HAL
open science

Trajectory planning and control of collaborative systems : Application to trirotor UAVS.

Etienne Servais

To cite this version:

Etienne Servais. Trajectory planning and control of collaborative systems : Application to trirotor UAVS.. Automatic. Université Paris Sud - Paris XI, 2015. English. NNT : 2015PA112188 . tel-01252904

HAL Id: tel-01252904

https://theses.hal.science/tel-01252904v1

Submitted on 8 Jan 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ PARIS-SUD

ÉCOLE DOCTORALE : Sciences et Technologie de l'Information, des Télécommunications et des Systèmes

Laboratoire des Signaux et Systèmes (L2S)
Centre de Robotique MinesParisTech (CAOR)

DISCIPLINE : Physique

THÈSE DE DOCTORAT

Présentée et soutenue publiquement par

Étienne SERVAIS

le 18 septembre 2015.

TRAJECTORY PLANNING AND CONTROL OF COLLABORATIVE SYSTEMS: APPLICATION TO TRIROTOR UAVS.

Directeurs de thèse :	Brigitte d'ANDRÉA-NOVEL	Professeur (Mines ParisTech)
	Hugues MOUNIER	Professeur (Université Paris XI)
Composition du jury :		
Rapporteurs :	Tarek HAMEL	Professeur (Université de Nice Sophia Antipolis)
	Miroslav KRSTIĆ	Professeur (Université de Californie à San Diego)
Examinateurs :	Jean-Michel CORON	Professeur (Université Pierre et Marie Curie)
	Joachim RUDOLPH	Professeur (Université de la Sarre)
	Claude SAMSON	Directeur de recherche (INRIA)
Membres invités :	Silviu-Iulian NICULESCU	Directeur de recherche (CNRS)
	Arnaud QUADRAT	Ingénieur (Sagem-DS)

Planification de trajectoire et contrôle d'un système collaboratif : Application à un drone trirotor.

Résumé : L'objet de cette thèse est de proposer un cadre complet, du haut niveau au bas niveau, de génération de trajectoires pour un groupe de systèmes dynamiques indépendants. Ce cadre, basé sur la résolution de l'équation de Burgers pour la génération de trajectoires, est appliqué à un modèle original de drone trirotor et utilise la platitude des deux systèmes différentiels considérés.
La première partie du manuscrit est consacrée à la génération de trajectoires. Celle-ci est effectuée en créant formellement, par le biais de la platitude du système considéré, des solutions à l'équation de la chaleur. Ces solutions sont transformées en solution de l'équation de Burgers par la transformation de Hopf-Cole pour correspondre aux formations voulues. Elles sont optimisées pour répondre à des contraintes spécifiques. Plusieurs exemples de trajectoires sont donnés.
La deuxième partie est consacrée au suivi autonome de trajectoire par un drone trirotor. Ce drone est totalement actionné et un contrôleur en boucle fermée non-linéaire est proposé. Celui-ci est testé en suivant, en roulant, des trajectoires au sol et en vol. Un modèle est présenté et une démarche pour le contrôle est proposée pour transporter une charge pendulaire.

Mots clés : Commande par platitude, génération de trajectoires, équation de Burgers', systèmes multi-agent, optimisation par essaims particulaires, drone, rotors inclinables, commande non-linéaire, transport de charge pendulaire.

Trajectory planning and control of collaborative systems: Application to trirotor UAVs.

Abstract: This thesis is dedicated to the creation of a complete framework, from high-level to low-level, of trajectory generation for a group of independent dynamical systems. This framework, based for the trajectory generation, on the resolution of Burgers equation, is applied to a novel model of trirotor UAV and uses the flatness of the two levels of dynamical systems.
The first part of this thesis is dedicated to the generation of trajectories. Formal solutions to the heat equation are created using the differential flatness of this equation. These solutions are transformed into solutions to Burgers' equation through Hopf-Cole transformation to match the desired formations. They are optimized to match specific requirements. Several examples of trajectories are given.
The second part is dedicated to the autonomous trajectory tracking by a trirotor UAV. This UAV is totally actuated and a nonlinear closed-loop controller is suggested. This controller is tested on the ground and in flight by tracking, rolling or flying, a trajectory. A model is presented and a control approach is suggested to transport a pendulum load.

Keywords: Differential flatness, trajectory generation, Burgers' equation, multi-agent systems, particle swarm optimization, UAV, tilting rotors, non-linear control, pendulum load transportation.

License

This work is, unless otherwise noticed, the sole work of its author and is released, unless otherwise noticed, under the license Creative Commons Attribution-Non Commercial 4.0 International (CC BY-NC 4.0).

As a consequence, you are free to:

Share — copy and redistribute this dissertation in any medium or format.

Adapt — remix, transform, and build upon the material released under this license.

As long as you conform to the three following terms:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor, *i.e.* the author, endorses you or your use.

NonCommercial — You may not use the material for commercial purposes.

Respect of other licenses — You must conform to the licenses applicable to contents of this dissertation created by other authors.

Please refer to this work as:

Servais, Étienne. Trajectory planning and control of collaborative systems: applications to trirotor UAVs. PhD thesis, 2015, Université Paris-Sud (France).

Contents

License i

Contents iii

List of Figures vi

Remerciements ix

Introduction xi

Résumé étendu en français 1

I Trajectory planning for multi-agent systems 15

1 Motion planning for multi-agent systems, an overview 17
- 1.1 Different methods of managing a multi-agent system . . . 19
  - 1.1.1 Behavioral methods . . . 20
  - 1.1.2 Methods based on Particle Swarm Optimization . . . 22
  - 1.1.3 Potential methods . . . 24
  - 1.1.4 Graph based methods . . . 27
  - 1.1.5 Methods based on Partial Differential Equations . . . 28
- 1.2 Different problems of collaborative systems . . . 31
  - 1.2.1 Deployment problems . . . 31
  - 1.2.2 Cooperative transportation of a swinging load . . . 35
  - 1.2.3 Collision avoidance on determined paths . . . 38

2 Trajectory generation for PDE systems, existing works and available tools 41
- 2.1 Existing works: an overview . . . 41
- 2.2 Differential flatness and Gevrey functions . . . 47
  - 2.2.1 Gevrey functions, definition and examples . . . 47
  - 2.2.2 Efficient computation of the derivatives of Φ_σ . . . 49

3 Formal solutions to the heat equation 53
- 3.1 Rewriting the heat equation with formal differential operators of infinite order . 53
- 3.2 The heat equation with controls on both sides . . . 55
  - 3.2.1 Objectives . . . 55
  - 3.2.2 Formal derivation . . . 55
  - 3.2.3 Computational implementation . . . 61

3.3 Various properties of the T_x operator . . . 62
3.3.1 Polynomial states and controls . . . 62
3.3.2 Application to the Weierstrass approximation theorem . . . 66
3.3.3 Convergence of the T_x operator . . . 67
3.3.4 Application of T_x to exponential functions . . . 69
3.3.5 The operator T_x and product of functions . . . 73
3.4 The Hopf-Cole transformation and Gevrey functions . . . 79

4 PDE-based motion planning framework 81
4.1 Generating solutions to the heat equation . . . 81
4.2 Generating solutions to Burgers' equation . . . 85
4.2.1 Optimization of the trajectories . . . 85
4.2.2 Leaders and followers . . . 87
4.2.3 Transition between successive steps . . . 87
4.3 Combining solutions to create trajectories . . . 89
4.4 Conclusion and perspectives . . . 91

II Modeling and control of a trirotor UAV 93

5 Unmanned Aerial Vehicles: a brief review 95
5.1 Flight dynamics . . . 95
5.1.1 Basics of flight . . . 95
5.1.2 The roll-pitch-yaw convention and aircrafts' centers . . . 97
5.1.3 Fixed-wing aircrafts . . . 98
5.1.4 Rotary-wing aircrafts . . . 100
5.1.5 Convertible airplanes and other VTOL aircrafts . . . 104
5.2 A usual model, the quadrotor . . . 105
5.2.1 Design and basic model of a quadrotor helicopter . . . 106
5.2.2 Control and applications for a quadrotor UAV . . . 108
5.3 Tilting rotor multirotor UAVs . . . 115

6 The tricopter: an agile UAV 117
6.1 Design of the tricopter . . . 118
6.1.1 Mechanics design . . . 118
6.1.2 Electronics design . . . 120
6.1.3 The motion capture platform . . . 122
6.2 Mechanical model of the tricopter . . . 122
6.2.1 Formalism and assumptions . . . 122
6.2.2 Screws acting on the tricopter . . . 126
6.2.3 Equations of motion . . . 130
6.3 Introduction to flatness based control . . . 130

7 Various applications of the tricopter 133
7.1 Simulation platform, test trajectory and control approach . . . 133
7.1.1 The simulation platform . . . 133
7.1.2 The test trajectory . . . 135

7.1.3 The control approach 136
7.2 The rolling tricopter 138
7.2.1 Motivations and mechanical modifications 138
7.2.2 Controlling the rolling tricopter, a flatness-based control approach 139
7.2.3 Admissible trajectories and open-loop simulations 142
7.2.4 Experiments 145
7.3 The flying tricopter 151
7.3.1 Position and attitude stabilization 153
7.3.2 Altitude tracking 154
7.3.3 Trajectory tracking 155
7.4 Carrying a load 159
7.4.1 The pendulum load 159
7.4.2 Dynamics of the system 161
7.5 Perspectives 171

Conclusion **173**

Bibliography **175**

List of Figures

1.1 Example of potential force. 25

2.1 The heated rod of [Laroche et al., 2000] 42
2.2 Shock-like equilibrium profiles for Burgers' equation 45
2.3 Normalized value of $\varphi_\sigma(t)$ for various value of σ. 48
2.4 The basis function $\Phi_\sigma(t)$ for various values of σ. 49

3.1 First odd order Bernoulli polynomials used by the T_x operator and by T_{1-x} 58
3.2 Evolution of a polynomial state, 3d view. 65
3.3 Evolution of a polynomial state, cross-sectional view. 66
3.4 Numerical evaluation of the first a_k functions. 72
3.5 Numerical evaluation of the error according to γ. 73
3.6 Numerical evaluation of $t_\gamma(x)$. 74
3.7 The first $d_n(x)$ functions. 76
3.8 Numerical evaluation of $|\Phi_\sigma^{(1)}(t)\Sigma(x,t)|$. 79

4.1 Example of controls. 84
4.2 Numerical resolution of the heat equation. 84
4.3 Time evolution of the cost for a team of four leaders. 88
4.4 Three return possibilities. 89
4.5 Two solutions to Burgers' equation for 3 leading agents and 18 follower agents. 90
4.6 Trajectories in a plane of 3 leading agents and 18 follower agents. 90
4.7 Trajectories of the three leaders and of two chosen followers. 92

5.1 Aerodynamic forces: lift and drag on a foil (NACA4612) 96
5.2 Geometry of an airfoil (NACA4612) 97
5.3 The roll-pitch-yaw convention. 98
5.4 The different elements of an aeroplane. 99
5.5 Dissymetry of lift. 101
5.6 Various helicopter configurations. 102
5.7 Bell Boeing V-22 Osprey. 105
5.8 Control of a quadrotor. 108

6.1 Tricopter geometry. 119
6.2 Axial view of the first nacelle. 119
6.3 The tricopter (courtesy of D. Kastelan). 121
6.4 Example of position and attitude estimate of a still laying object as seen by the motion tracking system. 123

6.5 Delay of the motion tracking system. 123
6.6 Tricopter geometry and reference frames [Kastelan et al., 2015]. 124
6.7 The crane 131

7.1 The graphical model of the tricopter developed for use in FlightGear Flight Simulator.134
7.2 Reference trajectories. 135
7.3 The reference path 136
7.4 The rolling base used by the tricopter. 139
7.5 Influence of the choice of T_z on saturations. 146
7.6 Differentiation of noisy data 147
7.7 Discrepancy of the derivation 148
7.8 Evolution of the reference trajectories, their real values and the error. 149
7.9 Reference and real paths. 149
7.10 Evolution of velocity and acceleration of the rolling tricopter. 150
7.11 Yaw tracking on the ground 151
7.12 Evolution of the controls of the rolling tricopter. 152
7.13 Position and attitude stabilization. 153
7.14 Controls for position and attitude flight stabilization. 154
7.15 Tracking an increase in altitude reference. 155
7.16 Tracking a trajectory in level flight. 156
7.17 Trajectory tracking in level flight: xy path. 156
7.18 Reference and experimental value of the yaw during level flight 157
7.19 Reference and experimental value of the roll and pitch angles 157
7.20 Controls for trajectory tracking in level flight. 158
7.21 The pendulum 159
7.22 The load orientation measurement board. 161
7.23 Schematics of the Pendulum. 162
7.24 Validation of the parameters of the potentiometers. 162
7.25 Schematics of the tricopter carrying a pendulum load. 163
7.26 Axis-angle representation of the pendulum load. 165
7.27 Trajectory followed by the tricopter with pendulum load. 170
7.28 Feedforward for the thrusts and the tilt angles when flying the pendulum. 170

Remerciements

La soutenance de cette thèse est l'aboutissement de trois années de travail particulièrement intense. J'ai conscience que cette réussite doit beaucoup à de nombreuses personnes et je tiens ici à les remercier et leur rendre un hommage nécessaire.

Je tiens en tout premier lieu à remercier mon jury. Mes directeurs, Brigitte et Hugues, qui m'accompagnèrent durant ces trois années, orientèrent mon travail, soutinrent mes idées et mes efforts et me permirent ainsi d'accomplir plus que je ne croyais possible. J'ai la chance que deux grands chercheurs, Miroslav Krstić et Tarek Hamel, acceptent de rapporter cette thèse. J'eu le bonheur de les rencontrer à plusieurs reprises durant ces trois années et je leur sais sincèrement gré et me sens particulièrement honoré de leurs rapports. Tous les membres de mon jury m'honorent également en acceptant d'y siéger. Un grand merci donc à Jean-Michel Coron, Silviu-Iulian Niculescu, Arnaud Quadrat, Joachim Rudolph et Claude Samson. Je tiens parmi eux à remercier tout particulièrement Joachim Rudolph qui m'accueillit à l'occasion de plusieurs séjours dans son laboratoire à Sarrebruck. Ces séjours me permirent de réaliser toutes les expérimentations de la deuxième partie de cette thèse et peu aurait été possible sans cela. Je tiens également à remercier toute l'équipe de son laboratoire pour l'accueil chaleureux et sa gentillesse ; tout particulièrement David Kastelan, Matthias Konz et Frank Paulus-Rieth avec qui je travaillai durant mes séjours.

En France, ma thèse était hébergée simultanément par deux laboratoires qui furent tous deux importants à mes yeux. Au L2S, je tiens à exprimer toutes mon amitié et ma reconnaissance pour ces années ensemble. Le hasard (et surtout l'occupation des bureaux à mon arrivée en cours d'année) a voulu que je sois placé dans un bureau de la division « Télécoms » au sein de laquelle je garde de nombreux amis. Je pense notamment à Pierre, Elsa et François dont je partageais le bureau jusqu'à nos départs respectifs, à Olivier, Benjamin, Jean-François, qui m'introduisirent au sein de l'équipe de foot. J'ai une pensée au sein de cette équipe pour les Momos – le Grand et le Petit – pour Amir, Jérôme, Romaric, Djawad et tous ceux, trop nombreux pour être cités ici, avec qui je jouais deux fois par semaine dans le « cratère ». Merci aussi à Fred et Franck pour leur aide alors que je sais ne pas être le plus facile des doctorants. Merci à Maryvonne pour sa gentillesse, pour m'avoir placé dans ce bureau et avoir ensuite placé dans le bureau voisin une super stagiaire et très bonne amie – et qui au passage me fit rencontrer, par ricochets deux autres bonnes amies[1].

