d'une commande susceptible de stabiliser le vol d'un quadrotor est très sensible à ces paramètres, en particulier les poids du coût et l'horizon. Cette sensibilité se décompose en deux aspects.

Savoir quels paramètres de la fonction de coût produisent une "bonne trajectoire" n'est pas évident puisque les critères pour juger de la qualité d'une trajectoire se formulent habituellement en termes de coût de cette trajectoire, éventuellement par rapport à une trajectoire de référence. Dans cette thèse, j'ai procédé par essai-erreur pour obtenir des paramètres de la fonction de coût qui permettaient d'obtenir des trajectoires dont l'aspect me paraissait, subjectivement, assez stable. Il existe en particulier un compromis entre l'augmentation des coûts de l'erreur en vitesse et la capacité à corriger rapidement une erreur en position. Plus les vitesses non nulles sont pénalisées, moins la capacité à corriger une erreur en position est importante. Ce compromis, inhérent à la manière dont le coût de ces trajectoires est calculé, demande de trancher entre ces deux aspects. J'ai choisi d'observer les trajectoires et de choisir les poids du coût correspondant aux trajectoires dont le forme générale me paraissait la meilleure plutôt que de définir un critère quantitatif de stabilité.

Un deuxième aspect réside dans la capacité du solveur non linéaire à trouver des trajectoires suffisamment stables dans l'horizon de temps donné. La durée nécessaire dépend de la dynamique du robot et des états de départ de la trajectoire. Si l'horizon est trop court, la résolution du problème de contrôle optimal peut ne pas donner de trajectoire et de commande capable de stabiliser le robot, le caractère local de l'optimalité empêchant de faire progresser le robot vers la stabilité. Là encore, j'ai décidé de la durée de l'horizon par essai-erreur, et cette durée pourrait n'être valide que pour la fonction de coût et le pas de temps que j'ai choisis.

Nous avons choisi de nous intéresser à un contrôleur pour multirotor dont les commandes agissent sur la variation de l'intensité des forces produites par chacun de rotors. Il s'agit donc d'un contrôleur très bas niveau. L'avantage d'un contrôleur bas niveau est sa capacité à tirer parti au mieux du potentiel d'actionnement du robot. Ceci permet d'avoir un contrôleur plus réactif et capable de rejeter des perturbations importantes.

Un contrôleur plus haut niveau se repose sur un ou des sous-contrôleurs en charge de commander une partie de la dynamique, comme par exemple la vitesse angulaire, sur le principe du "backstepping". Ces contrôleurs sont moins aptes à réagir rapidement, puisque dépendant de la vitesse de convergence de la partie de la dynamique contrôlée par le sous-système. L'introduction de ces sous-contrôleurs indépendants constraint la dynamique du robot à fonctionnement moins libre que celui initial. Cependant, l'utilisation d'un contrôleur haut niveau est moins sensible aux erreurs de modèle qu'un contrôleur bas niveau. Le fait de découpler partiellement la dynamique en sous-systèmes hiérarchiques permet de compenser des inexactitudes de modèle par un asservissement via un retour d'état d'une partie spécifique de la dynamique. Le fait d'avoir isolé le contrôle de certaines parties du robot permet de réagir directement sur ces parties uniquement plutôt que sur le système complet pour corriger une éventuelle erreur. Ceci permet aussi une modularité plus grande dans la conception du contrôleur.

Dans le cas de contrôleurs basés sur la résolution d'un problème de contrôle optimal un aspect supplémentaire intervient : la modification de la dynamique par l'introduction de sous-contrôleurs peut rendre plus difficile la résolution du problème de contrôle optimal en modifiant la géométrie de l'espace de recherche des commandes et trajectoires. Il est possible d'introduire de nouvelles variables de décision correspondant aux états contrôlés par les sous-systèmes pour découpler en partie l'optimisation de la commande sur chacun des sous-systèmes. C'est une démarche similaire que j'ai conduite en introduisant comme variable de décision supplémentaire les variables d'état dans la formulation du problème de contrôle optimal de la section 2.4.2.

Il est bien entendu possible de formuler le problème de contrôle optimal uniquement avec la dynamique des états de haut niveau, en laissant au contrôleur asservissant les états plus bas niveau hors de la modélisation. Cela augmente cependant l'inexactitude du modèle utilisé pour calculer une commande optimale.

C'est avec ces considérations que j'ai décidé de formuler le modèle de la dynamique pour formuler le problème de contrôle optimal via des commandes de bas niveau. Pour ne pas devoir modéliser la dynamique réelle de rotors, j'ai fait le choix d'un contrôle plus bas niveau encore que celui possible sur le robot. Il est en effet possible de contrôler l'intensité de la force produite par les moteurs via une commande PWM à bord du Crazyflie. Mais cette commande n'a pas un effet instantané. Formuler le problème de commande optimale comme agissant directement et sans délai sur l'intensité de la force de chaque moteur aurait mené à une commande trop optimiste, et je n'ai pas voulu ajouter un modèle de la dynamique interne des rotors. C'est pour cette raison que j'ai choisi comme compromis entre une modélisation détaillée de la dynamique de moteur et le fait de l'ignorer totalement, de formuler la commande au niveau de la variation de cette intensité, bornant ainsi les changements d'intensité et limitant les écarts entre le modèle de la dynamique des moteurs et la dynamique réelle.

Cependant, pour un déploiement plus aisés d'un contrôleur à bord d'un robot réel, il pourrait être plus aisés de formuler le problème de contrôle optimal avec une commande plus haut niveau et susceptible d'être asservie de manière efficace. Un contrôle à l'aide de la vitesse angulaire, directement mesurable et contrôlable donc possible à asservir efficacement est par exemple une piste à explorer pour le déploiement d'un contrôleur à bord du Crazyflie.

### 7.3 Robustesse du contrôleur aux erreurs de modèle

Nous avons choisi d'utiliser un modèle de la dynamique d'un multirotor pour formuler un problème de contrôle optimal et d'en tirer une commande. Le modèle utilisé est presque identique à celui utilisé dans la simulation pour les résultats présentés dans cette thèse. Les différences résident dans la gestion de l'orientation : le quaternion qui la représente est normalisé à chaque étape de la simulation. Cette normalisation est absente du modèle utilisé dans la formulation du problème de contrôle optimal. En outre, des perturbations aléatoires et hors modèle sont appliquées durant la simulation (cf section 2.4.3).

Le fait d'utiliser un modèle quasiment identique à la simulation peut amener à des comportements de surapprentissage. Ces comportements de surapprentissage qui sont particulièrement répandus en apprentissage par renforcement et posent des problèmes de transfert des contrôleurs appris sur des robots réels ne s'appliquent pas directement sur le contrôleur appris (de manière supervisée) dans cette thèse. Néanmoins, le fait d'utiliser une commande qui provient de la résolution d'un problème de contrôle optimal formulé avec un modèle quasiment identique à la simulation

a des effets similaires : la trajectoire optimisée peut résulter d'une exploitation de propriétés du modèle qui ne sont pas vérifiées sur le robot. Si le modèle utilisé pour le contrôle optimal et la simulation ne correspondent pas à la dynamique réelle du Crazyflie, et c'est très certainement le cas, le contrôleur appris par imitation de cette commande optimale pourrait être complètement inadapté.

La première chose à faire serait de déterminer si le contrôleur appris est robuste à des erreurs de modèle. Pour cela, tout en restant en simulation, il serait possible d'utiliser un contrôleur appris sur un modèle dans un simulateur correspondant à un autre modèle et d'observer si le contrôleur appris garde sa capacité à produire une commande qui permet le vol. Il pourrait être par exemple pertinent de tester un contrôleur appris pour un tuyau d'un diamètre donné dans un tuyau d'un autre diamètre, une différence dans la masse du robot ou encore dans la position de ses rotors.

Une expérience que nous avons réalisée avec des contrôleurs PID dont les gains étaient optimisés pour stabiliser en simulation un multirotor a montré que les contrôleurs PID appris ainsi étaient, par exemple, extrêmement sensibles à des erreurs sur la longueur des bras portant les rotors.