Le CAOR représente une importance au moins égale à mon cœur. Certains m'y connaissaient déjà de mes années dans le cycle ingénieur civil ou de mon activité au sein de Minotaure. Pour autant, j'y rencontrai de très bons amis également. J'ai ainsi une pensée pour Sylvain avec qui je partageais des thématiques de recherche proches et dont le quadrocoptère inspira certains aspects de mon travail. J'ai une pensée également pour Vincent ; nous nous suivons depuis les classes préparatoires et jouions ensemble dans la fanfare de l'École. Merci aussi à Éva – qui me permit à plusieurs reprises de me rhabiller sur le compte de Dopamines. . .promis l'année prochaine je paye ma cotiz. Et, dans le désordre, un grand merci à Axel, JE – promis, juré, je n'ai pas emporté d'ordi DELL –, Jacky, David – tous deux m'apportent une aide depuis mes jeunes années à Minotaure et ils savent que j'apprécie discuter avec eux –, Amaury, Joël – à qui je dois

[1] Oui, Rocío, Teresa, Elena, je parle de vous !

de savoir réparer la prototypeuse 3D du labo et d'être devenu « expert » en CharlyRobot, quelle chance ! –, Arnaud, Séb – qui est peut être presque aussi chiant que moi, ce qui n'est pas peu dire –, Yannick, Hassan, Cyril, Marie-Anne, Alexis, François, Christine et Christophe, Arthur, JF, Jorge, Fabien, Bogdan – qui m'a appris à parler roumain, on verra bien si ce me sera utile. Merci aussi à Alex et Kenzo, nous avons passé suffisamment de soirées avec vous pour que je vous considère dans l'équipe ! Je tiens de plus à remercier tous les membres de l'École que j'ai croisés durant toutes mes années passées dans les locaux. Je pense notamment à mes anciens profs, à Franz et Bernard que je croisais presque tous les soirs et à la direction des études durant mon cycle ingénieur, elle m'a permis de devenir ce que je suis.

Nombreux sont ceux qui savent que je me suis beaucoup investi, au cours de la rédaction de ce manuscrit, sur des détails qui purent leur sembler futils mais qui permirent à la typographie et, plus généralement, à la forme de ce document d'atteindre un niveau dont je tire une fierté qui, je l'espère, est au moins en partie justifiée.

Je tiens en tout premier lieu à remercier Sophie pour sa relecture avisée. Elle a dû, je le crains, remarquer que ma maîtrise de l'anglais n'était pas aussi parfaite que je ne le souhaite. J'en profite pour remercier Macha, qui m'a soutenu durant ces trois années par son amitié constante et qui a, j'en suis sûr, contribué aux corrections effectuées par sa fille. Je tiens à remercier Emmanuel "Manu" à qui je dois une grande partie de la forme de ce manuscrit – notamment une bonne partie de la typographie mathématique. Je savais depuis notre collaboration à l'École au sein de Minotaure qu'il était bon, ses compétences typographiques sont, je m'en rends compte, toutes aussi épatantes que ses qualités de mécanicien, bonne chance pour ta thèse jeune padawan. J'ai eu aussi beaucoup recours à la communauté de tex.sx, merci à ses contributeurs, peu de mes questions sont restées ouvertes grâce à eux. Je tiens aussi à remercier Till Tantau et Christian Feursänger. Toutes les figures de ce manuscrit ont été réalisées grâce à pgf/tikz et pgfplots qu'ils ont créés, merci à eux pour ces formidables outils.

En tout dernier lieu, et pour finir par ceux qui me sont le plus chers, je veux redire merci à toute ma famille et leur répéter combien ils me sont importants. Papa Pierrot, je pense encore à toi, à ce que tu m'as appris, aux livres que tu m'as offerts. Tu étais un super ingénieur et c'est beaucoup à toi que je dois d'avoir eu cette passion. Je suis fier, quelque cinquante ans après ta thèse réalisée, elle-aussi, en partie à Sarrebruck, de suivre tes pas. Merci aussi Pépé, je suis heureux que tu sois encore à mes côtés pour assister à ma soutenance. Tu m'as énormément apporté : cette thèse est aussi celle d'un petit garçon à qui tu as appris à skier, à nager, à travailler le bois. Merci aussi à tous mes frères et sœurs, dans l'ordre : Élisabeth, Jacques, Bernard, Claire, Marie, Agnès, Blandine, vous êtes à tout jamais dans mon cœur. Merci beaucoup à toi Odile, tu m'as aidé tout du long de ma vie et notamment durant mes études, mon succès est en partie le tien, tu es une super marraine ! Et pour finir par le plus important, merci maman, merci papa. Vous m'avez toujours soutenu et je vous dois tout même si vous avez toujours refusé que je vois les choses ainsi. Je n'ai qu'un mot suffisamment fort pour vous dire merci : je vous aime.

See you on the other side...

Introduction

In the past few years, an intense fervor among customers in many countries has arisen for helicopter models of a new type. Indeed while flying, mechanically evolved, and thus expensive helicopter models were long a niche market for passionate hobbyists, technological advances have opened this pastime to the general public in trading the mechanical complexity of swashplates and hinges for the addition of motors. These allowed everyone to discover the passion of flying and hovering.

Discovering the ability to hover, the former outdoor hobby models evolved into indoor toys. The cost of entry level models is nowadays rather low and customers can choose from a broad selection of models. Furthermore, embedded video cameras and high speed data connections turned them into marvelous toys. And as they are small, light and reasonably stable, they can be used by children inside rooms and are nowadays successful birthday and Christmas presents.

In addition to the general public, various professions – firemen, farmers, construction engineers, policemen – have seen the emergence of these platforms as a great opportunity. They give the user a quick overview of a situation which would have been difficult to monitor without these models because of time, money or access constraints. However, in some cases, one may wish to have more than one of these helicopters flying together. Having more than one UAV (Unmanned Aerial Vehicle) is a great advantage to oversee a bigger area, to add redundancy to a system, or to transport a load too heavy for a single helicopter.

In this context, this thesis has two goals. On the one hand, we propose a solution to let several agents – e.g. helicopters, or any other type of agents moving in any number of dimensions, such as wheeled robots, submarines, trains – move and complete tasks together. This is done by proposing an algorithm generating trajectories for multi-agent systems. On the other hand, we address the trajectory tracking problem and the load transportation problem for a novel type of multirotor helicopter.

This work has been conducted during three years, starting in May 2012, both at the Laboratoire des Signaux et Systèmes (L2S) in Gif-sur-Yvette under the supervision of Hugues Mounier and at the Centre de Robotique of Mines ParisTech (CAOR) in Paris under the supervision of Brigitte d'Andréa-Novel. The thesis was part of the project "EcceHomo" (for *Estimation et contrôle commande embarquée pour véhicules en formation – Estimation and embedded control for vehicles in formation*). EcceHomo is a joint project between L2S, CAOR, the Laboratoire d'Informatique of École Polytechnique (LIX) in Palaiseau and the embedded systems department of École Supérieure d'Ingénieur en Électricité et Électronique (ESIEE) and was supported by grants from DIGITEO and Région Île-de-France.

During this time, I was lucky enough to be supported by a grant from the CNRS, a so-called PEPS *Projet Exploratoire Premier Soutien* nick-named "Concorde" (standing for *Contrôle Coopératif de Drones et EDPs, Cooperative Control of UAVs and PDEs*). This grant was awarded to a joint project between L2S, CAOR, the Laboratoire Jacques-Louis Lions (LJLL) of the University of Paris VI and the Lehrstuhl für Systemtheorie und Regelungstechnik (Chair of Systems Theory and Control Engineering, short: LSR) of Saarland University in Saarbrücken (Germany). This grant gave me the opportunity to visit the LSR a few times. I had, during these visits, the opportunity

to develop a sensing module for embedded load state estimation and to conduct experimental validations of my trajectory tracking algorithms on LSR's multirotor UAV prototype, the Tricopter.

As a consequence of the two dimensions of the problem addressed in this work, the present document is organized as follows:

Part I – Trajectory planning for multi-agent systems: The first part is dedicated to the problem of trajectory planning for multi-agent systems. We present the framework we created based on solutions to Burgers' partial differential equation.

- **chapter 1 – Motion planning for multi-agent systems, an overview**: In this first chapter, we introduce the problem of motion planning. Firstly we present various works originating from different fields of research – mostly from mobile robotics. Then we study more specifically path planning solutions for multi-agent systems based on Partial Differential Equations. We then present works based on two famous equations: Burgers' equation and the heat equation.
- **chapter 2 – Trajectory generation for PDE systems, existing works and available tools**: In this chapter, we present various result on the control of Burgers' equation and its link to the heat equation. A short introduction to the theory of differential flatness and Gevrey functions is given.
- **chapter 3 – Formal solutions to the heat equation**: In this chapter, we present the contribution of this thesis in the field of motion planning for multi-agent systems. It is based on the resolution of Burgers' equation using Hopf-Cole transformation and the differential flatness of the resulting heat equation.
- **chapter 4 – PDE-based motion planning framework**: In this last chapter of the first part, we show that the framework developed in the previous chapter can be efficiently used to generate trajectories. We give some numerical results and show some sample of trajectories.

Part II – Modeling and control of a trirotor UAV: This second part is devoted to the description of a novel trirotor UAV. This trirotor helicopter is totally actuated and a flatness based control approach is presented. This approach is applied to trajectory tracking and load transportation.

- **chapter 5 – Unmanned Aerial Vehicles: a brief review**: In this first chapter, we give a broad review of aerial vehicles. After a brief historical and technological review, we concentrate on the current results in the field of control of multirotor helicopter UAV. Several works on control of quadrotors for trajectory tracking, pendulum load and inverted pendulum transportation are presented. The field of tilting rotor multirotor UAV is finally introduced.
- **chapter 6 – The tricopter: an agile UAV**: Here we introduce the trirotor UAV developed at LSR. We start by presenting the experimental platform and introducing its mechanics and electronics design. Then we present the formalism adopted in this thesis for equations of motions and elaborate a mechanical model of the drone.
- **chapter 7 – Various applications of the tricopter**: This chapter presents the various experiments and results conducted with the drone. First, in the aim of reducing

the energy consumption, the tricopter has to track a ground trajectory. This first experiment allows to carefully test the platform and the chosen control approach. We then present trajectory tracking in flight mode. Finally, we add a pendulum load to the tricopter. We present the sensor developed for this new task and present a complete model. We present then a flatness based open-loop controller and show its pertinence in the transport of a pendulum load.

Conclusion: We draw a global conclusion to the work presented in this thesis. We outline the coherence of the chosen approach relying on infinite and finite differentially flat systems to perform, respectively, trajectory planning and tracking and give perspectives for future works.

Résumé étendu en français

Introduction

Au cours des dernières années, une intense ferveur est apparue parmi les consommateurs de nombreux pays pour des maquettes d'hélicoptères d'un nouveau type. En effet, alors que les maquettes volantes, évoluées mécaniquement et, par conséquent, coûteuses, d'hélicoptères ont longtemps été un marché de niche pour des amateurs passionnés, des avancés technologiques ont ouvert ce passe-temps au grand public en remplaçant la mécanique compliquée d'un hélicoptère par l'addition de moteurs. Ceci permit à tout un chacun de découvrir la passion du vol contrôlé.

En acquérant la capacité d'effectuer un vol stationnaire, les maquettes qui étaient auparavant cantonnées aux vols en extérieur se sont muées en jeux d'intérieur. Le prix des modèles d'entrée de gamme est actuellement assez bas et les consommateurs peuvent choisir parmi une vaste gamme de modèles. De plus, des appareils d'acquisition vidéo embarqués et les connexions de données à haut débit les ont transformés en jouets merveilleux. Étant petits, légers et raisonnablement stables, ils peuvent être utilisés par des enfants en intérieur et sont de nos jours des cadeaux à succès pour les anniversaire ou les fêtes.

En plus du grand public, diverses professions – pompiers, exploitants agricoles, ingénieurs des travaux publics, forces de sécurité – ont vu l'émergence de ces plateformes comme une opportunité intéressante. Elles donnent à l'utilisateur un aperçu rapide d'une situation qui aurait été difficile à observer en l'absence de ces modèles, faute de temps, d'argent ou d'un accès aisé. Cependant, en certains cas, d'aucun peut souhaiter disposer de plus qu'un de ces hélicoptères volant de concert. Avoir plus d'un de ces aéronefs sans pilotes (ASP, nous ferons aussi communément usage des appellation drone et UAV suivant en cela l'usage anglo-saxon) est un avantage important pour observer une zone plus grande, pour ajouter de la redondance à un système ou pour transporter une charge trop lourde pour un hélicoptère seul.

Dans ce contexte, cette thèse a deux objectifs. D'une part, nous proposons une solution pour permettre à plusieurs agents – par exemple des hélicoptères, ou tout autre agent se déplaçant en un nombre arbitraire de dimensions, tels que robots à roues, sous-marins, trains – de se déplacer et accomplir une tâche ensemble. Ceci est atteint en proposant un algorithme générant des trajectoires pour des systèmes multi-agents. D'autre part, nous résolvons le problème du suivi de trajectoire et du transport de charge pour un nouveau type d'hélicoptère à rotor multiples.

Ce travail a été réalisé durant trois années, débutant en mai 2012, simultanément au Laboratoire des Signaux et Systèmes (L2S) à Gif-sur-Yvette sous la supervision d'Hugues Mounier et au Centre de Robotique de Mines ParisTech (CAOR) à Paris sous la supervision de Brigitte d'Andréa-Novel. La thèse faisait partie du projet « EcceHomo » (pour Estimation et Contrôle Commande Embarquée pour véhicules en formation). EcceHomo est un projet commun au L2S, au CAOR, au Laboratoire d'Informatique de l'École Polytechnique (LIX) de Palaiseau et au département des systèmes embarqués de l'École Supérieure d'Ingénieur en Électricité et Électronique (ESIEE) et a reçu le soutien financier de DIGITEO et de la Région Île-de-France.

Pendant ces trois années, j'ai eu la chance d'être supporté part un financement du CNRS, en l'occurrence le PEPS (Projet Exploratoire Premier Soutien) « Concorde » (pour Contrôle

Coopératif de Drones et EDPs). Ce financement a été accordé à un projet joint entre le L2S, le CAOR, le Laboratoire Jacques-Louis Lions (LJLL) de l'Université Pierre et Marie Curie (Paris VI) et la Lehrstuhl für Systemtheorie und Regelungstechnik (Chaire de théories des systèmes et des techniques de régulation, LSR) de l'Université de la Sarre à Sarrebruck (Allemagne). Ce financement m'a donné l'opportunité de séjourner au LSR à plusieurs reprises. Durant ces séjours, j'ai développé un module d'acquisition pour l'estimation d'état de la charge transportée par le drone et de conduire des validations expérimentales de mes algorithmes de suivi de trajectoire sur le prototype de drone multirotor développé par le LSR, le Tricoptère.

Conséquence des deux dimensions du problème traité dans ce travail, le présent document est divisé en deux parties distinctes, dédiées d'une part à la planification de trajectoire pour systèmes multi-agents et d'autre part à la modélisation et au contrôle d'un drone trirotor.

Partie I – Planification de trajectoire pour systèmes multi-agents

Chapitre 1 – Introduction à la planification de trajectoire pour systèmes multi-agents

De 1985 à 1989, Michael Girard and Susan Amkraut réalisèrent l'un des premiers films d'animation représentant une nuée d'oiseau[Girard and Amkraut, 1990]. Ce court métrage de 3 minutes 45 seconde représentait une nuée d'une quarantaine d'oiseaux volant dans un temple où elle rejoignait un groupe de danseur dans une chorégraphie commune. La trajectoire des oiseaux était élaborée à partir d'un modèle physique. Chaque oiseau était en effet soumis à des forces répulsives en provenance des autres oiseaux et des obstacles. Simultanément, les oiseaux était attirés vers leurs objectifs prédéfinis par des forces attractives en spirales. La trajectoire de chaque oiseau était ensuite évaluée comme la résolution numérique d'une équation différentielle linéaire ce qui prenait alors environ douze minutes par image.

À la même époque, [Reynolds, 1987] rédigea l'une des premières approches théoriques du problème de l'animation de nuées d'oiseau, de troupeaux ou de bancs de poissons. Il identifia tout d'abord différentes limitations apparaissant lors de la résolution de tels systèmes multi-agents basée sur des modèles de forces.

En effet, alors qu'il n'est pas inhabituel de rencontrer dans certaines régions du monde des essaims de sauterelles contenant plusieurs milliards d'individus, l'une des limites des modèles mathématiques est généralement leur non-scalabilité, ce qui signifie que la complexité de la résolution numérique croît plus vite que le nombre d'agents, ce qui rend difficile la résolution de problèmes avec un grand nombre d'agents. Dans le cas de [Girard and Amkraut, 1990], la difficulté était vraisemblablement au moins quadratique attendu qu'il était nécessaire de calculer pour chaque oiseau les interactions avec tous les autres oiseaux.

D'autres problèmes peuvent survenir durant l'évolution de tels systèmes multi-agents. La formation peut collapser – tous les agents terminent en un point en raison de forces attractives trop intenses – ou se scinder en raison de forces répulsives elles-aussi trop intenses, ce qui dans les deux cas n'est pas désirable.

Reynolds identifie l'usage de forces non bornées pour l'évitement de collision comme l'une des origines de ces problèmes. En effet, de telles forces agissent sans limite de distance même

lorsque l'agent ne se dirige pas vers l'obstacle en question. Il énonce alors que le mouvement en formation obéit aux trois règles suivantes, énoncées par ordre décroissant d'importance, que l'on appelle aujourd'hui *les trois règles de Reynolds* et qui sont

- Éviter les collisions avec les voisins et les obstacles ;
- Apparier sa vitesse à celle de ses voisins ;
- Tant que possible, rester à proximité du centre du groupe.

Néanmoins, l'approche de Reynolds était dédiée à la simulation du monde animal. Tandis que les systèmes multi-agents sont un champ de recherche très vaste recouvrant notamment les grilles de capteurs ou les protocoles de communication, nous nous concentrons sur les systèmes d'agents mobiles.

On peut distinguer parmi les algorithmes de planification deux classes différentes. D'une part, il existe des algorithmes dit de planification de chemin qui résolve un problème géométrique indépendant du temps. D'autre part, on parlera de planification de trajectoires quand l'évolution dynamique du système est prise en compte. Dans ce travail, nous nous consacrons donc au problème de planification coordonnée de trajectoires.

Pour le cas général des systèmes multi-agents, [Beard et al., 2001] ont classifié les méthodes existantes en trois catégories différentes :

Meneur-suiveur : dans ces méthodes, les suiveurs suivent un meneur désigné. Ce meneur peut être global ou local et peut être un meneur virtuel comme par exemple le barycentre de la formation. Ces méthodes permettent aisément de déplacer la formation dans une direction spécifique. Malgré sa simplicité, ces méthodes présentent un certain nombre de problème, comme par exemple la nécessité pour chaque agent de connaître la position du meneur.

Méthodes comportementales : dans ces méthodes, tous les agents ont un rôle identique et se comportent en fonction des informations provenant de leur environnement immédiat. Ces méthodes sont généralement assez simples et décentralisées et par conséquent sont généralement adaptées à de très grands systèmes. Néanmoins les propriétés générales du groupe sont alors difficiles à prévoir.

Structures virtuelles : ces méthodes assignent les agents sur une structure virtuelle pré-établie. Ceci permet de définir aisément un comportement de groupe et de définir des formations choisies mais il est par conséquent difficile de changer la structure.

Cependant, comme nous le verrons dans l'étude suivante, des croisements entre ces différentes techniques existent et certaines ne peuvent réellement se classer dans celles-ci.

Le reste du chapitre est divisé en deux sections. La première section étudie différentes méthodes ayant été utilisées pour définir l'évolution de systèmes multi-agents. Il traite d'abord du cas des méthodes comportementales. Nous étudions ensuite des méthodes basées sur *l'optimisation par essaims particulaires*, un algorithme ou plusieurs agents collaborent pour trouver un « point optimal ». La troisième sous-section décrit les méthodes basées sur les champs de potentiels. Une dernière sous-section présente diverses méthodes faisant usage *d'équations aux dérivées partielles* pour décrire l'évolution d'essaims.

Nous étudions dans la deuxième section trois classes de problèmes liées aux travaux de cette thèse. Dans une première sous-section, nous étudions les problèmes de déploiement, c'est à dire l'évolution planifiée d'une formation entre deux ensembles de points précis. Dans la section suivante, nous étudions des solutions proposées au problème du transport de charge coopératif. Enfin, dans la dernière sous-section, nous décrivons des algorithmes assurant l'évitement de trajectoires pour des agents dont les trajectoires prédéterminées se croisent.

Cette partition est pensée pour donner un aperçu large des différentes méthodes introduites dans la littérature et pour donner des fondements à nos travaux tout en montrant que les problèmes traités peuvent être abordés par plusieurs des méthodes présentées.

Chapitre 2 – Génération de trajectoire pour systèmes d'équation aux dérivées partielles, travaux existants et outils à disposition

De nombreux auteurs, comme nous l'avons montré au chapitre précédent, utilisent une solution à une équation aux dérivées partielles pour générer des trajectoires appliquées à des systèmes multi-agents. Dans ce chapitre nous présentons tout d'abord différents résultats sur la résolution de l'équation de la chaleur et de l'équation de Burgers. Nous introduisons notamment le concept de platitude pour les équations aux dérivées partielles. Dans un deuxième temps, nous introduisons le concept des fonctions de classe Gevrey, une classe de fonctions largement utilisée pour générer des solutions d'une équation aux dérivées partielles plates.

2.1 – États de l'art des méthodes existantes

L'exemple canonique de système plat régi par l'équation de la chaleur est la barre conductrice introduite par [Laroche et al., 2000], présentée figure 2.1 page 42, isolée thermiquement à une extrémité et contrôlée en flux à l'autre. Celle-ci peut être modélisée par le système présenté en équation (2.1) page 41 où $\theta(x,t)$ représente la température à l'instant t à l'emplacement x. Les auteurs proposent alors une méthode pour construire le contrôle $u(t)$ à partir de $\theta(0,t)$. Ils se basent sur les fonctions de classe Gevrey telles que définies en définition 2.1.1 page 42. En utilisant des fonctions $y(t) = \theta(0,t)$ choisies dans une classe de fonctions Gevrey adaptée (théorème 2.1.2 page 43), les auteurs montrent que les fonctions :

$$\theta(x,t) = \sum_{i\geqslant 0} y^{(i)}(t)\frac{x^{2i}}{(2i)!}, \tag{1}$$

sont solutions du système défini par l'équation (2.1) page 41. Autrement dit, la donnée de $\theta(0,t)$ définit intégralement l'évolution du système. La fonction $\theta(0,t)$ est alors appelée sortie plate du système.

Nous présentons alors d'autres résultats, notamment les travaux de [Meurer and Krstić, 2011; Frihauf and Krstic, 2011; Krstić and Smyshlyaev, 2008; Krstić et al., 2008], qui se basent sur différents modèles d'équations aux dérivées partielles et sur l'équation de Burgers. L'équation de Burgers est notamment connue pour la transformation de Hopf-Cole (théorème 2.1.3 page 44) qui permet de la ramener, sous certaines conditions, à l'équation de la chaleur. Des profils à l'équilibre obtenus par [Krstić et al., 2008] sont représentés figure 2.2 page 45. Deux résultats de contrôlabilité sont alors donnés pour l'équation de Burgers en théorèmes 2.1.4 et 2.1.5 page 46 et page 47.

2.2 – Platitude et fonctions Gevrey

Cette section introduit le concept de fonctions Gevrey (définition 2.2.1 page 47) qui sont des fonctions indéfiniment dérivables, dont la série de Taylor peut diverger et dont les dérivées vérifient une majoration de l'accroissement. Ces fonctions permettent de créer des fonctions « bosses » (figure 2.3 page 48) ou permettant des transitions entre deux plateaux (figure 2.4 page 49). Elles sont utilisées notamment pour créer des solutions à des systèmes plats (proposition 2.2.2 page 47). Différents exemples de fonctions Gevrey sont donnés (équations (2.29) à (2.32) page 48 et page 49) et un algorithme de calcul efficace des dérivés de l'une de ces fonctions est présenté (section 2.2.2 page 49).

Chapitre 3 – Solutions formelles à l'équation de la chaleur

Ce chapitre présente notre contribution principale à l'étude de l'équation de la chaleur avec des contrôles aux deux extrémités. En nous basant sur un résultat bien connu de Holmgren, nous dérivons l'expression d'un nouvel opérateur différentiel formel d'ordre infini et étudions son impact sur certaines fonctions de base. Nous étudions son implémentation et sa convergence numérique. Nous présentons finalement les contrôles Gevrey adaptés pour créer des solutions à l'équation de la chaleur avec contrôles aux deux côtés permettant une transition en temps fini entre deux états et présentons une solution pour générer ces contrôles.

3.1 – Réécriture de l'équation de la chaleur grâce aux opérateurs différentiels formels d'ordre infini

L'utilisation du calcul opérationnel permet de réécrire le résultat obtenu en 1908 par Holmgren (équation (3.6) page 54) en une expression concise utilisant des opérateurs différentiels formels d'ordre infini (équation (3.17) page 55) où apparaissent les sorties plates « naturelles » de l'équation de la chaleur que sont la température en $x = 0$, $\varphi_0(t)$ et le flux en cet endroit $\varphi_{x,0}(t)$.

3.2 – L'équation de la chaleur avec contrôle aux deux bords

La sortie plate de l'équation de la chaleur considérée par [Laroche et al., 2000] apparaît comme un choix naturel pour des problèmes de transferts de chaleur. Néanmoins, pour un problème de planification de trajectoires pour un système multi-agents il semble plus raisonnable de paramétrer l'équation de la chaleur en position aux deux bords et montrons comment exprimer cette équation à partir de ces données.

La réécriture formelle de l'équation de la chaleur sous la forme de l'équation (3.20) page 56 fait apparaître un nouvel opérateur différentiel formel d'ordre infini T_x. Nous donnons en proposition 3.2.1 page 56 une expression explicite de cet opérateur. Cette expression est obtenue par une dérivation formelle présentée dans la preuve de cette proposition. En utilisant les résultats suivants (proposition 3.2.2 et lemme 3.2.3 page 58 et page 59), nous démontrons le théorème 3.2.4 page 60 qui prouve que l'équation de la chaleur admet également comme sorties plates les températures aux deux bords $\varphi_0(t)$ et $\varphi_1(t)$. Nous présentons en dernier lieu l'implémentation informatique réalisée pour évaluer l'opérateur T_x et notamment les nombres de Bernoulli qui y apparaissent.

3.3 – Différentes propriétés de l'opérateur T_x

Nous étudions dans cette section différentes propriétés de l'opérateur T_x précédemment introduit. Nous étudions tout d'abord l'action de cet opérateur sur les polynômes réels. Cette étude nous permet de formuler le théorème 3.3.2 page 65. Celui-ci, à partir d'un état de départ polynomial donné, permet de créer des contrôles polynomiaux et de donner une solution explicite et polynomiale à l'équation de la chaleur. Une illustration de ce résultat est donnée figures 3.2 et 3.3 page 65 et page 66. La section 3.3.2 page 67 montre ensuite, en utilisant le théorème d'approximation de Weierstrass, qu'il est possible de construire de tels contrôles pour un état de départ continu arbitraire.

Nous étudions ensuite la convergence de cet opérateur pour d'autres types de fonctions. La proposition 3.3.6 page 68 montre que l'opérateur converge pour les fonctions Gevrey d'ordre 0 et de rayon suffisamment faible. Un exemple de telles fonctions est la fonction exponentielle. L'opérateur lui est donc appliqué construisant ainsi des solutions à l'équation de la chaleur pour des états de départs trigonométriques. Par la suite, des considérations sur la convergence numérique de l'opérateur sont présentées. Finalement, nous donnons le théorème 3.3.9 page 77 qui permet de faire la transition en temps fini entre deux états polynomiaux et donc, par densité, entre deux états continus. Néanmoins, une sommation classique pour l'opérateur conduit à une divergence et un procédé de resommation devrait être employé pour obtenir directement une expression de la solution à l'équation de la chaleur. Cette évaluation est obtenue par une résolution aux éléments finis et nous permet de donner une évaluation du terme « divergent » (figure 3.8 page 79).

Chapitre 4 – Une solution de planification de trajectoires par résolution d'équations aux dérivées partielles

Dans le chapitre précédent, nous avons présenté une méthode formelle pour construire une solution à l'équation de la chaleur avec contrôles aux deux bords. Dans ce chapitre nous présentons une solution pour créer des trajectoires adaptées. Dans une première section nous créons des solutions à l'équation de la chaleur. Dans une seconde section, nous utilisons la solution précédemment évoquée pour générer des solutions à l'équation de Burgers. Différents aspects de cette méthode sont évoqués et nous présentons une méthode adaptée à des systèmes multi-agents composés de meneurs et suiveurs.

4.1 – Génération de solutions à l'équation de la chaleur

Nous considérons tout d'abord le problème de génération de contrôles entre deux états polynomiaux interpolant des formations de départ et d'arrivée unidimensionnelles d'agents. Disposant de degrés de libertés, nous proposons de faire passer les agents aux bords par des points de passage. Des contrôles sont alors générés (figure 4.1 page 84) sur lesquelles l'influence de l'ordre Gevrey de la fonction de transition est montrée. Ceci permet de construire des solutions à l'équation de la chaleur comme montré figure 4.2 page 84.

4.2 – Génération de solutions à l'équation de Burgers

L'utilisation de la transformation de Hopf-Cole pour transformer une solution à l'équation de la chaleur en une solution à l'équation de Burgers fait appel à une division et donc implique un choix approprié de la solution à l'équation de la chaleur. Ce choix est assuré par une optimisation des degrés de liberté disponibles en suivant une fonction de coût adaptée (équation (4.12) page 85). Cette fonction de coût permet d'une part de s'assurer que la solution à l'équation de la chaleur est strictement positive tout en assurant une amplitude minimale pour les agents aux bords de la solution à l'équation de Burgers. Cette optimisation est réalisée par un processus d'optimisation par essaims particulaires. Des scénarios de transition entre formations successives (figure 4.4 page 89) sont donnés et étudiés et différents rôles d'agents, meneurs et suiveurs sont introduits pour permettre de gérer des formations nombreuses sans affecter la complexité numérique de la résolution.

4.3 – Créations de trajectoires

Les solutions créées dans la section précédente sont combinées pour créer des trajectoires en plusieurs dimensions. Différentes représentations d'une trajectoire en deux dimensions avec deux meneurs et dix-neuf suiveurs sont données pour illustrer cette méthode (figures 4.5 à 4.7 page 90 et page 92).

4.4 – Conclusions et perspectives

Dans cette première partie, nous avons tout d'abord présenté une méthode formelle, basée sur la platitude de l'équation de la chaleur pour créer des solutions à celle-ci. Cette méthode est appliquée, en utilisant un processus d'optimisation et la transformation de Hopf-Cole, pour créer des solutions à l'équation de Burgers. Ces solutions sont utilisées comme base à une solution de planification de trajectoires pour systèmes multi-agents.

Cette solution permet de créer des solutions pour un ensemble d'agents, soit meneurs (pour lesquels une position finale peut être choisie arbitrairement), soit suiveurs (dont la position finale est le résultat des positions finales des meneurs), évoluant dans un espace en dimension arbitraire. Le problème de déploiements successifs est considéré et l'efficacité numérique de l'optimisation de ces transitions est étudiée.

Cette solution apparaît efficace. Cependant, des travaux supplémentaires pourraient être menés pour assurer l'évitement d'obstacles et de collisions entre agents. Ces objectifs pourront être ajoutés à la fonction de coût du processus d'optimisation. Le suivi des trajectoires générées doit être réalisé en boucle fermée par des contrôleurs implantés sur les différents agents. Un tel problème va être considéré dans la partie suivante.

Partie II – Modélisation et contrôle d'un drone trirotor

Chapitre 5 – Un court historique des aéronefs sans pilotes

Ce chapitre est dédié à une courte revue des différents concepts d'aéronefs avec une attention particulière apportée aux aérodynes à voilure tournante sans pilotes.

5.1 – Dynamique du vol

Dans la première section, nous rappelons quelques concepts essentiels de la dynamique du vol. Un historique du vol humain nous permet de présenter différentes classes d'aérostats et d'aérodynes. Nous illustrons quelques concepts importants de l'aérodynamique des ailes et pales de rotor, notamment, les forces aérodynamiques (figure 5.1 page 96) et leur géométrie (figure 5.2 page 97). Nous introduisons également la convention d'orientation choisie – les axes « Nord-Est-Bas » pour décrire les angles de « roulis-tangage-lacet » – dans ce travail (figure 5.3 page 98). Nous décrivons rapidement l'histoire et l'évolution des aéronefs à voilure fixe, notamment leurs éléments (figure 5.4 page 99) et l'apparition récente d'aéronefs militaires dit « supermaneuvrables ». Nous introduisons ensuite les différentes classes d'aéronefs à voilure tournante et décrivons leurs caractéristiques techniques essentielles. Nous décrivons notamment le principe de dissymétrie de la poussée qui apparaît spécifiquement sur les rotors d'hélicoptères (figure 5.5 page 101). Nous donnons un rapide aperçu des aérodynes à voilure fixe, à voilure tournante et convertibles et présentons différents modèles existants et comparons leurs caractéristiques. Dans une dernière sous-section, nous introduisons les aéronefs considérés comme « convertibles ». Ceux-ci, tel le Bell Boeing V-22 Osprey (figure 5.7 page 105), peuvent appartenir successivement, en fonction de leur mode de vol, à diverses catégories d'aéronefs.