Si le contrôleur appris se montre effectivement trop sensible à des erreurs de modèle, plusieurs approches sont possibles pour tenter d'atténuer cette sensibilité. Une première approche consiste à calculer une commande qui ne soit pas la solution optimale d'un problème formulé sur un seul modèle, mais plutôt un compromis entre plusieurs modèles possibles de la dynamique du robot ou de l'environnement. La formulation du problème de contrôle optimal ferait alors intervenir un jeu de variables d'état par modèle et un unique jeu de variables de commande commun aux trajectoires sur les différents modèles. L'optimisation d'une commande simultanément sur plusieurs modèles serait plus coûteuse en calcul à cause de l'augmentation du nombre de variables, mais aussi du fait que cette optimisation soit conjointe aux différents modèles. Le coût à optimiser pourrait être le coût moyen sur les différents modèles, avec une éventuelle pondération par un a-priori sur la vraisemblance de chaque modèle. Une approche plus prudente vis-à-vis de la confiance dans les modèles utilisés pourrait optimiser le coût de la trajectoire ayant le coût le plus élevé, c'est-à-dire essayer d'améliorer le pire cas sur les modèles utilisés. Cette approche ne garantit pas que la commande générée serait adaptée à la dynamique réelle, mais pourrait réduire les risques de surexploitation d'une spécificité d'un modèle unique.

Une seconde approche pour prévenir une trop grande confiance dans un modèle de dynamique incorrect passerait par l'intégration de la fiabilité du modèle dans les prédictions de celui-ci. L'utilisation de processus gaussiens, comme pour le modèle des perturbations, permet par exemple d'obtenir une incertitude pour la dynamique. Avec un modèle fournissant une incertitude sur la dynamique, il serait possible de propager cette incertitude sur la dynamique à une incertitude sur l'état. Il est possible d'utiliser les techniques de l'estimation d'état pour propager cette incertitude : approximation au premier ordre de la dynamique, particules ou correspondance des moments. Ces incertitudes sur l'état peuvent être utilisées comme contraintes pour éviter les états trop incertains ou assurer avec une certaine probabilité d'éviter certains états (par exemple pour éviter les collisions avec les parois). Il serait aussi possible de définir le coût à partir de la distribution estimée pour un état.

Pour limiter les conséquences d'une sensibilité importante de la commande aux erreurs de modèle il est aussi possible de réduire ces erreurs. Cette réduction peut se faire hors ligne, en complétant un modèle dynamique, par exemple conçu avec des équations comme celles décrites dans la section 2.3, par un modèle appris sur des données réelles.

La correction du modèle dynamique pourrait aussi se faire en ligne, comme un contrôle adaptatif. Le contrôleur appris présenté dans cette thèse ne permet pas directement de tenir compte d'une mise à jour de la dynamique du modèle, mais il serait possible de le modifier pour, par exemple, prendre en entrée la masse du robot, le niveau actuel de tension de la batterie, etc., et d'estimer à l'aide d'un autre algorithme la valeur réelle de ces paramètres.

Une sensibilité importante aux différences entre le modèle utilisé pour la formulation du problème de contrôle optimal et la dynamique est une propriété qui peut poser des problèmes pour l'utilisation du contrôleur appris par imitation d'un contrôle optimal. Cette sensibilité reflète cependant le fait de prendre les informations fournies par un modèle de l'environnement et de la dynamique. Dans cette thèse nous avons choisi de collecter des données sur des perturbations présentes dans un tuyau, de se servir de ces données pour apprendre un modèle capable de réaliser des prédictions de ces perturbations et enfin d'intégrer ce modèle prédictif dans la boucle de contrôle pour prévenir ces perturbations.

Ce travail introduit donc volontairement de la sensibilité au modèle et à ses erreurs dans le contrôle. Sans cette sensibilité on ne peut s'attendre à ce que le contrôleur tire profit des informations fournies par le modèle. Pour concevoir un contrôleur informé de perturbations d'un environnement il est nécessaire d'accepter en compromis que le contrôleur soit moins adapté dans le cas où le modèle des perturbations utilisé ne correspond pas à la situation réelle.

## 7.4 Approche par la modification de l'architecture du robot.

Nous avons choisi de travailler sur l'aspect de la commande d'un multirotor pour améliorer ses capacités de vol stable dans un tuyau. Une autre voie pour améliorer cette stabilité pourrait résider dans l'architecture même du robot. Dans le chapitre 2 j'ai brièvement évoqué les multirotors complètement actionnés (RASHAD et al. 2020). Ces robots agencent leurs rotors de manière à être capables de produire des forces sur les 3 axes sans changer leur inclinaison. Ceci leur donne une meilleure capacité à rejeter des perturbations externes (JIANG et VOYLES 2014). Ces capacités de rejet des perturbations pourraient être suffisantes pour réaliser des vols stables dans un tuyau avec une commande qui ne soit pas spécifiquement adaptée à cette tâche en particulier.

Certaines limites soulevées dans la section 7.1 ne s'appliquent pas à ces architectures. L'importance de l'inclinaison du drone pour la mesure des perturbations et l'hypothétique dynamique créée par les changements d'orientation du robot nécessaires à la stabilisation de la position s'appliqueraient moins pour un multirotor complètement actionné puisque celui-ci n'aurait pas besoin de changer son inclinaison pour corriger sa position et réalisera des vols pour lesquels l'inclinaison du robot serait plus proche de l'horizontale que pour un quadrotor. La variation des perturbations créées avec l'intensité respective de la force créée par les différents moteurs pourrait cependant remplacer ces limites et pourrait nécessiter de réaliser de nombreuses mesures pour une même position.

Une étude expérimentale de l'agencement et de la taille des rotors pour limiter les perturbations pourrait aussi être conduite. La dynamique de l'écoulement des flux d'air causés par les rotors pourrait être améliorée par un changement d'échelle ou bien une limitation de l'interaction des différents flux entre eux (SHUKLA et KOMERATH 2018). Une telle approche demanderait un simulateur de la dynamique de ces fluides.

Lucien Renaud a conçu et construit un robot avec une architecture complètement actionnée à 6 rotors sur la base d'un Crazyflie. Nous avons tenté de réaliser un contrôleur qui soit capable de faire voler ce robot complètement actionné. Nous sommes parvenus uniquement à le faire voler via un contrôleur classique de quadrotor adapté, c'est-à-dire une commande qui ne tire pas parti de l'actionnement complet du robot.

Ce robot n'a donc jamais volé avec une commande complètement actionnée, mais il serait possible de l'utiliser pour réaliser des mesures de perturbations avec le protocole utilisé pour le quadrotor et ainsi mieux quantifier l'apport éventuel d'une telle architecture pour le vol dans un tuyau. En particulier, cela permettrait de vérifier si l'agencement non horizontal des rotors sur une architecture complètement actionnée ne produit par des perturbations plus importantes sur certains axes, ce qui pourrait réduire l'intérêt d'une telle architecture pour le vol dans un tuyau.

# Conclusion

Dans cette thèse je présente plusieurs éléments qui contribuent à faire voler un quadrotor Crazyflie dans un tuyau en simulation.

Je formule un problème de contrôle optimal dont la résolution permet la commande d'un Crazyflie en simulation. Cette formulation se traduit par une implémentation d'un simulateur et d'un contrôleur à l'aide de la bibliothèque CasADi et d'une transcription pour la résolution du problème de contrôle optimal à l'aide du solveur non linéaire IPOPT. La formalisation de ce problème de contrôle optimal et l'implémentation permettent la commande d'autres architectures de multirotors. Toute architecture dont le placement (position et orientation) des rotors est fixée peut être directement intégrée et commandée à l'aide d'un contrôleur MPC basé sur la résolution du problème de contrôle optimal associé à l'architecture et à un environnement éventuel.

Je détaille ensuite un dispositif de mesure des perturbations statiques qui s'exercent sur un Crazyflie en régime permanent dans des tuyaux cylindriques de diamètre 40cm, 50cm et 65cm de diamètre à l'aide d'un capteur de force 6 axes. Je décris aussi un protocole de traitement des mesures pour effacer le bruit important causé par les vibrations du robot. Je présente deux modèles, faisant intervenir des processus gaussiens et des réseaux de neurones, capables de réaliser des prédictions des perturbations en n'importe quel point à l'intérieur du tuyau. Ces modèles sont appris à l'aide des mesures ponctuelles réalisées dans les tuyaux.