5.2 – Un modèle usuel, le quadcoptère

Nous consacrons la deuxième section à l'étude des drones quadrirotors. Cette famille de drones est la classe de petits drones multirotors la plus répandue. Sa configuration se rapproche de la structure du drone étudié dans cette thèse. Nous illustrons son contrôle figure 5.8 page 108 et présentons différents travaux qui lui sont consacrés, notamment la modélisation introduite par [Pounds et al., 2002]. Nous présentons ensuite différents travaux consacrés au suivi de trajectoire, notamment par régulateur PID ou LQR sur des modèles linéarisés [Pounds et al., 2002; Bouabdallah et al., 2004; Bouabdallah and Siegwart, 2005, 2007], par platitude [Cowling et al., 2007; Konz and Rudolph, 2013] ou par commande sans modèle [Wang et al., 2011].

Nous présentons ensuite des travaux consacrés au transport de charge par quadcoptère, tout d'abord dans le cas de charge fixe [Pounds et al., 2012; Palunko and Fierro, 2011; Mellinger et al., 2013]. L'expérience tirée du contrôle de charge par grue peut aussi être utilisée, notamment pour atténuer les oscillations d'une charge pendulaire [Bisgaard et al., 2009]. D'autres exemples sont donnés pour permettre le transport de charge pendulaire [Palunko et al., 2012; Sreenath et al., 2013] et peuvent être étendus au cas du transport coopératif [Sreenath and Kumar, 2013].

Le dernier exemple d'applications présenté est le cas du pendule inversé pour lequel des exemples sont introduits [Hehn and D'Andrea, 2011; Lee et al., 2013; Figueroa et al., 2014].

5.3 – Drones multirotors à rotors inclinables

Dans cette dernière section, nous étudions les travaux ayant été réalisés sur des drones multirotors à rotors inclinables. Le premier exemple présenté est un quadcoptère dont les rotors sont tous indépendamment inclinables [Ryll et al., 2012] conduisant, avec ses huit contrôles indépendants à, un système suractionné. Le modèle ensuite présenté [Hua et al., 2012] évite cet écueil en équipant un drone avec une poussée orientable et prouve en simulation la capacité d'un tel aéronef à suivre des trajectoires arbitraires. Une solution est également apportée quand les capacités du drone sont dépassées pour différencier et accomplir objectifs primaires et secondaires. Le quadcoptère de [Thorel and d'Andréa-Novel, 2014] avec un axe inclinable est utilisé au sol. Cette stratégie est utilisée pour permettre une exploration efficace énergétiquement en environnement intérieur mais le modèle présente une singularité rendant sa stabilisation compliquée. D'autres modèles, avec un nombre varié de rotors, sont présentés, notamment des modèles trirotor [Escareño et al., 2008; Mohamed and Lanzon, 2012].

Chapitre 6 – Le tricoptère, un drone agile

Ce chapitre est dédié à la présentation du tricoptère, un drone multirotor à rotors inclinables développé à la Chaire de théorie des systèmes et génie de la commande de l'Université de la Sarre (Allemagne) sous la direction de Joachim Rudolph.

6.1 – Architecture du tricoptère

Dans cette première section, nous décrivons l'architecture du tricoptère. Nous commençons par décrire l'architecture mécanique du tricoptère. Celui-ci se compose, comme illustré figure 6.1 page 119, de trois rotors disposés régulièrement à 120 ° dans un plan. Ainsi que le suggère la figure 6.2 page 119, chaque bras peut s'incliner indépendamment autour de son axe grâce à des servomoteurs offrant une amplitude d'environ 120 °. Les caractéristiques techniques du tricoptère sont rassemblées tableau 6.1 page 120. Nous décrivons ensuite l'électronique embarquée du tricoptère. Il faut notamment noter les capacités du microcontrôleur embarqué dont les caractéristiques élevées permettent de faire, en temps réels, des évaluations numériques précises. On pourra aussi remarquer que le contrôle en inclinaison et en rotation des rotors est assuré par des contrôleurs en boucle ouverte. Les expérimentations sont conduites dans une salle dotée d'un système de suivi de mouvement de marque Vicon dont les capacités en précision et en latence sont illustrées respectivement figures 6.4 et 6.5 page 123, cette dernière figure illustrant également les capacités d'un algorithme de fusion des données du système de suivi de mouvement et de la centrale magnéto-inertielle embarquée sur le drone.

6.2 – Modèle mécanique du tricoptère

Un modèle mécanique du drone est ensuite construit à partir du modèle de poussée et couple présenté équation (6.2) page 124. Nous introduisons notamment le vecteur $\vec{f}_a$ (équation (6.21) page 127) dont sont tirés les contrôles (équation (6.22) page 127). Ceci nous permet d'obtenir les équations du mouvement du tricoptère telles que données équation (6.42) page 130.

6.3 – Introduction à la commande par platitude

Ce chapitre se termine par une introduction succincte à la théorie de la commande par platitude en dimension finie. Nous rappelons la définition d'un système plat (définition 6.3.1 page 131) et nous l'illustrons par un exemple classique de grue tiré de [Fliess et al., 1995] représenté par la figure 6.7 page 131.

Chapitre 7 – Différentes applications du tricoptère

Dans ce chapitre, nous présentons différentes applications du tricoptère. Dans la première section, nous présentons le simulateur et le contrôleur que nous avons développés pour le tricoptère. Afin de tester le modèle réel, nous étudions ensuite le tricoptère au sol en tant que plateforme roulante. Cette première expérience nous permet de tester l'ensemble de la plateforme ainsi que le contrôleur. Nous étudions ensuite le vol autonome du tricoptère et des expériences de suivi de trajectoire. Finalement, nous étudions l'utilisation du tricoptère pour le transport de charge pendulaire.

7.1 – Plateforme de simulations, trajectoire de test et contrôle

Nous présentons tout d'abord la plateforme de simulations développées pour le tricoptère. Celle-ci est composée de différents composants – contrôleur, intégrateur, générateur de trajectoire – développés en C et sont liés par une interface en python. Cette interface permet notamment d'envoyer les résultats de simulation vers un simulateur de vol : FlightGear Flight Simulator ce qui est représenté figure 7.1 page 134. La trajectoire utilisée est représentée figure 7.3 page 136. Elle a été choisie pour permettre des parcours répétés dans l'espace contraint disponible tout en ayant un profil similaire à ceux obtenus dans la première partie de cette thèse. Nous étudions ensuite des dynamiques d'erreur qui nous permettent, en nous servant des résultats de [Kastelan et al., 2015] pour le contrôle de l'orientation, de proposer un contrôleur permettant le contrôle des six degrés de liberté de l'appareil. Ce contrôleur est donné équation (7.18) page 138. Les sections suivantes sont dédiées à l'application de celui-ci.

7.2 – Le tricoptère roulant

Suivant l'exemple de [Thorel, 2014], nous proposons de faire évoluer le tricoptère au sol. Ceci permet de mettre en avant son caractère totalement actionné. Ceci permet également de tester le système en déplacement avant les phases de vol et laisse entrevoir des gains en terme de dépense énergétique, la poussée nécessaire étant réduite. Pour permettre ce déplacement, une base roulante, représentée figure 7.4 page 139 avec les paramètres donnés tableau 7.1 page 139, a été ajoutée au tricoptère. Dans cette configuration, le contrôleur du tricoptère se simplifie en équation (7.30) page 141. Des simulations effectuées à partir du modèle simplifié de translation (équation (7.40) page 145) montrent que la saturation des angles d'inclinaisons des rotors surviennent, pour des poussées verticales faibles, pour des accélérations particulièrement faibles. Cela permet de choisir une poussée verticale adaptée, en l'occurrence 4.3 N, soit environ un tiers de la poussée verticale nécessaire au vol stationnaire.

Ces premières expériences permettent également de mettre au point les outils d'analyse, notamment la reconstruction, hors-ligne, des vitesses et accélérations effectives du drone. Celles-ci s'obtiennent à partir des données de positions absolues, bruitées, et par l'algorithme de régularisation de [Chartrand, 2011] dont l'efficacité est matérialisée par la figure 7.6 page 147 et dont les biais sont illustrés figure 7.7 page 148.

Nous réalisons ensuite l'expérience de suivi de trajectoire. Les résultats de suivi de position sont satisfaisants (figure 7.8 page 149) avec une erreur en position constamment inférieure à 0.1 m. Ces résultats sont matérialisés figure 7.9 page 149. La vitesse et l'accélération sont également convenablement suivies même si des oscillations apparaissent lorsque l'on évalue la norme de celles-ci (figure 7.10 page 150). Dans le même temps, l'angle de lacet, la vitesse et l'accélération angulaire sont très bien suivis (figure 7.11 page 151). Les commandes ainsi que leurs prédictions sont représentées figure 7.12 page 152. Ces premières suivent par moment correctement ces dernières mais à de nombreux endroits, des pics non prévus dans les commandes apparaissent. Ces pics peuvent trouver une explication dans des effets non modélisés, par exemple l'effet de sol qui augmente l'efficacité des commandes en poussée.

7.3 – Le tricoptère volant

Le vol ajoutant deux degrés de liberté en orientation, la représentation par des angles et matrices de rotation induit une ambiguïté. Nous adoptons, pour la résoudre, la représentation décrite par l'équation (7.43) page 153.

Dans un premier temps, nous vérifions la stabilité du tricoptère en vol stationnaire. Les résultats sont encourageants (figure 7.13 page 153) avec des erreurs statiques inférieures respectivement à 1×10^{-1} m et à 2×10^{-2} rad et des écarts aux valeurs moyennes sur la fenêtre de temps considérée inférieurs respectivement à 2×10^{-2} m et 2×10^{-2} rad. Les commandes sont de l'ordre de grandeur des prédictions (figure 7.14 page 154) les rotors 2 et 3 présentant cependant des valeurs inférieures en moyenne à la prédiction.

Le suivi d'une demande d'augmentation de l'altitude est représenté (figure 7.15 page 155) pour deux essais différents. On remarque que durant les deux essais, la vitesse et l'accélération verticales présentent des oscillations tandis que la position présente une erreur statique importante.

Nous réalisons ensuite le suivi de trajectoire par le tricoptère. Les positions, vitesses et accélérations sont convenablement suivies (figure 7.16 page 156), cependant une erreur statique importante en x apparaît. Ceci se manifeste dans la représentation de la trajectoire parcourue (figure 7.17 page 156) ; une fois l'erreur statique corrigée, le suivi apparaît convenable. Le suivi en lacet (figure 7.18 page 157) et la stabilisation en tangage et roulis (figure 7.19 page 157) sont également convenables. Les commandes (figure 7.20 page 158) suivent globalement la forme donnée par les prédictions avec cependant un certain nombre d'écarts.

7.4 – Transport de charge

Cette section est consacré au transport d'une charge pendulaire par le tricoptère. Cette charge, pesant 100 g, est attachée à l'extrémité d'un tube de carbone de longueur 20 cm, lui-même fixé à l'axe d'un capteur industriel de type joystick. L'ensemble est représenté figure 7.21 page 159.

Une carte d'acquisition (figure 7.22 page 161) a été développée pour effectuer l'acquisition de donnée et sa transmission au tricoptère. Les résultats de l'identification des paramètres de l'unité de mesure d'angle sont représentés figure 7.24 page 162.

La dynamique du drone a été modifiée pour aboutir au modèle des équations (7.44) et (7.45) page 161 en se basant sur les nouveaux référentiels de la figure 7.25 page 163. Il est à noter que dans ce modèle, la charge pendulaire est fixé à l'écart du centre de gravité du tricoptère et implique donc une dynamique angulaire modifiée. Nous donnons ensuite les équations cinématiques de la charge, aboutissant à l'équation (7.49) page 163. La cinématique angulaire n'ayant que deux degrés de liberté est également simplifiée pour aboutir aux équations (7.56) et (7.57) page 164. En utilisant la dynamique du pendule (équation (7.60) page 165), nous aboutissons au modèle global de l'équation (7.62) page 166. Nous démontrons que l'inertie réduite du système global est inversible dans toutes les configurations du système (proposition 7.4.1 page 167). Ceci nous permet d'aboutir à l'un de nos principaux résultats (proposition 7.4.2 page 168) qui énonce que le système composé du tricoptère et de notre charge pendulaire est plat avec pour sortie plate l'ensemble constitué de la position du pendule, le lacet du tricoptère et l'orientation de la charge par rapport au tricoptère. Ce résultat nous permet de proposer des commandes en boucle ouvertes (figure 7.28 page 170) afin de suivre avec la charge la trajectoire de référence (figure 7.27 page 170).

7.5 – Perspectives

Dans ce chapitre, nous avons présenté un contrôleur en boucle fermée basé sur la platitude et des applications de celui-ci au contrôle du tricoptère dans deux scénarios. La première application, où le tricoptère doit suivre une trajectoire au sol en roulant sur des roues libres est une bonne illustration du caractère totalement actionné de cette plateforme. Cela nous a permis de tester et vérifier avec soin l'ensemble de la plateforme et, tout particulièrement, le contrôleur. De plus, cette approche peut se révéler intéressante pour l'exploration en intérieur, lorsque le sol est compatible avec la base-roulante. Cela apparaît en effet plus efficace en terme de déplacement. Dans notre cas, nous avons utilisé le tiers de l'énergie nécessaire au vol. Le contrôleur s'est montré efficace en terme de suivi de trajectoire, tant en orientation qu'en positon.

La seconde application est le cas « naturel » du vol. Le contrôleur a montré sa capacité a correctement stabiliser le tricoptère. Certains défauts mineurs pourraient être néanmoins corrigés dans le futur. Par exemple, l'ajout d'un terme intégral au suivi de position pourrait aider à éliminer les différentes erreurs statiques. De plus, comme nos vols expérimentaux ont été effectués dans une petite salle de test, le tricoptère a été particulièrement affecté par des turbulences. Par exemple, le suivi en altitude a été particulièrement perturbé par l'effet de sol dû à la proximité de celui-ci, ce qui a rendu impossibles les décollages autonomes. Une meilleur compréhension et une correction des divers effets aérodynamiques entrant en jeu aideraient à augmenter l'efficacité de notre contrôleur.

Dans le dernier scénario, le tricoptère est équipé d'une charge pendulaire. Un modèle précis est introduit en prenant en compte l'influence de la charge sur le tricoptère. Un contrôleur en boucle-ouverte basé sur la platitude du modèle est proposé pour suivre une position avec la charge. Cette approche pourrait bénéficier de diverses améliorations et études ultérieures. Notamment, la paramétrisation de la matrice de roulis-tangage dans la preuve de la platitude pourrait être améliorée, par exemple en suivant ce qu'a proposé [Konz and Rudolph, 2013]. Le contrôleur en

boucle-ouverte proposé pourrait être utilisé comme un premier pas en direction d'un contrôleur en boucle fermée. De plus, d'autres trajectoires et applications pourraient être étudiées, par exemple en faisant passer le tricoptère par une fenêtre moins haute que la longueur de la charge pendulaire ou le transport coopératif par plusieurs tricoptères de la charge pendulaire.

Conclusion

Le but de cette thèse était d'étudier et de présenter un environnement de planification de trajectoires et le contrôle de systèmes collaboratifs appliqués à un cas particulier : un nouveau drone trirotor.

Dans la première partie de cette thèse, nous avons créé l'environnement pour générer des trajectoires pour un système multi-agents. À l'issue d'une synthèse des diverses solutions existantes de contrôle de systèmes collaboratifs, nous avons examiné le cas de la génération de solutions pour certaines équations aux dérivées partielles.

Notre travail repose sur l'équation de Burgers, une équation unidimensionnelle non-linéaire classique de la mécanique des fluides. Les solutions à l'équation de Burgers peuvent être transformées à l'aide de la transformation de Hopf-Cole en solutions à l'équation de la chaleur. Nous prouvons, sous certaines conditions, que l'équation de la chaleur avec contrôles aux deux extrémités est plate. Ainsi, nous pouvons créer des solutions à l'équation de la chaleur en choisissant les trajectoire des agents aux extrémités de la formation. Nous prouvons divers résultats sur les structures admissibles pour les contrôles et montrons que ces contrôles peuvent être utilisés pour mener l'équation de la chaleur en temps-fini entre différents états arbitraires, et notamment des états non-nuls. Dans un second temps, nous utilisons l'optimisation par essaim particulaires pour optimiser les contrôles et créer des solutions adaptées à l'équation de Burgers. La solution ainsi créée définit les trajectoires d'un système collaboratif. Ce système est composé de meneurs (leaders) et de suiveurs (followers). Les positions des meneurs peuvent être choisies librement et sont utilisées comme contraintes de l'optimiseur. Les positions des suiveurs sont la conséquence de la position des meneurs et des critères d'optimisation. Dans notre environnement, les trajectoires sont construites de telle sorte que les trajectoires des meneurs soient les plus courtes possibles. Nous combinons plusieurs trajectoires unidimensionnelles en une trajectoire multidimensionnelle et montrons un exemple en deux dimensions pour deux meneurs et dix-neuf suiveurs.

Dans la seconde partie de cette thèse, nous considérons le problème du suivi par des drones des trajectoires générées. Après une courte introduction à l'aérodynamique avec une emphase portée sur les drones multirotors, nous présentons plusieurs problèmes actuels dans le domaine des drones tels que le transport de charge pendulaire et le transport coopératif de charge.

Notre travail est dédié au cas spécial d'un drone trirotor : le tricoptère. Ce modèle est le résultat d'études conduites sous la supervision du Professeur Rudolph au LSR (Université de la Sarre, Allemagne). Il a l'avantage spécifique de disposer de six contrôles indépendants. Il est par conséquent complètement actionné et peut suivre n'importe quelle trajectoire raisonnable. Nous présentons l'architecture du tricoptère et notre travail quant à la modélisation et à la simulation de la plateforme. Un approche de contrôle est présentée basée sur la platitude de ce système dynamique et un algorithme de suivi de trajectoire en boucle fermée est présenté.

L'efficacité énergétique des drones est un problème ouvert. Une solution possible dans le cas de l'exploration de bâtiments est de permettre au drone de rouler sur le sol pour économiser la poussée verticale. Ceci est particulièrement facile à appliquer pour notre tricoptère, celui-ci étant totalement actionné. Par conséquent, le contrôleur est tout d'abord testé au sol et suit une trajectoire dans le plan. Le résultat de ces premières expériences est concluant et le tricoptère est ensuite testé pour le suivi de trajectoires aériennes. Les performances du tricoptère sont également concluants dans ce cas. Une dernière application est suggérée : le transport d'une charge pendulaire. Nous prouvons que le tricoptère transportant une charge pendulaire est un système plat. Nous présentons un contrôleur en boucle ouverte pour parcourir les trajectoires voulues et présentons également le module de mesure d'orientation de la charge qui est nécessaire au suivi en boucle fermée des trajectoires.

En conclusion, cette thèse présente une solution complète et cohérente pour des systèmes collaboratifs. Un planificateur de trajectoires haut-niveau est introduit et complété avec le contrôle des agents bas-niveau : un drone trirotor totalement actionné. Néanmoins, différentes améliorations peuvent être envisagées. Par exemple, dans le cas du planificateur de trajectoire haut-niveau, l'optimiseur pourrait être adapté pour permettre l'évidement de collisions et d'obstacles. De plus, la dynamique des agents pourrait être prise en compte, par exemple en utilisant différentes viscosités (le paramètre μ) ou raideurs du contrôle (le paramètre γ) pour les dynamiques verticale et horizontales.

Le contrôle des agents bas-niveau pourrait être également amélioré. Le décollage et l'atterrissage autonomes sont les améliorations les plus évidentes qui pourraient être apportées. Ceci nécessiterait une meilleure compréhension et une meilleure modélisation de divers effets aérodynamiques. Par exemple, l'effet de sol pourrait être pris en compte en utilisant le modèle suggéré par [Johnson, 1994] :

$$\frac{T}{T_\infty} = \frac{1}{1-\alpha/z^2}.$$

ou les modèles plus sophistiqués qui y sont proposés. De plus, un terme intégral pourrait être ajouté à la partie translationnelle du contrôleur de position pour améliorer la qualité du suivi de trajectoire en vol. Dernièrement, les dynamiques des servomoteurs et des rotors pourraient être prises en compte dans le modèle. Ceci conduirait à un modèle dynamique de plus haut degré qui décrirait le tricoptère de manière plus adaptée.

L'une des premières améliorations à apporter au système de transport de charge pendulaire est la création d'un contrôleur en boucle fermée. Il serait également intéressant de proposer un contrôleur pour un pendule inversé qui n'utiliserait que la dynamique en rotation du tricoptère pour stabiliser le pendule tout en stabilisant le tricoptère en position. Ceci est possible avec notre plateforme et est reflété par les équations du modèle dynamique du tricoptère. Ceci conduirait à une amélioration intéressante par rapport aux travaux actuels sur la stabilisation de pendule inversé par drone. Pour finir, proposer une solution de transport coopératif de charge utilisant le tricoptère serait une avancé marquante.

Part I

Trajectory planning for multi-agent systems

CHAPTER 1

Motion planning for multi-agent systems, an overview

Introduction

From 1985 to 1989, Michael Girard and Susan Amkraut worked on one of the first known computer animation of a flock of birds [Girard and Amkraut, 1990]. The 3 minutes 45 seconds long animation staged a flock of about forty birds flying in a temple where they by the end join dancers in a mutual psychedelic choreography. The design of the birds' trajectories was elaborated on a physical model. Each bird was namely subjected to repulsive forces occurring from the other birds and obstacles. The birds were simultaneously driven toward their preset goals by spiral attracting forces. The trajectories of each bird were then computed as the numerical integration of a linear differential equation. The numerical evaluation of each step took then about twelve minutes. Using the trace and the discriminant of the linear equation matrix, Girard&Amkraut were then able to classify the flow patterns – such as e.g. spirals, sinks, sources, saddles and orbits – and to make changes to the matrix to drive the flock according to these patterns.

In the meantime, [Reynolds, 1987] made one of the first theoretical approach toward the problem of animating flocks of birds, herds of land animals and schools of fish. He first identified various limitations appearing while solving such multi-agent systems with forces based models. Indeed, while it is not unusual to encounter in certain region of the world swarms of locusts containing billions of individuals, one of the limits of mathematical models is usually the lack of scalability, meaning that the numerical complexity grows faster than the number of agents making difficult to compute solutions to problems with a lot of agents. In the case of [Girard and Amkraut, 1990], the difficulty was presumably at least quadratic, since for each bird it was necessary to compute forces from all the other birds. Other problems might occur during the evolution of such multi-agent systems. The swarm might either collapse – e.g. all the agents end up in one point due to too intense and unrealistic attracting forces – or split – e.g. the agents gather in several smaller groups, maybe even alone, because of too intense repulsing forces – which is, in both case, not desirable.

Reynolds identifies the use of unbounded forces for collision avoidance as one of the origins of these problems. As such forces act infinitely far, even if the agent is not heading toward the obstacle, they might turn out to be too important when in the neighborhood of other agents. As a consequence, if the global repulsion is too intense, the swarm might be led to split.

Reynolds states then that flocking comes from the three following rules, in decreasing precedence, called today *the three rules of Reynolds*:

Collision Avoidance: avoid collisions with nearby flockmates and obstacles.

Velocity Matching: attempt to match velocity with nearby flockmates.

Flock Centering: attempt to stay close to nearby flockmates.

Velocity matching is in fact already part of the collision avoidance task. Indeed, if all agents moves at the same – vectorial – speed they avoid collisions. However it is more important to avoid collision than to match the velocity of the neighborhood. If it appears necessary to turn to avoid an obstacle, this should be more important than keeping the flock's speed. Moreover, flock centering is also important as it acts against the splitting of the system. However, being the less important constraint, it won't prevent the flock to split around obstacles when necessary.

Nevertheless, the approach of Reynolds was really specific to the animal world and was putting efforts into reproducing the dynamical ability of animals and the limitations their narrow vision.

While multi-agent systems is a really wide topic of research spanning over different areas such as sensor arrays or communication protocols, we will focus on systems of moving agents such as the former birds appearing in computer animations or robots evolving in one, two or three dimensions, such as, respectively, convoy of cars on a highway, mobile ground robots, or aerial vehicles such as planes and helicopters.

In the following, we will equally use the terms swarms, flocks and formations as synonyms of multi-agent systems. However, as explained in the survey [Gazi and Fidan, 2007], swarms and flocks are usually used by most authors to describe a system that is sparsely structured with a high number of agents and uncertain trajectories for each agent. Conversely, the term formation is mostly used for well-structured systems with a relatively small number of agents having rather well determined and precise trajectories.

One can distinguish among the planning algorithm, two different classes. On the first hand, one can consider only path planning. That is the creation of a geometric function along which an agent will evolve, the time dependency of this evolution being considered as another problem. On the other hand, one can directly consider the problem of motion planning. That is the creation of a time-constrained trajectory. Trajectory planning and motion planning are synonyms but should not be confused with path planning which is not time constrained. In this work, we will mostly focus on the problem of coordinated trajectory – or motion – planning for formations. In this sense, we will consider the problem of giving a time determined path to each agent of a multi-agent system while trying to follow the three rules of Reynolds.

For the general case of multi-agent systems, [Beard et al., 2001] classified the existing methods into three different categories:

Leader-follower: in leader-follower methods, each agent follows a global leader, for example an officer in a platoon. In more evolved implementations, each agent may follow another agent, for example the previous agent which can be considered as a local leader, or another mathematical point, for example the barycenter of the formation. Such methods make it easy to move the formation in a specific direction assigning a specific trajectory tracking problem to the leader. However, basic implementations of this methods are not resilient in case of failure of the leader, provide no feedback among the system, and require every agent to have knowledge of the leader's state. The last condition can in fact be a real

burden to the communication network. However, the simplicity of this method gives it a great advantage.

Behavioral methods: in behavioral based methods, all the agents are considered equal and they adopt behaviors built on informations coming from their only neighborhood. The behavior of an agent is usually based on simple rules. Thanks to the feedback shared between neighboring agents, these methods are following a decentralized approach making it easily scalable. However, it is usually difficult to predict the group behavior, and the stability of the formation is generally not easy to prove either.

Virtual structure: in virtual structure based methods, the agents are treated as elements of a single structure. It is then easy to have a group behavior and to enforce the agents to comply to a precise formation, but as a consequence the formation is constrained to a specific structure which is not easy to change.

However, as we will see in the following survey, even if this classification is globally worthwhile, crossovers between the different methods have been introduced. Moreover, some methods cannot be totally related to one of these methods.

The chapter is split in two sections. The first section reviews some of the various methods that have been used to design the evolution of multi-agent systems. It treats first the case of behavioral methods, which relies only on local informations while being based on simple rules. We review then methods relying on the *Particle Swarm Optimization* algorithm where several robots cooperate to find a "best point". The third subsection describes potential based methods, a class of methods based on virtual forces and their potential fields. A final subsection presents some method that make use of *Partial Differential Equations* to describe the evolution of swarms.

The second section reviews three classes of problems that are related to the present thesis. In a first subsection we report on the deployment problem, that is the planned evolution of a formation between two precise sets of points. In the following subsection, we review solutions to the problem of cooperative transportation of a payload by flying agents. In a last subsection, we describe some algorithms ensuring collision avoidance between agents moving on predetermined concurring paths.

This partition is meant to give an insight in the various methods used in the literature and to give a background to our research while showing that similar problems can be solved by means of various methods stemming from the different subsections of the methods' section.

1.1 Different methods of managing a multi-agent system

In this first section, we try to give an overview of some methods used to steer multi-agent systems. Accross these sections, one may encounter methods that allow low-level control laws for each agent of the system. One will also run into rather abstract methods handling high-level coordination problems while deferring the control laws to be used to low-level controllers specific to each agent – which may or may not be given by the respective authors.

However, as introduced earlier, the partition is not made on wether or not the suggested methods give low-level control laws. This section rather gives a review of methods which were

classified as follows: behavioral methods, Particle Swarm Optimization techniques, potential-based methods, graph theory induced methods and partial differential equation descriptions of swarms.

1.1.1 Behavioral methods

Behavioral methods are local motion-planning algorithms based on simple rules, which they apply to single particles based on informations coming from the only neighborhood. These methods are among the first to have been used in motion planning for multi-agent systems as they are easily stated and generally efficiently scalable since their rules are supposed to be implemented independently for each agent.

As a first example, [Vicsek et al., 1995] uses a set of agents evolving in a plane where each agent is represented by its respective position $\mathbf{x}_i$. The speeds of the agents $\mathbf{v}_i$ have a constant – uniform among the swarm – absolute value v and orientation θ_i. At initial time the positions and orientations are randomly distributed. The time is discretized, with discrete increment Δt, and the position of each agent is then updated at each time step t_j as

$$\mathbf{x}_i(t_{j+1}) = \mathbf{x}_i(t_j) + \mathbf{v}_i(t_j)\Delta t, \tag{1.1}$$

and the steering angle of the speed vector of the i-agent is updated as

$$\theta_i(t_{j+1}) = \langle \theta_i(t_j) \rangle_r + \Delta\theta, \tag{1.2}$$

where $\langle \cdot \rangle_r$ is the average over a circle of radius r, including the i-th particle. $\Delta\theta$ is a uniform noise with $\Delta\theta < \eta/2$. The aim of these two rules is to obtain a group behavior conforming to Reynolds' second alignment rule. Indeed, there is no rule to handle collisions between agents.

A simple analysis of the limit cases shows that at low speed – $v \to 0$ – the particles do not move and as a consequence the system does not evolve. At high speed – $v \to \infty$ – the system is completely mixed between two updates. Based on numerical simulations, the authors show that, setting a selected value of v, at low density and high noise, agents gather in small groups evolving in random directions. At high density and low noise, swarming, *i.e.* alignment of the agents in a unique group, appears. The authors explain this behavior with an analogy to the phase transitions of certain physical systems. As we will see in the following, using analogies to other domain of physics is a commonly used method.

Vicsek's rules have been improved by several authors. For example, based on the previous work, [Jadbabaie et al., 2003] introduces a graph based matrix representation of equation (1.2). Instead of updating the steering angle based on the geometric neighborhood, the authors propose to update it based on the network neighborhood of each agent. The update of the steering angle vector is then given as

$$\theta(t_{i+1}) = F_{\sigma(t_i)}\theta(t_i), \tag{1.3}$$

where θ is the vector containing the steering angles of all the agents, $F_p = (I + D_p)^{-1}(I + A_p)$, with D_p the diagonal matrix whose j-th diagonal element is the valence of vertex j within the graph p and A_p is the adjacency matrix of graph p. The application σ is chosen in a suitably

defined subset of the set of simple graphs on n vertices. Selecting an appropriate switching signal σ, not necessarily based on the geometric position of the agents, the authors prove the convergence of the headings of the agents

$$\lim_{i\to\infty} \theta(t_i) = \theta_{ss}\mathbf{1}, \tag{1.4}$$

where $\mathbf{1}$ is the vector with all coordinates equal to 1 and θ_{ss} depends only on $\theta(0)$ and σ. The authors thus provide a general proof of the convergence of the headings observed by Vicsek et al.. Based on this work, Jadbabaie et al. propose a feedback controller $u(t)$

$$u(t) = -G^{-1}_{\sigma(t)} L_{\sigma(t)} \theta(t), \tag{1.5}$$

with L_p the Laplacian matrix of the graph and G_p a suitably defined, nonsingular, diagonal matrix. $L_{\sigma(t)}\theta(t) = e(t)$ can be seen as the average heading error. By suitably choosing, the matrix G, it is then possible to steer the system

$$\theta(t_{i+1}) = \theta(t_i) + u(t), \tag{1.6}$$

to any desired angle. Using this open-loop controller and the previous results, a leader-follower model is then introduced. A (virtual) leader moves at constant speed with constant orientation. The authors then show that under certain assumptions all the agents will align with the leader. The authors finally prove this result to be valid also in the continuous case for the continuous-time model $\dot{\theta} = u$. However, as in the preceding work of Vicsek et al., the collision avoidance between agents is not addressed.

Nonetheless, the rules may remain simple and there is no need for graph theory. For example [Balch and Arkin, 1998] introduces a behavior based coordination scheme based on four different vector forces, *move-to-goal*, *avoid-static-obstacles*, *avoid-robots* that are based on the previous force and *maintain-formation*. The area of influence of each force is partitioned in three zones, the *ballistic zone* above a certain distance, the *dead zone* below a certain distance and the *controlled zone* in between. In the ballistic zone and the dead zone, the forces, both direction and intensity are constant; they are linearly increasing – or decreasing – in the controlled zone.

This process is applied to four military automated vehicles travelling. The vehicles move in four different formations originating from US military engagement rules: diamond-like, wedge-like, line-like or column-like formations. The different vehicles are supposed to follow either the platoon leader or the center of the formation. The authors show in simulations that the formation accurately manages obstacles and stays in formation. However, the algorithm is based on shared knowledge and regular exchange of GPS data, thus the algorithm experiences a high latency due to the technological requirements at the time. The algorithm is therefore not adapted to handle high speed of the vehicles. Moreover, due to the numerous factors – e.g. gains, radii of the different spheres of influence – the authors couldn't provide a thorough study of the algorithm stability properties.