Je propose aussi l'adaptation de l'algorithme DAGGER pour l'apprentissage par imitation par un réseau de neurones d'un contrôleur optimal pour le Crazyflie et tenant compte des perturbations modélisées. Je présente les performances de ce contrôleur appris dans une simulation d'environnement reproduisant les perturbations mesurées dans un tuyau réel.

Enfin, je décris trois approches pour estimer l'état du robot en utilisant les particularités de l'environnement. Ces approches permettent en particulier d'estimer la position du robot, qui est cruciale pour prédire les perturbations, et éviter les collisions avec les parois. La première suppose que le robot est presque à l'horizontale et permet de déduire la distance aux parois à l'aide de télémètres laser. La seconde nécessite de connaître l'orientation du robot et permet d'estimer sa position sur deux axes à l'aide de télémètres laser. La dernière approche détaille les équations d'un filtre de Kalman étendu intégrant un modèle de mesures des télémètres laser avec les mesures de la centrale inertielle.

Les contrôleurs présentés dans cette thèse n'ont été testés qu'en simulation, et utilisent un modèle quasi identique au simulateur. L'implémentation fonctionnelle du contrôleur appris à bord du Crazyflie constitue donc une extension immédiate du travail que nous avons réalisé.

Cette implémentation, si elle permet le contrôle stable d'un Crazyflie dans un tuyau, permettrait de réaliser plus facilement des mesures de perturbations en vol libre, c'est-à-dire sans l'influence du dispositif de mesures actuel et rendrait ainsi possible de s'intéresser à l'aspect dy-

namique des perturbations dans un tuyau en présence d'un quadrotor.

Par ailleurs, l'estimateur d'état avec modèle inverse des mesures des télémètres n'a été testé qu'en simulation. L'implémentation du filtre de Kalman étendu décrit dans la section 5.6 n'a pu être terminée. L'implémentation de ces estimateurs en C et embarqués à bord du Crazyflie pour des tests en environnement réel constitue un autre axe d'extension direct et indépendant du contrôleur.

# Bibliographie

- [1] P. ABBEEL, A. COATES et A. Y. NG. « Autonomous Helicopter Aerobatics through Apprenticeship Learning ». In : *The International Journal of Robotics Research* 29.13 (1<sup>er</sup> nov. 2010), p. 1608-1639. ISSN : 0278-3649. DOI : 10.1177/0278364910371999.
- [2] F. AMIGONI, J. BANFI et N. BASILICO. « Multirobot Exploration of Communication-Restricted Environments : A Survey ». In : *IEEE Intelligent Systems* 32.6 (nov. 2017), p. 48-57. ISSN : 1541-1672. DOI : 10.1109/MIS.2017.4531226.
- [3] B. AMOS et al. « Differentiable MPC for End-to-End Planning and Control ». In : *Advances in Neural Information Processing Systems*. Sous la dir. de S. BENGIO et al. T. 31. Curran Associates, Inc., 2018.
- [4] J. A. E. ANDERSSON et al. « CasADi : A Software Framework for Nonlinear Optimization and Optimal Control ». In : *Mathematical Programming Computation* 11.1 (1<sup>er</sup> mars 2019), p. 1-36. ISSN : 1867-2957. DOI : 10.1007/s12532-018-0139-4.
- [5] F. J. ANDRADE CHAVEZ. « Force-Torque Sensing in Robotics ». 8 avr. 2019. DOI : 10.15167/andrade-chavez-francisco-javier\_phd2019-04-08.
- [6] T. ANTONSSON. *Measuring Propeller RPM : Part 3 / Bitcraze*. 9 fév. 2015. URL : <https://www.bitcraze.io/2015/02/measuring-propeller-rpm-part-3/>.
- [7] S. BANSAL et al. « Learning Quadrotor Dynamics Using Neural Network for Flight Control ». In : *2016 IEEE 55th Conference on Decision and Control (CDC)*. 2016 IEEE 55th Conference on Decision and Control (CDC). Déc. 2016, p. 4653-4660. DOI : 10.1109/CDC.2016.7798978.
- [8] M. BELKHEIRI et al. « Different Linearization Control Techniques for a Quadrotor System ». In : *CCCA12*. 2012 2nd International Conference on Communications, Computing and Control Applications (CCCA). Marseilles, France : IEEE, déc. 2012, p. 1-6. ISBN : 978-1-4673-4695-5 978-1-4673-4694-8 978-1-4673-4693-1. DOI : 10.1109/CCCA.2012.6417914.
- [9] P. BERNAL-POLO et H. MARTÍNEZ-BARBERÁ. « Kalman Filtering for Attitude Estimation with Quaternions and Concepts from Manifold Theory ». In : *Sensors* 19.1 (1 jan. 2019), p. 149. DOI : 10.3390/s19010149.
- [10] J. T. BETTS. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming, Second Edition*. Advances in Design and Control. Society for Industrial and Applied Mathematics, 1<sup>er</sup> jan. 2010. 442 p. ISBN : 978-0-89871-688-7. DOI : 10.1137/1.9780898718577.

- [11] D. BICEGO et al. « Nonlinear Model Predictive Control with Enhanced Actuator Model for Multi-Rotor Aerial Vehicles with Generic Designs ». In : *Journal of Intelligent & Robotic Systems* 100.3-4 (déc. 2020), p. 1213-1247. ISSN : 0921-0296, 1573-0409. DOI : [10.1007/s10846-020-01250-9](https://doi.org/10.1007/s10846-020-01250-9).
- [12] E. V. BONILLA, K. CHAI et C. WILLIAMS. « Multi-Task Gaussian Process Prediction ». In : *Advances in Neural Information Processing Systems*. T. 20. Curran Associates, Inc., 2008.
- [13] A. BRIOD, A. KLAPOCZ et al. « The AirBurr : A Flying Robot That Can Exploit Collisions ». In : *2012 ICME International Conference on Complex Medical Engineering (CME)*. 2012 ICME International Conference on Complex Medical Engineering (CME). Kobe, Japan : IEEE, juill. 2012, p. 569-574. ISBN : 978-1-4673-1618-7 978-1-4673-1617-0 978-1-4673-1616-3. DOI : [10.1109/ICCME.2012.6275674](https://doi.org/10.1109/ICCME.2012.6275674).
- [14] A. BRIOD, P. KORNATOWSKI et al. « A Collision-resilient Flying Robot : A Collision-resilient Flying Robot ». In : *Journal of Field Robotics* 31.4 (juill. 2014), p. 496-509. ISSN : 15564959. DOI : [10.1002/rob.21495](https://doi.org/10.1002/rob.21495).
- [15] M. BROSSARD, A. BARRAU et S. BONNABEL. « A Code for Unscented Kalman Filtering on Manifolds (UKF-M) ». In : *International Conference on Robotics and Automation (ICRA)*. Paris, France, mai 2020. DOI : [10.1109/ICRA40945.2020.9197489](https://doi.org/10.1109/ICRA40945.2020.9197489).
- [16] C. BUDACIU et al. « On the Evaluation of the Crazyflie Modular Quadcopter System ». In : *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). Sept. 2019, p. 1189-1195. DOI : [10.1109/ETFA.2019.8869202](https://doi.org/10.1109/ETFA.2019.8869202).
- [17] W. BULTEN, A. C. VAN ROSSUM et W. F. G. HASELAGER. « Human SLAM, Indoor Localisation of Devices and Users ». In : *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*. 2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI). Berlin : IEEE, avr. 2016, p. 211-222. ISBN : 978-1-4673-9948-7. DOI : [10.1109/IoTDI.2015.19](https://doi.org/10.1109/IoTDI.2015.19).
- [18] S. BUTTERWORTH. « On the Theory of Filter Amplifiers ». In : *Wireless Engineer* 7.6 (1930), p. 536-541.
- [19] G. CANO LOPEZ et al. « Intelligent Control of a Quadrotor with Proximal Policy Optimization Reinforcement Learning ». In : *2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE)*. 2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE). Nov. 2018, p. 503-508. DOI : [10.1109/LARS/SBR/WRE.2018.00094](https://doi.org/10.1109/LARS/SBR/WRE.2018.00094).
- [20] B. B. CARLOS et al. « An Efficient Real-Time NMPC for Quadrotor Position Control under Communication Time-Delay ». In : *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. Shenzhen, China : IEEE, 13 déc. 2020, p. 982-989. ISBN : 978-1-72817-709-0. DOI : [10.1109/ICARCV50220.2020.9305513](https://doi.org/10.1109/ICARCV50220.2020.9305513).