The approach adopted in [Turpin et al., 2012] differs in many aspects from the previous works but can also be considered as a behavior-based method. The authors consider a tight formation of quadrotors which is described for each pair of agents (i, j) by a shape vector $\mathbf{s}_{i,j}$. Each agent assigns a confidence coefficient $c_{i,j}$ to the relative estimate of the state of the j-agent. These

weights are set such that $\sum_{j \neq i} c_{i,j} = 1$. The matrix of the confidence indexes, the confidence matrix C, is thus by construction right stochastic. The authors design the shape vectors over t so that collisions or interferences (for examples the downwash effect, a perturbation created by the air stream from a propeller on another propeller) are avoided. In such conditions, the desired position of each agent can be described by $\mathbf{x}_i^d = \sum_{j \neq i} c_{i,j}(\mathbf{x}_j + \mathbf{s}_{i,j})$ where $\mathbf{x}_j$ is the state (position and pitch) of the j-agent. The error in the position can then be estimated as $\mathbf{e}_i^d = \sum_{j \neq i} c_{i,j}(\mathbf{x}_i - \mathbf{x}_j - \mathbf{s}_{i,j})$.

One of the agents – potentially a virtual one – is defined as a leader and its trajectory is computed based on the low-level method introduced in [Mellinger and Kumar, 2011]. Once the trajectory is computed, the coefficients defining the leader's trajectory are sent to all the other agents, they can then compute their respective trajectories. This computation occurs inside an optimization process and at each step of this process the coefficients describing their trajectories are exchanged between the agents. Under certain assumptions given by the authors, this process is shown to converge to a set of optimal trajectories adopted by the agents.

The authors prove this procedure to be relatively efficient and the experiments involving a team of four quadrotors are relatively impressive as the formation follows a relatively aggressive trajectory. The formation even remains stable in the case of the failure of one of its members. However, the creation of the shape vector $\mathbf{s}_{i,j}$ is not particularly studied. Indeed, in the experiments, a constant square formation is used and no obstacle avoidance is introduced.

1.1.2 Methods based on Particle Swarm Optimization

Particle Swarm Optimization (**PSO**) was introduced by [Eberhart and Kennedy, 1995; Shi and Eberhart, 1998], who cited Reynolds flocking as an inspiration. It mimics the evolution of a swarm of particles in an arbitrary number of dimensions to optimize nonlinear functions. In this context, it can be seen as an application of a simple behavioral model to an abstract multi-agent system. This optimization method is particularly appreciated as it makes no assumptions on the function being optimized. Since the algorithm makes no use of the gradient of the function, it does not need the function to be differentiable and can handle noisy data. However it is not ensured that the algorithm will find an optimal solution – if there is one. The method is based on an evaluation function f, on the knowledge by each agent of its own best position over time with respect to the evaluation function and on the shared knowledge of the globally best position.

A simple version of the PSO algorithm – for which a pseudocode version is given in algorithm 1.1 – can be described as follows:

Initialization: A set of particles is created with – uniformly distributed over $[\mathbf{x}_{min}, \mathbf{x}_{max}]$ – random positions over the solution space. The speeds of the particles are also initialized – with uniformly distributed over $[-|\mathbf{x}_{max} - \mathbf{x}_{min}|, |\mathbf{x}_{max} - \mathbf{x}_{min}|]$ – random speeds. The best particle of each particle is initialized to its initial position while the global best particle is searched among them.

Evolution: At each step of the evolution, the speed of each particle is updated based on its current speed and position and on the knowledge of the best local and global particles. The particles are then moved according to the new speed and evaluated.

```
Data:
ω, φ_p, φ_g                                          ▷ Gains
n                                                    ▷ Number of particles
function PSO
    for i ← 0, n − 1 do
        x_i ← rand(x_min, x_max)                     ▷ Initialization of the positions
        p_i ← x_i                                    ▷ p_i is the local best
        if f(p_i) < f(g) then                        ▷ f is the evaluation function
            g ← p_i                                  ▷ g is the global best
        v_i ← rand(−|x_max − x_min|, |x_max − x_min|)  ▷ Initialization of the speeds
    while Termination criterion do                   ▷ Either number of steps or quality of the result
        for i ← 0, n − 1 do
            r_p = rand(0, 1)                         ▷ Random local gain
            r_g = rand(0, 1)                         ▷ Random global gain
            v_i ← ωv_i + φ_p r_p(p_i − x_i) + φ_g r_g(g − x_i)  ▷ Update of the speed
            x_i ← x_i + v_i                          ▷ Update of the position
            if f(x_i) < f(p_i) then
                p_i ← x_i                            ▷ Update of the local best
                if f(p_i) < f(g) then
                    g ← p_i                          ▷ Update of the global best
    return g
```

Algorithm 1.1 – Particle Swarm Optimization

It is believed that the swarm will converge to the global optimum if it exists or, if not, converge to a local optimum. However this convergence depends on the various parameters that can be chosen and the swarm can, in some cases, diverge or oscillate. The choice of these parameters has thus to be performed with great care whereas with a certain freedom.

We present Particle Swarm Optimization thoroughly because this algorithm will be used in the present thesis – however in a different way, at a rather higher level – to optimize sets of parameters in our motion planning algorithm. Nonetheless, some concrete low-level applications to trajectory planning problems have been conducted, making the PSO a currently used motion planner for multi-agent systems.

For instance, in [Pugh and Martinoli, 2007], the authors use wheeled robots to find an object using PSO. The particles of the algorithm are matched one-to-one with the robots. The latter detect the intensity of a signal emitted by the searched object. This intensity is used as the evaluation function of the PSO, the evaluation function is thus rather sparse and the gradient of the intensity would require more data to be evaluated. In comparison to basic PSO, the authors adapt the algorithm to take into account the limited speed of the robots, their limited acceleration and collision avoidance. In comparison to abstract PSO, the travel times are not homogeneous and, at each step, the robots have to stop to synchronize. In a first version of the presented algorithm , the robots have access to their own absolute position and have a global data connexion allowing all the robots to know the exact position of the local and global best position. In another more realistic version, the authors propose to allow the robots a short

memory of the past bests, with only relative positioning and information sharing with the only neighbors. Even in this simplified version, the search algorithms give good results and the robots converge to the origin of the signal.

The authors of [Couceiro et al., 2011] built on the work of [Pugh and Martinoli, 2007]. They improved the collision avoidance by adding an evaluation function for collision risks, based e.g. on the output of distance sensors, for example infrared or ultrasound sensors. A term based on the optimisation of this sensing function is added when updating the speed. The authors also make use of the Darwinian PSO [Tillett et al., 2005], a modified version of PSO which was shown to be less likely to be trapped in local extrema. This method pretends indeed to mimic the Darwinian behavior of social groups. The swarm is split in several sub-swarms that represent the social groups. Over the time, these sub-swarms reject the worst agents of their swarm and integrate the best lone robots. Doing this, the sub-swarms may escape locale extrema thanks to the knowledge of the integrated lone agents.

Such bio-inspired multi-agent algorithms are however diverse, as proves the work by [Turduev et al., 2014] in which a decentralized and asynchronous version of PSO as well as other biologically inspired optimization algorithm such as Bacterial Foraging Optimization and Ant Colony Optimization are used to find a gas leak using mobile ground robots equipped with gas sensors.

1.1.3 Potential methods

Back in 1992, [Rimon and Koditschek, 1992] tried to solve the problem of trajectory planning and tracking for a single robot using a class of potential functions. The potentials were designed to be high on obstacles and low on the chosen goals. The authors then suggested a control law that leads the agent to follow down to the goal the decrease of the potential. This solution was called the Navigation Function Method (NFM). However, this first step had several flaws. Since the suggested potential had infinite span, the controller needed to know the position of all the obstacles, thus requiring at every moment a global knowledge. Moreover, the agent runs the risk to be trapped in one of the local minima, the solution to this risk being to find accurate gains for the potentials. This solution is a computationally intensive task. Furthermore this work was not planned for multi-agent systems and further work was needed to extend this method.

Building on [Rimon and Koditschek, 1992], [Leonard and Fiorelli, 2001] proposed a motion planner for multi-agent systems. In addition to the real agents of the system, some virtual agents were introduced. Collision avoidance between real agents and formation constraints were obtained through artificial potentials which originated from both the real and the virtual agents. In a decentralized manner, a global potential field is evaluated for each real agent. The real agents are supposed to be totally actuated to follow the gradient of the global potential field and, in this way, to converge to local minima, which are created by the virtual agents to be the accurate positions in the formation. The potentials are proposed as

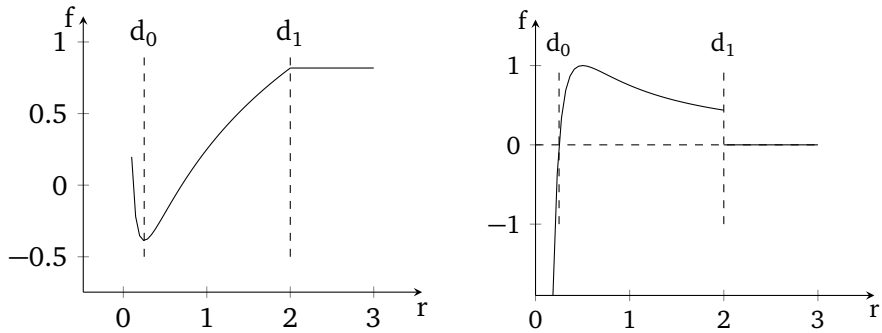
$$V = \begin{cases} \alpha\left(\ln(r) + \frac{d_0}{r}\right) & \text{if } 0 < r \leqslant d_1, \\ \alpha\left(\ln(d_1) + \frac{d_0}{d_1}\right) & \text{if } r \geqslant d_1, \end{cases} \tag{1.7}$$

as plotted, in one dimension, in figure 1.1a. The distance d_0 is the optimal distance between

two agents in the formation and d_1 is the influence threshold above which two agents do not influence each other. The forces, originating from these potentials, are of the form

$$f = \begin{cases} \nabla V & \text{if } 0 < r \leqslant d_1 \\ 0 & \text{if } r \geqslant d_1 \end{cases} \tag{1.8}$$

for which a one dimensional example (thus with $f = \frac{1}{r} - \frac{d_0}{r^2}$) is plotted in figure 1.1b. The



(a) Potential example. (b) Force example.

Figure 1.1 – Example of potential force with $d_0 = 0.25$ and $d_1 = 2$.

important point in figure 1.1a is the potential well found at $r = d_0$. Another important point seen in figure 1.1b is the steep gradient of the force around $r = d_0$ and the infinite repulsive force for collisions (which occur when $r = 0$). Errors in distance have thus an important contribution around $r = d_0$. They tend to constrain the agents to stay at the right distance each from another. Furthermore, above $r = d_1$ the agents have no influence on each other. This threshold makes the resolution easier. Indeed, the potential field around an agent is the sum of the only agents being close enough. Hence, the agents do not need the global knowledge of the positions of the other agents but only of a subset of them.

The control for each real agent is chosen to be

$$\mathbf{u}_i = -\sum_{\substack{j=1 \\ j\neq i}}^{n} f_I(r_{ij}) \frac{\mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|} - \sum_{k=0}^{m-1} f_h(h_{ik}) \frac{\mathbf{h}_{ik}}{\|\mathbf{h}_{ik}\|} + \mathbf{f}_i, \tag{1.9}$$

where $\mathbf{r}_{ij}$ is the vector from agent i to the real agent j and $\mathbf{h}_{ik}$ is the vector from agent i to the virtual agent k. f_I is the force between real agents. f_h is the force between real and virtual agents. $\mathbf{f}_i$ is a dissipative force used to ensure stability of the system.

A Lyapunov function for this system is

$$\Phi = \frac{1}{2} \sum_{i=1}^{n} \left(\dot{\mathbf{r}}_i \cdot \dot{\mathbf{r}}_i + \sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{f_I(r_{ij})}{r_{ij}} \mathbf{r}_{ij} + 2 \sum_{k=0}^{m-1} \frac{f_h(h_{ik})}{h_{ik}} \mathbf{h}_{ik} + \mathbf{f}_i \right), \tag{1.10}$$

where $\mathbf{r}_i$ is the position of the agent i. Using the control law suggested in equation (1.9), the

derivative of the Lyapunov function is:

$$\dot{\Phi} = \sum_{i=1}^{n} \dot{\mathbf{r}}_i \cdot \mathbf{f}_i. \tag{1.11}$$

Based on this derivative, the authors propose to chose the force as $\mathbf{f}_i = -a_i \dot{\mathbf{r}}_i, a_i > 0$ to make the system stable. However, even if a few simple examples with one, two or three agents are shown and rules are given to build some simple geometric formations with more agents, several problems arise. For example, generating trajectories for the virtual leaders to avoid collisions with static obstacles, splitting or grouping the system, changing the shape of the formation are left as open problems. The uniqueness of the formation for a given virtual agent distribution is also not ensured. Moreover, the suggested force $\mathbf{f}_i$ is based on the evaluation of the speed of the agent which may be difficult to obtain for example in the case of UAVs.

[Chang et al., 2003] extend the NFM framework of [Rimon and Koditschek, 1992] to the case of multi-agent systems. However, to avoid collision with obstacles, they replace the repelling potentials introduced in the NFM framework by gyroscopic forces. These forces, acting perpendicularly to the direction of motion, solve the problem that appears when agents move exactly toward the obstacles. Indeed, in this case the agents were unable avoid the obstacle, they were simply slowed down and repelled. The authors however add a braking force in case of eventual collision to complete the effect of the rotation force. These two forces were chosen because they don't affect the potential field used to find the goal (in this case, a virtual leader) and are closer, in comparison to repulsive forces, to the capabilities of usual ground vehicles. Furthermore, these forces were set to act only at close distances thus decentralizing the problem and making it scalable.

[Olfati-Saber, 2006] merges previous works on flocking – presented in [Olfati-Saber and Murray, 2003] – and on consensus in networks of agents – presented in [Olfati-Saber and Murray, 2004]. Consensus in a network of agents is the ability to converge to a common state, here a common goal speed, with a sparse communication topology. The author introduces three scalable flocking algorithms, the first two for agents evolving in free space while the last one introduces obstacle avoidance. The author shows that these three algorithms respect the three rules of Reynolds. The algorithms are based on three different types of agents, called α, β and γ-agents.

The first algorithm is a gradient-based algorithm with velocity consensus protocol for regular α-agents. This algorithm, which forms a basis for the following two, creates so-called α-lattices – a regular structure similar to structures – in the flock, and is shown to respect the three rules of Reynolds. The interactions between α-agents are short ranged so that the algorithm remains decentralized. The following algorithms, building on this "Reynolds compatible" algorithm, are also compatible with the rules of Reynolds. The second algorithm adds γ-agents which states are known from all the α-agents and play the role of virtual leaders, thus assigning objectives to the α-agents. One should note that in a flock the movement of which is based on the second algorithm following a γ-agent, the speed of the flock does not necessarily equal the speed of the virtual leader. The third algorithm introduces obstacle avoidance via the addition of β-agents. These agents are repulsive agents evolving on the boundaries of obstacles (which are supposed to be smooth and convex)

The suggested control is based on the contributions of each type of agents and is decomposed

as follows

$$V(q) = c^{\alpha}V_{\alpha}(q) + c^{\beta}V_{\beta}(q) + c^{\gamma}V_{\gamma}(q), \tag{1.12}$$

where q represents the configuration of the flock. V_{α} is the potential of the difference between the perfect lattice and the current configuration, where local minima are perfect lattices. V_{β} is induced by the obstacle avoidance β-agents and V_{γ} comes from the goal defining γ-agents.

Simulations are run in two dimensions with a hundred and fifty agents for the three algorithms. The split and rejoin maneuver is shown to work correctly for such a flock. Further runs are shown in three dimensions with fifty agents. They show the pertinence of the suggested framework. However, this framework is constrained by the lattice and by the fact that the knowledge of the position of the γ-agents, the trajectories of which have to be independently planned, have to be shared among the whole flock.

The condition on the γ-agents can be however lessened. For example, [Su et al., 2009] propose an algorithm that does not need all the α-agents to know of the γ-agents, which are not constrained to a constant speed as in previous works. The proposed algorithm will indeed asymptotically track γ-agent speed.

Whereas the previous works are based on an idea originating from the field of potential methods, which were originally developed for single-agent systems, they extensively use graph theory. Methods based on graph theory are hence studied in the following subsection.

1.1.4 Graph based methods

In multi-agent systems, graph theory appears in many cases, for instance in the case of peer-to-peer communication or maintaining formations, as a natural study tool.

[Desai et al., 2001] represent the control topology of a group of unicycles as a directed acyclic graph. In this graph, every robot is a vertex and communication links are edges. All but one have one or two incoming edges and are followers. The lonely agent is the leader. Based on whether a robot has one or two leaders, it will be controlled respectively by a $l-\psi$ controller – maintaining a defined distance and angular offset to its preceding leader – or a $l-l$ controller – maintaining a defined distances to its two preceding leaders. Meanwhile, the global leader follows an independently a defined trajectory. The authors evaluate the number of possible formations and suggest a framework to change the topology of the graph and thus the shape of the formation.

The graph of a formation can also represent the ability of the agents to sense each other. [Fax and Murray, 2002] study a system of N agents where no absolute measurements are available but where each agent can sense at least one other agent. The latter agent is not supposed to sense the former thus making the graph not necessarily undirected. The authors identify the role the Laplacian matrix of the graph plays in the analysis of the stability of the formation. The Laplacian matrix has at least $\lambda = 0$ as an eigenvalue which can be seen to correspond to the lack of knowledge of the agents' absolute positions. The authors give a Nyquist criterion on the stability of the formation according to the eigenvalue of the Laplacian matrix. Therefore, the dispersion of the eigenvalues is equivalent to the information sharing rate. In complete graphs – all the possible arcs exist and the information sharing is maximum – all, but one, eigenvalues are at a single position while for a single directed cycle – the formation forms a circle and each agent

senses only one other agent; the information sharing is minimal – the eigenvalues are maximally dispersed on the border of the circle $1 - e^{j(i-1)/2\pi}, i \in [1, N]$ making it difficult to stabilize due to the Nyquist criterion. Indeed, in the first case, the noise due to one of the agent is averaged by the formation while in the second case, the noise is hardly averaged and propagates becoming a periodic perturbation. The authors thus draw the conclusion that aperiodicity is a desirable property of formation interconnection topologies.

One can also use graphs to define the shape of a formation and to determine controls based on this shape. For example [Olfati-Saber and Murray, 2002] build on the potential functions introduced in [Leonard and Fiorelli, 2001] and use a graph theoretical approach. Indeed they consider the formation as a weighted undirected graph, where the weights represent the distance that neighboring agents in the graph should maintain between them, and derive potential functions from this graph. Thereby, they can show that if the graph representing the formation is *rigid* and *unfoldable* the formation is unambiguous – unlike the formations proposed in [Leonard and Fiorelli, 2001] where the agents could possibly have converged to any of the formations that were left possible by the potential functions.

For a pair of neighboring vertices $\varepsilon = (v_i, v_j)$ of the graph $\mathcal{E}$ with weight d_{ij} and position q_i and q_j, the authors introduce the function $\varphi_\varepsilon(q_i, q_j) = \|q_i - q_j\| - d_{ij}$ and the vector function $\Phi(q) = \{\varphi_\varepsilon | \varepsilon \in \mathcal{E}\}$. This function Φ is called the *structural constraint function*. A configuration $\bar{q}$ of the formation is an equilibrium of the formation if and only if $\Phi(\bar{q}) = 0$. It is shown – based on Lyapunov analysis – that the control $u = \nabla\langle\Phi(q), \Phi(q)\rangle - D(q, \dot{q})$ where $D(q, \dot{q})$ is some damping force verifying certain assumptions, guarantees collision-free local asymptotic stabilization of the vehicles formation to a intended undirected and unambiguous formation graph.

The idea of *structural constraint functions* appeared also, under the name of *formation constraint functions* in [Egerstedt and Hu, 2001]. The authors used these functions, defined in a similar way to [Olfati-Saber and Murray, 2002] to compute the desired path of each agent based on a single virtual leader path. The tracking of the desired paths is then left to the controllers of each agent. The authors propose to add obstacle avoidance to the agents by changing the low-level controller rather than changing the high-level path. The cooperation of these two levels of control is illustrated with the simulation of a formation of three unicycles in an equilateral formation avoiding a circular obstacle.

The case of graph topologies varying with the state of the system has been studied in [Mesbahi, 2005] who introduces a concept of controllability for state-dependent graphs. The author proves links between the controllability of the formation graph and the controllability of the underlying multi-agent system exist. In another work, [Olfati-Saber and Murray, 2004] study the problem of consensus among the agents of a formation, where the communication network between the agents can have either fixed or switching topology, while potentially experiencing time-delays.

1.1.5 Methods based on Partial Differential Equations

Considering the formalism introduced in some of the previous works, using representations originating from the theory of Partial Differential Equations (**PDE**) appears to be a convenient tool to study large populations. Indeed, instead of considering each agent and its influence on its neighbors separately, considering a swarm – which has, in our terminology, a high agent density – as a continuous medium seems to be relevant. Thus, instead of considering a finite set of finite

difference equations describing the motion of each agent, a single partial differential equation of infinite order appears to bring some simplifications. However, this approach contains in itself several difficulties, due notably to the difficulty of studying general PDEs. Some of the points that will have to be studied with a particular interest are, for example, the evolution of the boundaries of the swarm, the evolution of the population of the swarm or of its density and the evolution over long time periods of the swarm.

As a first example, [Mogilner and Edelstein-Keshet, 1999] decide to represent the evolution of a continuous swarm as the result to a diffusion-advection integro-differential equation. The equation used is

$$\partial_t \rho = \partial_x \left(D \, \partial_x \rho - v\rho \right), \tag{1.13}$$

where x is a one-dimensional space coordinate, $\rho(x,t)$ the density of agents in the swarm at point x and time t, D a density independent diffusion coefficient and v a density dependent velocity. The first term in the right-hand side of the equation is thus the effect of diffusion while the second term represents the effect of advection. Convection is the sum of diffusion and advection and is responsible for attraction, repulsion and macroscopic motion of the swarm. The velocity is taken as a convolution in the form

$$v(\rho) = K * \rho = \int_\Omega K(x - x')\rho(x', t)dx', \tag{1.14}$$

where the normalized kernel K has a compact support over Ω and represents the effects of the surrounding of a point on the velocity of this specific part of the swarm. The goal of this work is to study the evolution of a square-shaped density distribution of the swarm where the density has to remain constant at ρ_0 inside the swarm. When K is an even kernel with a support sufficiently small in comparison to the size of the swarm – velocities are influenced by a close neighborhood in comparison to the swarm – and $v = A_e K * \rho$, the velocity can be approximated to be

$$v(\rho) = \begin{cases} \frac{1}{2} A_e \rho_0 & \text{at the back edge of the swarm,} \\ A_e \rho_0 & \text{inside the swarm, sufficiently far from the edges,} \\ \frac{1}{2} A_e \rho_0 & \text{at the front edge of the swarm.} \end{cases} \tag{1.15}$$

Thus, an even kernel initiates a drift proportional to the average density in the direction of the sign of A_e. The boundaries travel at a lower speed, meaning that the back of the swarm dislocates and the front of the swarm is taken over by the inside. In the case of an odd kernel with "small" support and $v = A_o K * f$, the velocity can be approximated to be

$$v(f) = \begin{cases} \frac{1}{2} A_o \rho_0 & \text{at the back edge of the swarm,} \\ 0 & \text{inside the swarm, sufficiently far from the edges,} \\ -\frac{1}{2} A_o \rho_0 & \text{at the front edge of the swarm.} \end{cases} \tag{1.16}$$

Thus, an odd kernel leads the edges of the swarm to either converge to the center of the swarm if $A_0 > 0$ and to diverge if $A_0 < 0$. These two observations lead the author to propose the velocity of the swarm to be

$$v(\rho) = a_e \rho + \left(A_a - A_r \rho \right) (K_o * \rho), \tag{1.17}$$

Where $a_e\rho$ is the drift term (the kernel is taken to be a Dirac distribution), A_a an attractive term and $A_r f$ a repulsive term. The kernel is odd and is taken to be

$$K_o(x) = \begin{cases} \frac{1}{2r} & -r \leqslant x < 0, \\ -\frac{1}{2r} & 0 \leqslant x \leqslant r, \\ 0 & |x| > r, \end{cases} \tag{1.18}$$

where r is the maximum interaction distance. This term determines how far the agents influence each other.

It is shown that with such a representation, the swarm is not stable as there will be agents either at the rear or at the front of the swarm leaving it. This instability is due to the infinite range of the diffusion, which is however necessary. Furthermore, the swarm breaks up when moving too fast, *i.e.* if the advection becomes stronger than the aggregation – $a_e > \frac{1}{4}A_a$ – or if the diffusion is too high. However the swarm can still be locally stable for appropriate values of the diffusion coefficient D. The authors however show that there exist band-like solutions which can be stable for possibly long times.

[Topaz and Bertozzi, 2004] found their study on the kinematic of groups of bacteria-like agents on the work of [Mogilner and Edelstein-Keshet, 1999]. They study the partial integro-differential model introduced therein in the case of a bi-dimensional swarm. The population of the swarm is supposed to be constant as births and deaths are supposed to occur in a negligible quantity on the time scale of the swarming dynamics. As the movement of the swarm is supposed to originate only from social interactions, which are supposed to be linearly decreasing with distance, between pairs of agents, the velocity $\vec{v}$ of the swarm is taken to be a function of the density of the swarm $\rho(\vec{x})$.

Based on these assumptions, the authors assume the velocity and the density of the swarm to be solutions of

$$\rho_t + \nabla \cdot (\vec{v}\rho) = 0, \tag{1.19}$$

which is a two-dimensional version of equation (1.13) where D – the diffusion coefficient – is taken to be zero, thus cancelling the effects of diffusion. The velocity $\vec{v}$ is the result of the convolution

$$\vec{v} = \vec{K} * \rho = \int_{\Omega \subset \mathbb{R}^2} \vec{K}(|\vec{x} - \vec{y}|)\rho(\vec{y})d\vec{y}, \tag{1.20}$$

where $\vec{K}$ is a spatially-decaying and isotropic kernel, which embodies the various assumptions made by the authors on the social interactions of the swarm. The authors use Hodge's decomposition theorem to decompose the velocity and thus the kernel into

$$\vec{K} = \nabla^{\perp} N + \nabla P, \tag{1.21}$$

where P models the interaction pressure which is the motion due to the inhomogeneity in density and their social consequences. N represents additional motions. The authors study both limit cases $\nabla P = 0$ and $\nabla^{\perp} N = 0$. The first case corresponds to incompressible motion. Injecting this assumption in equation (1.19), we get the equation

$$\rho_t + \nabla \cdot (\rho \nabla^{\perp} N * \rho) = 0. \tag{1.22}$$

The authors study the effect of N being a Gaussian interaction. In the two limit cases where N is a Dirac distribution – which is the identity of the convolution – and where N is constant, the left hand side of equation (1.19) simply reduces to ρ_t leaving the swarm invariant. Between these two limit cases, it is shown that rotation in the sens of the sign of N is induced to the swarm. Whatever the initial distribution is, numerical simulations show a tendency of the swarm to aggregate in a galaxy-like distribution rotating with spiral arms, acquiring thus a rotational symmetry.

In the case of potential motion – $\nabla^{\perp} N = 0$ – equation (1.19) reads

$$\rho_t + \nabla \cdot (\rho \nabla P * \rho) = 0. \tag{1.23}$$

The behavior of the solution to this equation depends on the sign of P. If P is negative, the limit case where P is constant leads to a steady state $\rho_t = 0$. In the opposite, when P is a negative Dirac distribution, equation (1.23) becomes Darcy's law for flow in porous media showing dispersion of the swarm. In-between, the swarm experiments both diffusion and convection leading the swarm to disperse. In the opposite case, where P is positive, a linear stability analysis shows aggregation of the swarm.

There are of course other ways to describe a flock by partial differential equations. [Toner and Tu, 1998; Toner et al., 2005] develop a hydrodynamical model for multi-agent system. The chosen model is in some way similar to Navier-Stokes equation for a simple compressible fluid. However the model contains more terms than Navier-Stokes equation and lacks the conservation of momentum which is a consequence of the authors' choice for the system not to be Galilean invariant. Unlike the previously cited works using PDEs, the authors, beside an isotropic model, also provides an anisotropic model based on the fact that, e.g., birds prefer flying forward rather than upward or laterally.

1.2 Different problems of collaborative systems

In the following section, we review how some of the previously detailed methods are used when confronted to some of the problems we want to address in this thesis. Three different, however related, classes of problems are reviewed. The first one is dedicated to the problems of deployment, *i.e.* how to drive a formation to precise positions. The second class of problems concerns the cooperative transportation of a load with the aim to transport a payload to a precise position while adding dynamical coupling to the agents. The last subsection reviews the problem of collision avoidance. Indeed, some of the methods used to solve the first two classes of problems don't bear collision avoidance in mind.

1.2.1 Deployment problems

In some cases, it is not sufficient to drive the global movement of a multi-agent system at a high level, as it was done in the various works presented in the previous section. Indeed, some

specific applications may require the various agents to be at the beginning in specific positions and orientations and to move to precise successive states.

Inspired by earth based astronomical interferometers such as the Very Large Telescope[1], the Very Large Array[2] or the Atacama Large Millimeter Array[3], there have been plans to build such a space based interferometer. For example, [Beard et al., 2001] suggested a coordination architecture for a spacecraft formation dedicated to this mission. The suggested approach based on the sharing of a dynamic coordination variable can be either centralized or decentralized. In the first case, a central satellite broadcasts coordination variables describing the wanted goal state. In the second case, every agent embeds an observer of the coordination variable. This work is applied to a formation of three spacecrafts that have to perform data acquisitions in different coordinated positions while respecting constraints on attitude, relative position and energy minimization.

[Sultan et al., 2007] consider a similar problem for a fleet of spacecrafts which are supposed to be point-mass double-integrators evolving in deep-space, where gravity can be neglected. The aim of the authors is to determine a set of energy optimal trajectories passing through a succession of way-points. These trajectories are shown to be cubic polynomials in time while rational in the way-points times and linear in the way-points locations:

Lemma 1.2.1. *Let* $\{(t_j, w_j, v_j), j \in \{1, \dots, M+2\}\}$ *be a sequence of way-points specifying time, position and spacecraft velocity, with* $t_j < t_{j+1}$. *Let* $r(t)$ *denote* C^1 *trajectories going through these waypoints. Then the unique trajectory that minimizes the energy of the spacecraft is given by:*

$$r(t) = \frac{1}{6}c_j(t^3 - t_j^3) + \frac{1}{2}d_j(t^2 - t_j^2) - \frac{1}{2}\left(c_j t_j^2 - 2d_j t_j - 2v_j\right)(t - t_j) + w_j, t_j \leqslant t \leqslant t_{j+1}, \tag{1.24}$$

for $j \in \{1, \dots, M+1\}$, *where*

$$c_j = \frac{-12(w_{j+1} - w_j) + 6(v_{j+1} + v_j)(t_{j+1} - t_j)}{(t_{j+1} - t_j)^3}, \tag{1.25}$$

$$d_j = \frac{v_{j+1} - v_j}{t_{j+1} - t_j} + \frac{t_{j+1} + t_j}{(t_{j+1} - t_j)^3}\left(6(w_{j+1} - w_j) - 3(v_{j+1} + v_j)(t_{j+1} - t_j)\right). \tag{1.26}$$

Collision avoidance between the spacecrafts is ensured ahead of the optimization process by considering forbidden spheres around the agents. The trajectories are then optimized to respect the collision constraints while minimizing the energy consumption. The obtained global trajectories are a set of C^1 piecewise polynomials in time. This framework is however particularly specific to completely actuated agents with double-integrator dynamics. Moreover, the agents are modelled as point-masses. Thus the suggested trajectories don't take into account the attitude of the spacecrafts. It might therefore be necessary to rotate the spacecrafts at the end in specific directions, and thus to consume additional energy.

Another deployment problem is addressed by [Dunbar and Murray, 2006] for a team of robots. Although the robots have decoupled dynamics, the robots' states are coupled by a cost function

[1] Operated by the European Southern Observatory, the VLT, located in the Atacama desert, Chile, combines eight telescopes of which four are moveable.

[2] Operated by the National Radio Astronomy Observatory, the VLA, located in New Mexico, USA, combines twenty-seven independent moveable antennae.