- [21] K. CESARE et al. « Multi-UAV Exploration with Limited Communication and Battery ». In : *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015 IEEE International Conference on Robotics and Automation (ICRA). Seattle, WA, USA : IEEE, mai 2015, p. 2230-2235. ISBN : 978-1-4799-6923-4. DOI : 10.1109/ICRA.2015.7139494.
- [22] K. CHATZILYGEROUDIS et al. « A Survey on Policy Search Algorithms for Learning Robot Controllers in a Handful of Trials ». In : *IEEE Transactions on Robotics* 36.2 (avr. 2020), p. 328-347. ISSN : 1552-3098, 1941-0468. DOI : 10.1109/TR0.2019.2958211.
- [23] L. C. CHEESEMAN et W. E. BENNETT. « The Effect of the Ground on a Helicopter Rotor in Forward Flight ». In : *Aeronautical Research Council Reports and Memoranda* 3021 (1955).
- [24] A. CHEMORI et N. MARCHAND. « A Prediction-Based Nonlinear Controller for Stabilization of a Non-Minimum Phase PVTOL Aircraft ». In : *International Journal of Robust and Nonlinear Control* 18.8 (25 mai 2008), p. 876-889. ISSN : 10498923, 10991239. DOI : 10.1002/rnc.1248.
- [25] K. CHUA et al. « Deep Reinforcement Learning in a Handful of Trials Using Probabilistic Dynamics Models ». 30 mai 2018. arXiv : 1805.12114 [cs, stat].
- [26] L. DANJUN et al. « Autonomous Landing of Quadrotor Based on Ground Effect Modelling ». In : *2015 34th Chinese Control Conference (CCC)*. 2015 34th Chinese Control Conference (CCC). Juill. 2015, p. 5647-5652. DOI : 10.1109/ChiCC.2015.7260521.
- [27] G. DANTZIG. « Maximization of a Linear Function of Variables Subject to Linear Inequalities. » In : *Activity Analysis of Production and Allocation* (1951).
- [28] M. DORIGO et al. « Swarmroid : A Novel Concept for the Study of Heterogeneous Robotic Swarms ». In : *IEEE Robotics & Automation Magazine* 20.4 (déc. 2013), p. 60-71. ISSN : 1070-9932. DOI : 10.1109/MRA.2013.2252996.
- [29] B. P. DUISTERHOF et al. « Learning to Seek : Autonomous Source Seeking with Deep Reinforcement Learning Onboard a Nano Drone Microcontroller ». Version 6. In : (2019). DOI : 10.48550/ARXIV.1909.11236.
- [30] M. FAESSLER, A. FRANCHI et D. SCARAMUZZA. « Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories ». In : *IEEE Robotics and Automation Letters* 3.2 (avr. 2018), p. 620-626. ISSN : 2377-3766. DOI : 10.1109/LRA.2017.2776353.
- [31] B. S. FAİÇAL et al. « The Use of Unmanned Aerial Vehicles and Wireless Sensor Networks for Spraying Pesticides ». In : *Journal of Systems Architecture* 60.4 (1<sup>er</sup> avr. 2014), p. 393-404. ISSN : 1383-7621. DOI : 10.1016/j.sysarc.2014.01.004.
- [32] R. FEATHERSTONE. *Robot Dynamics Algorithms*. Kluwer International Series in Engineering and Computer Science Robotics : Vision, Manipulation and Sensors 22. New York : Springer Science+Business Media, 1987. 211 p. ISBN : 978-1-4757-6437-6 978-0-89838-230-3.

- [33] H. J. FERREAU et al. « qpOASES : A Parametric Active-Set Algorithm for Quadratic Programming ». In : *Mathematical Programming Computation* 6.4 (déc. 2014), p. 327-363. ISSN : 1867-2949, 1867-2957. DOI : 10.1007/s12532-014-0071-1.
- [34] J. FÖRSTER. « System Identification of the Crazyflie 2.0 Nano Quadrocopter ». In : (août 2015). DOI : 10.3929/ethz-b-000214143.
- [35] A. FRANCHI et al. « Full-Pose Tracking Control for Aerial Robotic Systems With Laterally Bounded Input Force ». In : *IEEE Transactions on Robotics* 34.2 (avr. 2018), p. 534-541. ISSN : 1552-3098, 1941-0468. DOI : 10.1109/TRO.2017.2786734.
- [36] G. FREIRE et R. COTA. « Capture of Images in Inaccessible Areas in an Underground Mine Using an Unmanned Aerial Vehicle ». In : *Proceedings of the First International Conference on Underground Mining Technology*. First International Conference on Underground Mining Technology. Australian Centre for Geomechanics, Perth, 2017. ISBN : 978-0-9924810-7-0. DOI : 10.36487/ACG\_rep/1710\_54\_Freire.
- [37] E. FRESK et G. NIKOLAKOPOULOS. « Full Quaternion Based Attitude Control for a Quadrotor ». In : *2013 European Control Conference (ECC)*. 2013 European Control Conference (ECC). Juill. 2013, p. 3864-3869. DOI : 10.23919/ECC.2013.6669617.
- [38] D. GANDHI, L. PINTO et A. GUPTA. « Learning to Fly by Crashing ». In : *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Sept. 2017, p. 3948-3955. DOI : 10.1109/IROS.2017.8206247.
- [39] R. GANTENBRINK. *Upuaut Project*. 1999. URL : [https://web.archive.org/web/20201030225847fw\\_/http://cheops.org/startpage/therobots/robots.htm](https://web.archive.org/web/20201030225847fw_/http://cheops.org/startpage/therobots/robots.htm).
- [40] W. GIERNACKI et al. « Crazyflie 2.0 Quadrotor as a Platform for Research and Education in Robotics and Control Engineering ». In : *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*. 2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR). Miedzyzdroje, Poland : IEEE, août 2017, p. 37-42. ISBN : 978-1-5386-2402-9. DOI : 10.1109/MMAR.2017.8046794.
- [41] J. C. GILBERT. « Fragments d'Optimisation Différentiable - Théories et Algorithmes ». Master. France, mars 2021.
- [42] R. GILL, M. W. MUELLER et R. D'ANDREA. « Full-Order Solution to the Attitude Reset Problem for Kalman Filtering of Attitudes ». In : *Journal of Guidance, Control, and Dynamics* 43.7 (juill. 2020), p. 1232-1246. ISSN : 1533-3884. DOI : 10.2514/1.G004134.
- [43] S. GOMES et J. RAMOS. « Airship Dynamic Modeling for Autonomous Operation ». In : *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*. IEEE International Conference on Robotics and Automation. T. 4. Leuven, Belgium : IEEE, 1998, p. 3462-3467. ISBN : 978-0-7803-4300-9. DOI : 10.1109/ROBOT.1998.680973.

- [44] M. GREIFF. « Modelling and Control of the Crazyflie Quadrotor for Aggressive and Autonomous Flight by Optical Flow Driven State Estimation ». In : (2017) (2017). ISSN : 0280-5316.
- [45] P. D. GROVES. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*. 2nd ed. GNSS Technology and Application Series. Boston : Artech House, 2013. 776 p. ISBN : 978-1-60807-005-3.
- [46] J. GUERRERO-CASTELLANOS et al. « Bounded Attitude Control of Rigid Bodies : Real-time Experimentation to a Quadrotor Mini-Helicopter ». In : *Control Engineering Practice* 19.8 (août 2011), p. 790-797. ISSN : 09670661. DOI : 10.1016/j.conengprac.2011.04.004.
- [47] M. HAMANDI et al. « Design of Multirotor Aerial Vehicles : A Taxonomy Based on Input Allocation ». Mai 2021.
- [48] S. HAUERT, J.-C. ZUFFEREY et D. FLOREANO. « Evolved Swarming without Positioning Information : An Application in Aerial Communication Relay ». In : *Autonomous Robots* 26.1 (jan. 2009), p. 21-32. ISSN : 0929-5593, 1573-7527. DOI : 10.1007/s10514-008-9104-9.
- [49] S. S. HAYKIN. *Neural Networks : A Comprehensive Foundation*. 2nd ed. Upper Saddle River, N.J : Prentice Hall, 1999. 842 p. ISBN : 978-0-13-273350-2.
- [50] K. HE et al. « Deep Residual Learning for Image Recognition ». In : *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA : IEEE, juin 2016, p. 770-778. ISBN : 978-1-4673-8851-1. DOI : 10.1109/CVPR.2016.90.
- [51] G. HIRZINGER. « Robots in Space - a Survey ». In : *Advanced Robotics* 9.6 (jan. 1994), p. 625-651. ISSN : 0169-1864, 1568-5535. DOI : 10.1163/156855395X00337.
- [52] K. HORNIK, M. STINCHCOMBE et H. WHITE. « Multilayer Feedforward Networks Are Universal Approximators ». In : *Neural Networks* 2.5 (1<sup>er</sup> jan. 1989), p. 359-366. ISSN : 0893-6080. DOI : 10.1016/0893-6080(89)90020-8.
- [53] B. HOUSKA, H. J. FERREAU et M. DIEHL. « ACADO Toolkit-An Open-Source Framework for Automatic Control and Dynamic Optimization : ACADO TOOLKIT ». In : *Optimal Control Applications and Methods* 32.3 (mai 2011), p. 298-312. ISSN : 01432087. DOI : 10.1002/oca.939.
- [54] M. A. HSIEH et al. « Maintaining Network Connectivity and Performance in Robot Teams ». In : *Journal of Field Robotics* 25.1-2 (jan. 2008), p. 111-131. ISSN : 15564959, 15564967. DOI : 10.1002/rob.20221.
- [55] J. HUGHES et D. LYONS. « Wall Detection Via IMU Data Classification In Autonomous Quadcopters ». In : *2021 7th International Conference on Control, Automation and Robotics (ICCAR)*. 2021 7th International Conference on Control, Automation and Robotics (ICCAR). Avr. 2021, p. 189-195. DOI : 10.1109/ICCAR52225.2021.9463486.
- [56] J. HWANGBO et al. « Control of a Quadrotor with Reinforcement Learning ». In : *IEEE Robotics and Automation Letters* 2.4 (oct. 2017), p. 2096-2103. ISSN : 2377-3766, 2377-3774. DOI : 10.1109/LRA.2017.2720851. arXiv : 1707.05110.

- [57] « Introduction to Feedback Control of Underactuated VTOL vehicles : A Review of Basic Control Design Ideas and Principles ». In : *IEEE Control Systems* 33.1 (fév. 2013), p. 61-75. ISSN : 1066-033X, 1941-000X. DOI : 10.1109/MCS.2012.2225931.
- [58] S. IOFFE et C. SZEGEDY. « Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift ». 10 fév. 2015. arXiv : 1502.03167 [cs].
- [59] J. JAFFE. « Bottleneck Flow Control ». In : *IEEE Transactions on Communications* 29.7 (juill. 1981), p. 954-962. ISSN : 0096-2244. DOI : 10.1109/TCOM.1981.1095081.
- [60] G. JIANG et R. VOYLES. « A Nonparallel Hexrotor UAV with Faster Response to Disturbances for Precision Position Keeping ». In : *2014 IEEE International Symposium on Safety, Security, and Rescue Robotics (2014)*. 2014 IEEE International Symposium on Safety, Security, and Rescue Robotics (2014). Oct. 2014, p. 1-5. DOI : 10.1109/SSRR.2014.7017669.
- [61] G. JIANG et R. VOYLES. « Hexrotor UAV Platform Enabling Dextrous Interaction with Structures-Flight Test ». In : *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. 2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR). Oct. 2013, p. 1-6. DOI : 10.1109/SSRR.2013.6719377.
- [62] E. JONES et al. « Advances and Applications for Automated Drones in Underground Mining Operations ». In : Ninth International Conference on Deep and High Stress Mining. 2019, p. 323-334. DOI : 10.36487/ACG\_rep/1952\_24\_Jones.
- [63] G. KAHN et al. « PLATO : Policy Learning Using Adaptive Trajectory Optimization ». In : *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017 IEEE International Conference on Robotics and Automation (ICRA). Mai 2017, p. 3342-3349. DOI : 10.1109/ICRA.2017.7989379.
- [64] E. KAUFMANN et al. « Beauty and the Beast : Optimal Methods Meet Learning for Drone Racing ». In : *2019 International Conference on Robotics and Automation (ICRA)*. 2019 International Conference on Robotics and Automation (ICRA). Mai 2019, p. 690-696. DOI : 10.1109/ICRA.2019.8793631.
- [65] M. KELLY. « An Introduction to Trajectory Optimization : How to Do Your Own Direct Collocation ». In : *SIAM Review* 59.4 (1<sup>er</sup> jan. 2017), p. 849-904. ISSN : 0036-1445. DOI : 10.1137/16M1062569.
- [66] A. A. KHUWAJA et al. « A Survey of Channel Modeling for UAV Communications ». In : *IEEE Communications Surveys & Tutorials* 20.4 (hiver 2018), p. 2804-2821. ISSN : 1553-877X, 2373-745X. DOI : 10.1109/COMST.2018.2856587.
- [67] D. P. KINGMA et J. BA. « Adam : A Method for Stochastic Optimization ». Mai 2015. arXiv : 1412.6980 [cs].
- [68] W. KOCH et al. « Reinforcement Learning for UAV Attitude Control ». In : *ACM Transactions on Cyber-Physical Systems* 3.2 (30 avr. 2019), p. 1-21. ISSN : 2378-962X, 2378-9638. DOI : 10.1145/3301273.

- [69] P. LACLAU et al. « Signal-Based Self-Organization of a Chain of UAVs for Subterranean Exploration ». In : *Frontiers in Robotics and AI* 8 (23 avr. 2021), p. 614206. ISSN : 2296-9144. DOI : 10.3389/frobt.2021.614206.
- [70] N. O. LAMBERT et al. « Low-Level Control of a Quadrotor With Deep Model-Based Reinforcement Learning ». In : *IEEE Robotics and Automation Letters* 4.4 (oct. 2019), p. 4224-4230. ISSN : 2377-3766, 2377-3774. DOI : 10.1109/LRA.2019.2930489.
- [71] B. LANDRY et al. « Aggressive Quadrotor Flight through Cluttered Environments Using Mixed Integer Programming ». In : *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016 IEEE International Conference on Robotics and Automation (ICRA). Mai 2016, p. 1469-1475. DOI : 10.1109/ICRA.2016.7487282.
- [72] Y. A. LECLUN et al. « Efficient BackProp ». In : *Neural Networks : Tricks of the Trade : Second Edition*. Sous la dir. de G. MONTAVON, G. B. ORR et K.-R. MÜLLER. Lecture Notes in Computer Science. Berlin, Heidelberg : Springer, 2012, p. 9-48. ISBN : 978-3-642-35289-8. DOI : 10.1007/978-3-642-35289-8\_3.
- [73] T. LEE, M. LEOK et N. H. MCCLAMROCH. « Geometric Tracking Control of a Quadrotor UAV on  $SE(3)$  ». In : *49th IEEE Conference on Decision and Control (CDC)*. 49th IEEE Conference on Decision and Control (CDC). Déc. 2010, p. 5420-5425. DOI : 10.1109/CDC.2010.5717652.
- [74] S. LEVINE et V. KOLTUN. « Guided Policy Search ». In : *International Conference on Machine Learning*. International Conference on Machine Learning. 13 fév. 2013, p. 1-9.
- [75] S. LI et al. « Aggressive Online Control of a Quadrotor via Deep Network Representations of Optimality Principles ». In : *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020 IEEE International Conference on Robotics and Automation (ICRA). Paris, France : IEEE, mai 2020, p. 6282-6287. ISBN : 978-1-72817-395-5. DOI : 10.1109/ICRA40945.2020.9197443.
- [76] W. LI et E. TODOROV. « Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems ». In : *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics*. First International Conference on Informatics in Control, Automation and Robotics. Setúbal, Portugal, 2004, p. 222-229. ISBN : 978-972-8865-12-2. DOI : 10.5220/0001143902220229.
- [77] J. LIN et al. « Flying through a Narrow Gap Using Neural Network : An End-to-End Planning and Control Approach ». In : *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Macau, China : IEEE, nov. 2019, p. 3526-3533. ISBN : 978-1-72814-004-9. DOI : 10.1109/IROS40897.2019.8967944.
- [78] S. O. H. MADGWICK, A. J. L. HARRISON et R. VAIDYANATHAN. « Estimation of IMU and MARG Orientation Using a Gradient Descent Algorithm ». In : *2011 IEEE International Conference on Rehabilitation Robotics*. 2011 IEEE Internatio-

- nal Conference on Rehabilitation Robotics. Juin 2011, p. 1-7. DOI : 10.1109/ICORR.2011.5975346.
- [79] S. MAFRICA, A. SERVEL et F. RUFFIER. « Optic-Flow Based Car-like Robot Operating in a 5-Decade Light Level Range ». In : *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016 IEEE International Conference on Robotics and Automation (ICRA). Stockholm : IEEE, mai 2016, p. 5568-5575. ISBN : 978-1-4673-8026-3. DOI : 10.1109/ICRA.2016.7487774.
- [80] A. MAHE, C. PRADALIER et M. GEIST. « Trajectory-Control Using Deep System Identification and Model Predictive Control for Drone Control under Uncertain Load ». In : *2018 22nd International Conference on System Theory, Control and Computing (ICSTCC)*. 2018 22nd International Conference on System Theory, Control and Computing (ICSTCC). Sinaia : IEEE, oct. 2018, p. 753-758. ISBN : 978-1-5386-4444-7. DOI : 10.1109/ICSTCC.2018.8540719.
- [81] R. MAHONY et al. « Nonlinear Complementary Filters on the Special Linear Group ». In : *International Journal of Control* 85.10 (oct. 2012), p. 1557-1573. ISSN : 0020-7179, 1366-5820. DOI : 10.1080/00207179.2012.693951.
- [82] A. MANECY. « Stratégies de Guidage Visuel Bio-Inspirées : Application à La Stabilisation d'un Micro-Drone et à La Poursuite de Cibles ». Theses. EEATS, juill. 2015.
- [83] S. S. MANSOURI, M. CASTAÑO et al. « Autonomous MAV Navigation in Underground Mines Using Darkness Contours Detection ». In : *Computer Vision Systems*. Sous la dir. de D. TZOVARAS et al. T. 11754. Lecture Notes in Computer Science. Cham : Springer International Publishing, 2019, p. 164-174. ISBN : 978-3-030-34994-3 978-3-030-34995-0. DOI : 10.1007/978-3-030-34995-0\_16.
- [84] S. S. MANSOURI, C. KANELAKIS et al. « Subterranean MAV Navigation Based on Nonlinear MPC with Collision Avoidance Constraints ». 7 juin 2020. arXiv : 2006.04227 [cs].
- [85] P. MARTIN et E. SALAÜN. « The True Role of Accelerometer Feedback in Quadrotor Control ». In : *2010 IEEE International Conference on Robotics and Automation*. 2010 IEEE International Conference on Robotics and Automation. Mai 2010, p. 1623-1629. DOI : 10.1109/ROBOT.2010.5509980.
- [86] K. N. MC GUIRE et al. « Minimal Navigation Solution for a Swarm of Tiny Flying Robots to Explore an Unknown Environment ». In : *Science Robotics* 4.35 (23 oct. 2019). ISSN : 2470-9476. DOI : 10.1126/scirobotics.aaw9710.
- [87] K. MC GUIRE. *Introduction to the Crazyflie*. 27 avr. 2021. URL : <https://www.bitcraze.io/about/events/epfl2021/>.
- [88] D. MELLINGER et V. KUMAR. « Minimum Snap Trajectory Generation and Control for Quadrotors ». In : *2011 IEEE International Conference on Robotics and Automation*. 2011 IEEE International Conference on Robotics and Automation. Mai 2011, p. 2520-2525. DOI : 10.1109/ICRA.2011.5980409.
- [89] T. MIKI et al. « Learning Robust Perceptive Locomotion for Quadrupedal Robots in the Wild ». In : *Science Robotics* 7.62 (), eabk2822. DOI : 10.1126/scirobotics.abk2822.

- [90] MINH-DUC HUA et al. « A Control Approach for Thrust-Propelled Underactuated Vehicles and Its Application to VTOL Drones ». In : *IEEE Transactions on Automatic Control* 54.8 (août 2009), p. 1837-1853. ISSN : 0018-9286, 1558-2523. DOI : 10.1109/TAC.2009.2024569.
- [91] A. MOLCHANOV et al. « Sim-to-(Multi)-Real : Transfer of Low-Level Robust Control Policies to Multiple Quadrotors ». In : *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Macau, China : IEEE, nov. 2019, p. 59-66. ISBN : 978-1-72814-004-9. DOI : 10.1109/IROS40897.2019.8967695.
- [92] M. A. Z. MORA et al. « PODS : Policy Optimization via Differentiable Simulation ». In : *Proceedings of the 38th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, 1<sup>er</sup> juill. 2021, p. 7805-7817.
- [93] P. MORIN et P. BIDAUD. « Aerial Robotics : A Birds'-Eye View ». In : *AerospaceLab Journal* Issue 8 (2014), 2 pages. DOI : 10.12762/2014.AL08-00.
- [94] A. MORRIS et al. « Recent Developments in Subterranean Robotics ». In : *Journal of Field Robotics* 23.1 (jan. 2006), p. 35-57. ISSN : 1556-4959, 1556-4967. DOI : 10.1002/rob.20106.
- [95] M. W. MUELLER, M. HAMER et R. D'ANDREA. « Fusing Ultra-Wideband Range Measurements with Accelerometers and Rate Gyroscopes for Quadrocopter State Estimation ». In : *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015 IEEE International Conference on Robotics and Automation (ICRA). Mai 2015, p. 1730-1736. DOI : 10.1109/ICRA.2015.7139421.
- [96] M. W. MUELLER, M. HEHN et R. D'ANDREA. « Covariance Correction Step for Kalman Filtering with an Attitude ». In : *Journal of Guidance, Control, and Dynamics* 40.9 (1<sup>er</sup> sept. 2017), p. 2301-2306. DOI : 10.2514/1.G000848.
- [97] M. W. MÜLLER. « Increased Autonomy for Quadrocopter Systems : Trajectory Generation, Fail-Safe Strategies and State Estimation ». ETH Zurich, 2016, 211 p. DOI : 10.3929/ETHZ-A-010655275.
- [98] T. NAKATA et al. « Aerodynamic Imaging by Mosquitoes Inspires a Surface Detector for Autonomous Flying Vehicles ». In : *Science* 368.6491 (8 mai 2020), p. 634-637. DOI : 10.1126/science.aaz9634.
- [99] T. NESTMEYER et al. « Decentralized Multi-target Exploration and Connectivity Maintenance with a Multi-robot System ». In : RSS 2015 Workshop : Reviewing the Review Process. Juill. 2015, p. 1-8.
- [100] D.-T. NGUYEN, M. FILIPPONE et P. MICHIARDI. « Exact Gaussian Process Regression with Distributed Computations ». In : *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. SAC '19. New York, NY, USA : Association for Computing Machinery, 8 avr. 2019, p. 1286-1295. ISBN : 978-1-4503-5933-7. DOI : 10.1145/3297280.3297409.

- [101] S. OZANA et al. « A Comparative Survey Of Software Computational Tools In The Field Of Optimal Control ». In : *2021 23rd International Conference on Process Control (PC)*. 2021 23rd International Conference on Process Control (PC). Juin 2021, p. 284-289. DOI : [10.1109/PC52310.2021.9447510](https://doi.org/10.1109/PC52310.2021.9447510).
- [102] D. PALOSSI et al. « A 64-mW DNN-Based Visual Navigation Engine for Autonomous Nano-Drones ». In : *IEEE Internet of Things Journal* 6.5 (oct. 2019), p. 8357-8371. ISSN : 2327-4662, 2372-2541. DOI : [10.1109/JIOT.2019.2917066](https://doi.org/10.1109/JIOT.2019.2917066).
- [103] A. S. PAUL et E. A. WAN. « RSSI-Based Indoor Localization and Tracking Using Sigma-Point Kalman Smoothers ». In : *IEEE Journal of Selected Topics in Signal Processing* 3.5 (oct. 2009), p. 860-873. ISSN : 1932-4553. DOI : [10.1109/JSTSP.2009.2032309](https://doi.org/10.1109/JSTSP.2009.2032309).
- [104] L. PEDERSEN et al. « A Survey of Space Robotics ». In : *ISAIRAS*. 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space : I-SAIRAS. Nara, Japan, 2003.
- [105] C.-H. PI, W.-Y. YE et S. CHENG. « Robust Quadrotor Control through Reinforcement Learning with Disturbance Compensation ». In : *Applied Sciences* 11.7 (7 jan. 2021), p. 3257. DOI : [10.3390/app11073257](https://doi.org/10.3390/app11073257).
- [106] L. S. PONTRYAGIN. « The Mathematical Theory of Optimal Processes ». In : *CRC Press , FL, USA* (1962).
- [107] C. POWERS et al. « Influence of Aerodynamics and Proximity Effects in Quadrotor Flight ». In : *Experimental Robotics : The 13th International Symposium on Experimental Robotics*. Sous la dir. de J. P. DESAI et al. Springer Tracts in Advanced Robotics. Heidelberg : Springer International Publishing, 2013, p. 289-302. ISBN : 978-3-319-00065-7. DOI : [10.1007/978-3-319-00065-7\\_21](https://doi.org/10.1007/978-3-319-00065-7_21).
- [108] R. PRESTON et J. ROY. « Use of Unmanned Aerial Vehicles to Supplement Conventional Investigation Methods for Underground Open Void Stability and Mitigation ». In : *Proceedings of the First International Conference on Underground Mining Technology*. First International Conference on Underground Mining Technology. Australian Centre for Geomechanics, Perth, 2017, p. 609-616. ISBN : 978-0-9924810-7-0. DOI : [10.36487/ACG\\_rep/1710\\_49\\_Preston](https://doi.org/10.36487/ACG_rep/1710_49_Preston).
- [109] A. QUATERONI. *Méthodes numériques pour le calcul scientifique : programmes en MATLAB*. Collection IRIS. Paris Berlin Heidelberg : Springer, 2000. 444 p. ISBN : 978-2-287-59701-5.
- [110] S. RAJAPPA et al. « Modeling, Control and Design Optimization for a Fully-Actuated Hexarotor Aerial Vehicle with Tilted Propellers ». In : *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015 IEEE International Conference on Robotics and Automation (ICRA). Mai 2015, p. 4006-4013. DOI : [10.1109/ICRA.2015.7139759](https://doi.org/10.1109/ICRA.2015.7139759).
- [111] S. V. RAKOVIĆ et W. S. LEVINE, éd. *Handbook of Model Predictive Control*. Control Engineering. Birkhäuser Basel, 2019. ISBN : 978-3-319-77488-6.
- [112] P. RAMACHANDRAN, B. ZOPH et Q. V. LE. « Searching for Activation Functions ». 27 oct. 2017. arXiv : [1710.05941 \[cs\]](https://arxiv.org/abs/1710.05941).

- [113] A. V. RAO. *A Survey of Numerical Methods for Optimal Control*. 2012.
- [114] R. RASHAD et al. « Fully Actuated Multirotor UAVs : A Literature Review ». In : *IEEE Robotics Automation Magazine* 27.3 (sept. 2020), p. 97-107. ISSN : 1558-223X. DOI : 10.1109/MRA.2019.2955964.
- [115] C. E. RASMUSSEN et C. K. I. WILLIAMS. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. Cambridge, Mass : MIT Press, 2006. 248 p. ISBN : 978-0-262-18253-9.
- [116] J. B. RAWLINGS, D. Q. MAYNE et M. DIEHL. *Model Predictive Control : Theory, Computation, and Design*. T. 2. Nob Hill Publishing Madison, WI, 2017. ISBN : 978-0-9759377-5-4.
- [117] A. RICHARD et al. « A Comprehensive Benchmark of Neural Networks for System Identification ». Sept. 2019.
- [118] R. RICHARDSON et al. « The “Djedi” Robot Exploration of the Southern Shaft of the Queen’s Chamber in the Great Pyramid of Giza, Egypt : Exploration the Great Pyramid - Djedi ». In : *Journal of Field Robotics* 30.3 (mai 2013), p. 323-348. ISSN : 15564959. DOI : 10.1002/rob.21451.
- [119] C. RIZZO et al. « Signal-Based Deployment Planning for Robot Teams in Tunnel-like Fading Environments ». In : *The International Journal of Robotics Research* 32.12 (oct. 2013), p. 1381-1397. ISSN : 0278-3649, 1741-3176. DOI : 10.1177/0278364913501779.
- [120] S. ROSS, G. GORDON et D. BAGNELL. « A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning ». In : *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. JMLR Workshop and Conference Proceedings, 14 juin 2011, p. 627-635.
- [121] F. RUFFIER et al. « Bio-Inspired Optical Flow Circuits for the Visual Guidance of Micro Air Vehicles ». In : *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS ’03*. ISCAS 2003. International Symposium on Circuits and Systems. T. 3. Bangkok, Thailand : IEEE, 2003, p. III-846-III-849. ISBN : 978-0-7803-7761-5. DOI : 10.1109/ISCAS.2003.1205152.
- [122] D. E. RUMELHART, G. E. HINTON et R. J. WILLIAMS. « Learning Representations by Back-Propagating Errors ». In : *Nature* 323.6088 (oct. 1986), p. 533-536. ISSN : 0028-0836, 1476-4687. DOI : 10.1038/323533a0.
- [123] M. RYLL et al. « 6D Physical Interaction with a Fully Actuated Aerial Robot ». In : *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017 IEEE International Conference on Robotics and Automation (ICRA). Singapore, Singapore : IEEE, mai 2017, p. 5190-5195. ISBN : 978-1-5090-4633-1. DOI : 10.1109/ICRA.2017.7989608.
- [124] P. SANCHEZ-CUEVAS, G. HEREDIA et A. OLLERO. « Characterization of the Aerodynamic Ground Effect and Its Influence in Multirotor Control ». In : *International Journal of Aerospace Engineering* (2017). DOI : 10.1155/2017/1823056.

- [125] M. SAUTER. *From GSM to LTE : An Introduction to Mobile Networks and Mobile Broadband*. Chichester, West Sussex, U.K : Wiley, 2011. 414 p. ISBN : 978-0-470-97822-1 978-0-470-66711-8.
- [126] A. SAVITZKY et M. J. E. GOLAY. « Smoothing and Differentiation of Data by Simplified Least Squares Procedures. » In : *Analytical Chemistry* 36.8 (1<sup>er</sup> juill. 1964), p. 1627-1639. ISSN : 0003-2700, 1520-6882. DOI : 10.1021/ac60214a047.
- [127] J. SCHULMAN et al. « Proximal Policy Optimization Algorithms ». 19 juill. 2017. arXiv : 1707.06347 [cs].
- [128] G. SHI et al. « Neural Lander : Stable Drone Landing Control Using Learned Dynamics ». In : *2019 International Conference on Robotics and Automation (ICRA)*. 2019 International Conference on Robotics and Automation (ICRA). Montreal, QC, Canada : IEEE, mai 2019, p. 9784-9790. ISBN : 978-1-5386-6027-0. DOI : 10.1109/ICRA.2019.8794351.
- [129] D. SHUKLA et N. KOMERATH. « Multirotor Drone Aerodynamic Interaction Investigation ». In : *Drones* 2.4 (4 déc. 2018), p. 43. DOI : 10.3390/drones2040043.
- [130] G. SILANO et L. IANNELLI. « CrazyS : A Software-in-the-Loop Simulation Platform for the Crazyflie 2.0 Nano-Quadcopter ». In : *Robot Operating System (ROS) : The Complete Reference (Volume 4)*. Sous la dir. d'A. KOUBA. Studies in Computational Intelligence. Cham : Springer International Publishing, 2020, p. 81-115. ISBN : 978-3-030-20190-6. DOI : 10.1007/978-3-030-20190-6\_4.
- [131] D. SIMON. *Optimal State Estimation : Kalman, H [Infinity] and Nonlinear Approaches*. Hoboken, N.J : Wiley-Interscience, 2006. 526 p. ISBN : 978-0-471-70858-2.
- [132] J. O. SMITH. *Introduction to Digital Filters with Audio Applications*. W3K Publishing, 2007. ISBN : 978-0-9745607-1-7.
- [133] J. SOLÀ, J. DERAY et D. ATCHUTHAN. « A Micro Lie Theory for State Estimation in Robotics ». 8 déc. 2021. arXiv : 1812.01537 [cs].
- [134] Y. SONG et D. SCARAMUZZA. « Policy Search for Model Predictive Control With Application to Agile Drone Flight ». In : *IEEE Transactions on Robotics* (2022), p. 1-17. ISSN : 1552-3098, 1941-0468. DOI : 10.1109/TR0.2022.3141602.
- [135] E. STUMP et al. « Visibility-Based Deployment of Robot Formations for Communication Maintenance ». In : *2011 IEEE International Conference on Robotics and Automation*. 2011 IEEE International Conference on Robotics and Automation (ICRA). Shanghai, China : IEEE, mai 2011, p. 4498-4505. ISBN : 978-1-61284-386-5. DOI : 10.1109/ICRA.2011.5980179.
- [136] L. STUTTERS et al. « Navigation Technologies for Autonomous Underwater Vehicles ». In : *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38.4 (juill. 2008), p. 581-589. ISSN : 1558-2442. DOI : 10.1109/TSMCC.2008.919147.
- [137] Q. SUN et al. « Toward Long-Term Sailing Robots : State of the Art From Energy Perspectives ». In : *Frontiers in Robotics and AI* 8 (5 jan. 2022), p. 787253. ISSN : 2296-9144. DOI : 10.3389/frobt.2021.787253.

- [138] R. TEDRAKE. *Underactuated Robotics : Algorithms for Walking, Running, Swimming, Flying, and Manipulation - Chapter 8*. 2021. URL : [https://underactuated.mit.edu/lqr.html#finite\\_horizon\\_nonlinear](https://underactuated.mit.edu/lqr.html#finite_horizon_nonlinear) (visité le 01/04/2022).
- [139] R. TEDRAKE. *Underactuated Robotics : Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832)*. 2021. URL : <http://underactuated.mit.edu/trajopt.html>.
- [140] S. THRUN et al. « Autonomous Exploration and Mapping of Abandoned Mines ». In : *IEEE Robotics & Automation Magazine* 11.4 (déc. 2004), p. 79-91. ISSN : 1070-9932. DOI : [10.1109/MRA.2004.1371614](https://doi.org/10.1109/MRA.2004.1371614).
- [141] T. TOMIĆ, M. MAIER et S. HADDADIN. « Learning Quadrotor Maneuvers from Optimal Control and Generalizing in Real-Time ». In : *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014 IEEE International Conference on Robotics and Automation (ICRA). Mai 2014, p. 1747-1754. DOI : [10.1109/ICRA.2014.6907087](https://doi.org/10.1109/ICRA.2014.6907087).
- [142] K. P. VALAVANIS et G. J. VACHTSEVANOS. *Handbook of Unmanned*. New York : Springer, 2014. ISBN : 978-90-481-9706-4 978-90-481-9707-1 978-90-481-9708-8.
- [143] E. VINOGRADOV et S. POLLIN. « Tutorial : Wireless Communications with Unmanned Aerial Vehicles ». In : (2019). DOI : [10.13140/RG.2.2.15185.38249](https://doi.org/10.13140/RG.2.2.15185.38249).
- [144] A. WÄCHTER. « Short Tutorial : Getting Started With Ipopt in 90 Minutes ». In : *Combinatorial Scientific Computing*. Sous la dir. d'U. NAUMANN et al. Dagstuhl Seminar Proceedings 09061. Dagstuhl, Germany : Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009.
- [145] A. WÄCHTER et L. T. BIEGLER. « On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming ». In : *Mathematical Programming* 106.1 (1<sup>er</sup> mars 2006), p. 25-57. ISSN : 1436-4646. DOI : [10.1007/s10107-004-0559-y](https://doi.org/10.1007/s10107-004-0559-y).
- [146] S. WASLANDER et al. « Multi-Agent Quadrotor Testbed Control Design : Integral Sliding Mode vs. Reinforcement Learning ». In : *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems. Août 2005, p. 3712-3717. DOI : [10.1109/IROS.2005.1545025](https://doi.org/10.1109/IROS.2005.1545025).
- [147] R. WEI, R. BEARD et E. ATKINS. « Information Consensus in Multivehicle Cooperative Control ». In : *IEEE Control Systems* 27.2 (avr. 2007), p. 71-82. ISSN : 1066-033X, 1941-000X. DOI : [10.1109/MCS.2007.338264](https://doi.org/10.1109/MCS.2007.338264).
- [148] L. WHITCOMB et al. « Advances in Underwater Robot Vehicles for Deep Ocean Exploration : Navigation, Control, and Survey Operations ». In : *Robotics Research*. Sous la dir. de J. M. HOLLERBACH et D. E. KODITSCHEK. London : Springer London, 2000, p. 439-448. ISBN : 978-1-4471-1254-9 978-1-4471-0765-1. DOI : [10.1007/978-1-4471-0765-1\\_53](https://doi.org/10.1007/978-1-4471-0765-1_53).
- [149] S. WITHROW et al. « An Advanced Mars Helicopter Design ». In : *ASCEND 2020*. ASCEND 2020. Virtual Event : American Institute of Aeronautics and Astronautics, 16 nov. 2020. ISBN : 978-1-62410-608-8. DOI : [10.2514/6.2020-4028](https://doi.org/10.2514/6.2020-4028).

- [150] YUANTENG PEI, M. W. MUTKA et NING XI. « Coordinated Multi-Robot Real-Time Exploration with Connectivity and Bandwidth Awareness ». In : *2010 IEEE International Conference on Robotics and Automation*. 2010 IEEE International Conference on Robotics and Automation (ICRA 2010). Anchorage, AK : IEEE, mai 2010, p. 5460-5465. ISBN : 978-1-4244-5038-1. DOI : [10.1109/ROBOT.2010.5509803](https://doi.org/10.1109/ROBOT.2010.5509803).
- [151] T. ZHANG et al. « Learning Deep Control Policies for Autonomous Aerial Vehicles with MPC-guided Policy Search ». In : *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016 IEEE International Conference on Robotics and Automation (ICRA). Stockholm, Sweden : IEEE, mai 2016, p. 528-535. ISBN : 978-1-4673-8026-3. DOI : [10.1109/ICRA.2016.7487175](https://doi.org/10.1109/ICRA.2016.7487175).
- [152] Q. ZHAO, J. HUGHES et D. M. LYONS. « Drone Proximity Detection via Air Disturbance Analysis ». In : *Unmanned Systems Technology XXII*. Unmanned Systems Technology XXII. Sous la dir. de C. M. SHOEMAKER, P. L. MUENCH et H. G. NGUYEN. Online Only, United States : SPIE, 23 avr. 2020, p. 29. ISBN : 978-1-5106-3627-9 978-1-5106-3628-6. DOI : [10.1111/12.2556385](https://doi.org/10.1111/12.2556385).