[3] The ALMA is an international partnership located in the Atacama desert, Chile, combines sixty-six movable radio telescopes.

which determines the wanted equilibrium state for the formation. However, while the cost function is centralized, the control problem is distributed: each agent solves, and optimizes, an independent Model Predictive Control (**MPC**) problem. For that purpose each robot has to make assumptions on its neighbors' trajectories at each time step of the MPC controller. While the suggested solution can be considered as decentralized since the robots work on independent assumptions without the need to communicate, the algorithm still needs the presence of a central server for synchronization purposes. Nonetheless, if the receding horizon of the MPC is sufficiently close – implying that the open-loop controllers do not deviate to much – the system will converge to the chosen positions.

Building on the MPC controller used by Dunbar and Murray, [Prodan et al., 2011] add inter-agent collision avoidance constraints. To do so they define around each agent i a convex safety region as a convex polytope S_i. The authors want the agents to come as close as possible to the origin. As a consequence, they suggest an optimization of the function

$$\min_{\mathbf{x}_i} \sum_{i=1}^{n} \|\mathbf{x}_i\|_2, \tag{1.27}$$

under the conditions

$$S_i \cap S_j = \emptyset, i \neq j. \tag{1.28}$$

However, the resulting optimization constraints are non-convex, making the problem difficult to solve. The authors suggest to address this problem using Mixed-Integer Programming techniques, which are known to be NP-hard – their complexity is increasing dramatically fast. Nevertheless, the authors succeed – in comparison with the previous literature in the field of optimization on non-convex problem – in formulating the problem with fewer variables, reducing as a result the computational time, but not the complexity, of the problem.

Assuming the agents are interchangeable, and using the result of the aforementioned optimization, the authors optimize the task assignment – *who goes where ?* – problem. Finally a model predictive problem is optimized using the previous collisions constraints, while solving the task assignment problem anew at each step of the MPC. This approach seems to be effective for collision avoidance. However its high – exponential – complexity seems to make it unbearable for systems with more than a few agents.

In comparison with the previous work, the environment in which the agents of [Turpin et al., 2013] move, has obstacles. However, all of them are known to the planner. As previously, the agents are identical and interchangeable. Once again, the planner is split into a high-level and a low-level layer. The high level planner computes all the possible trajectories in a graph of acceptable states – *i.e.* rejecting collisions – using Dijkstra's algorithm. Using the costs evaluated by this step, the task assignment problem is solved through the Hungarian Algorithm [Munkres, 1957; Kuhn, 1955]. The low-level distributed layer then elaborates trajectories adapted to the agents, in this case quadroters, using specific algorithms, here the minimal snap trajectories introduced in [Mellinger and Kumar, 2011].

The computational complexity of Dijkstra's algorithm being quasilinear in the size of the graph (linear in the number of edges, quasilinear in the number of vertices), the complexity of the high-level planner is mainly due to the Hungarian algorithm which is cubic in the number of agents. It might appear to be far better than the difficulty of, for example, the approach

of [Prodan et al., 2011] but it should be pointed out that the state graph is supposed to be precomputed, which is an acceptable hypothesis only if the test area is known beforehand.

To avoid this complication, the problem is addressed by [Panagou et al., 2014] not with a graph approach, but using Lyapunov functions. Instead of having the graph beforehand, the agents, which have only short-range communication ability, solve two Lyapunov problems successively. First, when establishing a communication network with m other neighboring agents, the agents share out among the members of the communication their current respective goals using the Hungarian algorithm. The cost of the affectation is, in this case, the sum of the distances between their current positions and their possible goals, which is a Lyapunov function. This switched system has been proved to be globally asymptotically stable. The collision avoidance and convergence toward the goals is then also ensured by a Lyapunov function.

In a recent paper, which greatly inspired the work presented in this thesis, [Meurer and Krstić, 2011] suggest a deployment solution based on Burgers' equation. Unlike the models presented in the subsection 1.1.5 – Methods based on Partial Differential Equations on pages 28 to 31 where the equations depended on $\rho(x,t)$, where ρ was the density of agents at the position x of space, the equation, now, depends on $x(\alpha,t)$ where x is a position in space and α is a continuous index of the agents. The previous models were particularly well adapted to describe a group of indistinguishable agents spanning over a defined volume of space. On the contrary, the model suggested by Meurer and Krstić is built to describe a curve-like formation of distinguishable agents.

Burgers' equation is a famous partial differential equation originating from fluid mechanics. It was chosen because, as a second order parabolic nonlinear equation, it offers a good trade-off between simplicity to stabilize and the span of reachable equilibria by the existence of shock-like equilibria, thanks to its nonlinear nature, allowing, for example, switchbacks and corner-like shapes in the formation.

The agents in the group are referred to by their continuous index $\alpha \in [0,1]$. The authors call the end agents $\alpha = 0$ and $\alpha = 1$ respectively the anchor and the leader. The authors then compute the trajectories of the real agents by finitely discretizing solutions obtained independently for each dimensions.

The authors use a custom version of Burgers' equation

$$x_t(\alpha,t) = a x_{\alpha\alpha}(\alpha,t) - b x(\alpha,t) x_\alpha(\alpha,t) + c(t) x(\alpha,t). \tag{1.29}$$

It differs from the genuine Burgers' equation by the addition of the term $c(t)x(\alpha,t)$. The equation has the following inhomogeneous Dirichlet boundary conditions

$$\begin{cases} x(0,t) = u_0(t) \\ x(1,t) = u_1(t) \end{cases}, \tag{1.30}$$

and the initial condition

$$x(\alpha,t_0) = x_0(\alpha). \tag{1.31}$$

As the authors wish for the system to be steady before the deployment – *i.e.* for $t \leqslant t_0$ – and after the deployment $t \geqslant t_1$, the function $c(t)$ should read for $t \geqslant t_1$

$$c(t)\bar{x}(\alpha) = b\bar{x}(\alpha)\bar{x}_\alpha(\alpha) - a\bar{x}_{\alpha\alpha}(\alpha), \tag{1.32}$$

where barred values are time stable functions of α with

$$\begin{cases} \bar{x}(0) = \bar{u}_0 \\ \bar{x}(1) = \bar{u}_1 \end{cases} . \tag{1.33}$$

Since $\bar{x}(\alpha)$ is not zero, taking the time derivative of equation (1.32) leads to the fact that $\partial_t^n c(t) = 0$ while $c(t)$ is not constant. The function c is consequently non-analytic at each steady state t_0 (start), t_1 (finish) or at any intermediate desired equilibrium.

The authors search for a solution to equation (1.29) in terms of a formal power series

$$x(\alpha, t) = \sum_{n \geqslant 0} \hat{x}_n(t) \frac{(\alpha - \hat{\alpha})^n}{n!}, \tag{1.34}$$

with $\hat{\alpha} \in (0, 1)$ fixed. Substituting this power series in equation (1.29) leads to the following differential recursion equation

$$x_n(t) = \frac{1}{a} \sum_{i=0}^{n-2} \binom{n-2}{i} \hat{x}_{n-2-i}(t) \hat{x}_{i+1}(t) - c(t) \hat{x}_{n-2}(t) + \partial_t \hat{x}_{n-2}(t), n \geqslant 2, \tag{1.35}$$

and

$$\begin{cases} \hat{x}_0(t) = y_0(t) \\ \hat{x}_1(t) = y_1(t) \end{cases} . \tag{1.36}$$

It is then shown that $x(\alpha, t)$, $u_0(t)$ and $u_1(t)$ can be written in terms of power series of the functions y_0 and y_1 and of their time derivatives. The functions y_0 and y_1 are called flat outputs of the system. This name of "flat outputs" stems from the theory of differentially flat systems to which an introduction will be given in section 6.3 – Introduction to flatness based control. The power series defining $x(\alpha, t)$, $u_0(t)$ and $u_1(t)$ are shown to be converging when y_0, y_1 and c belong to a special class containing both analytic and non-analytic functions, the Gevrey class. Under such circumstances, the authors determine the finite radius of convergence of the power series of equation (1.34).

Since they are flat outputs, functions y_0 and y_1 are suggested to satisfy all the previous conditions. In the opposite case, the power series are divergent, but the authors introduce a resummation scheme that still allows trajectory generation. The functions y_0, y_1 and c, which are of the form $C_1 + C_2\varphi(t)$ where C_1 and C_2 are constants and $\varphi(t)$ is a smooth function allowing the transitions between $\varphi(t \leqslant t_0) = 0$ and $\varphi(t \geqslant t_1) = 1$, are set to ensure the transition between the two steady states. Various formations can be attained by varying the available parameters in the suggested y_0, y_1 and c functions. The authors give numerical examples of deployments for a system of twenty-five agents. For example, trajectories are given changing from a line-like formation to a circle-like formation.

1.2.2 Cooperative transportation of a swinging load

The idea of using a UAV to transport a load, fixed to the UAV body or swinging at the end of a cable, has been around for a long time. Almost all the rescue helicopters are equipped with a hoist used, for instance, by coast guards to save castaways or disasters' victims. This task,

however, needs the pilots in manned helicopters to be experienced in order to maintain a stable hovering position. The task's complexity drastically increases if the pilot has to move the payload at the end of the cable or if the aircraft cannot hover – e.g. in the case of a plane [Murray, 1996]. The problem is indeed difficult due to the limited controllability of the payload.

Using multiple agents tied to the payload is a solution to ensure a better control of the charge. At the same time it is a simple way to increase the payload lift ability of a single agent. Nevertheless, linking the agents to a common single payload introduces a new type of constraints for the formation – in contrast to the literature we already reviewed – by dynamically coupling the agents of the system.

Before trying to transport a load with UAVs, the problem of cooperative transportation was first applied to ground robots. For instance [Ogren et al., 2001] use a virtual structure, enforced by a Lyapunov function approach, to maintain a formation of ground robots. It is used to let two robots carry a beam on a construction site. To do so, the robots have to maintain a strict distance between them. Unfortunately, the virtual leader approach used to displace the formation – and thus the beam – is not easily suited to control the orientation of the beam.

To transport a load in the air, more possibilities are available than when limited to displacements parallel to the ground as it offers a higher degree of freedom. It is however possible to constrain the aircrafts in a plane. [Mellinger et al., 2013] suggest, for example, a way to transport rigid payloads with predefined shapes – namely line-, cross-, "T"- and "L"- shaped – with quadrotor UAVs. Unlike usual aerial transportation techniques, the aircrafts are firmly bound to the payload. The UAVs thus remain parallel during the entire flight which adds constraints to the usual quadrotors controllers. To design the controller for the multiple agents, they are all assumed to be parallel and to know their respective positions. That way, after defining the trajectory of the payload, the authors can immediately compute the trajectories and the controllers of the UAVs. In a sense, this solution is similar, and indeed it takes part of its inspiration therein, to that of the Distributed Flight Array by [Oung et al., 2010] where several instable monorotor UAVs assemble rigidly together in random patterns to create a distributed stable aircraft.

However, linking the UAVs directly to the payload leads the transportation problem to be greatly over-constrained. In comparison, transporting a load linked to UAVs with cables gives more freedom in the set of reachable trajectories. However, path planning and control are more difficult when adding degrees of freedom with cables.

For example, [Maza et al., 2010] use three middle scale model helicopters – 12.5 kg each – to carry a 4 kg load – the maximum load for a single helicopter being 1.5 kg – linked to each helicopter by 13 m ropes. The helicopters are simply constrained to move on the same height in a triangular shape formation, 8 m apart. They have to translate the point-mass payload smoothly, the payload being at the center of the projection on the ground of the formation. The helicopters are controlled in a decentralized manner. Indeed, every helicopter is embedded with a direction sensor made out of magnetic encoders, and a force sensor attached to the rope. Thus the authors can get a precise estimate of the state of the load and implement their controller in a decentralized way thanks to this feedback.

Inspired by the example of cooperative fixed cranes imposing poses – position and orientation – to payloads thanks to variable rope lengths, [Michael et al., 2011] suggest to apply similar methods to the aerial transportation of a payload. Instead of varying the lengths of the cables,

the authors use the agility of the quadrotors to act on the pose of the payload. The authors focus on a system of three quadrotors and solve the two following problems: finding optimal positions for the quadrotors in order to obtain a chosen pose of the charge and finding the pose of the payload for specific quadrotor positions. The answer to the first problem is solved using the Moore-Penrose inverse. The solution is not unique and for a precise pose, one can choose for the quadrotors a solution that avoids collision while equalizing the tension in the cables. The solution to the second – direct – problem is found thanks to an optimization-based reformulation. Motion planning for the multi-agent system is solved using the computed set of quadrotors' positions suitable for a chosen payload pose. Unfortunately this approach is limited by the inability to damp out oscillations of the payload leading the authors to move the payload relatively slowly.

Another solution is adopted by [Sreenath and Kumar, 2013] by using the fact that – under certain assumptions – the system formed by several differentially flat agents carrying a payload suspended by cables is flat. The authors make the following assumptions:

- The cables are massless and do not stretch nor contract – it is partially equivalent to the assumption that the cables are rigid.
- The cables are attached at the quadrotors' center of mass, which is a common simplifying assumption.
- Air drag on the quadrotors and the load is negligible, which again is a common assumption at low speed.

Assuming that T_i is the tension in the i-th cable binding the i-th quadrotor to the payload and that $\mathbf{q}_i$ is the unit direction vector of the i-th cable, the system follows the kinematic relation

$$\mathbf{x}_i = \mathbf{x}_L - L_i \mathbf{q}_i. \tag{1.37}$$

where $\mathbf{x}_i$ is the position of the i-th quadrotor and $\mathbf{x}_L$ is the position of the load. With the notations of Part II – Modeling and control of a trirotor UAV, the system follows, for a point-mass load in the inertial frame, the following kinetic equations

$$\begin{cases} m_i \ddot{\mathbf{x}}_i = f_i R_{\mathcal{B}}^{\mathcal{I}} \mathbf{z}_{\mathcal{B}} - m_i g \mathbf{z}_{\mathcal{I}} + T_i \mathbf{q}_i, & (1.38a) \\ \mathbf{M}_i = J_i \dot{\mathbf{\Omega}}_i + S(\mathbf{\Omega}_i) J_i \mathbf{\Omega}_i, & (1.38b) \\ m_L \ddot{\mathbf{x}}_L = -\sum_i T_i \mathbf{q}_i - m_L g \mathbf{z}_{\mathcal{I}}. & (1.38c) \end{cases}$$

The authors introduce the following result

Lemma 1.2.2. *(Differential-Flatness of the* n *quadrotor, point-mass load system,* $n \geqslant 1$*, [Sreenath and Kumar, 2013]) $\mathcal{Y}_n = (\mathbf{x}_L, T_i \mathbf{q}_i, \psi_j)$, for $i \in \{2, \dots, n\}, j \in \{1, \dots, n\}$ is a set of flat outputs for the* n *quadrotor, point-mass load system, where ψ_j is the yaw angle of the jth quadrotor.*

Indeed, the dimension of the flat output is $3 + 3(n-1) + n = 4n$ which matches the number of independent controls stemming from the n quadrotors ; T_1 and $\mathbf{q}_1$ are obtained from equation (1.38c) while the $x_i, i \in \{2, \dots, n\}$ are obtained from equation (1.37). Since simple

quadrotors such as the one modeled here are known to be flat [Mellinger and Kumar, 2011] with flat output $(\mathbf{x}, \psi)$, we obtain all the remaining quantities from $(\mathbf{x}_i, \psi_i, T_i\mathbf{q}_i)$.

A similar lemma is shown for a rigid-body load, with orientation matrix $R_{\mathcal{B}}^{\mathcal{I}}$ and inertia matrix J_L. The cables are attached to the body at the fixed points $\mathbf{r}_i$ so that

$$\mathbf{x}_i = \mathbf{x}_L + R_{\mathcal{B}}^{\mathcal{I}}(\mathbf{r}_i - L_i\mathbf{q}_i) \tag{1.39}$$

Using again, as in [Michael et al., 2011] , the Moore-Penrose inverse, the authors introduce the following lemma.

Lemma 1.2.3. *(Differential-Flatness of the* n *quadrotor, rigid-body load system,* $n \geqslant 3$*, [Sreenath and Kumar, 2013]) $\mathcal{Y}_n = (\mathbf{x}_L, R_{\mathcal{B}}^{\mathcal{I}}, \Lambda, \psi_j)$, for $j \in \{1, \dots, n\}$ is a set of flat outputs for the* n *quadrotor, rigid-body load system, where $\Lambda \in \mathbb{R}^{3n-6}$ satisfies*

$$\underline{T} = \Phi^{+}W + N\Lambda, \tag{1.40}$$

with $\underline{T}, W$ defined as

$$\underline{T} = \begin{pmatrix} T_1\mathbf{q}_1 \\ T_2\mathbf{q}_2 \\ \vdots \\ T_n\mathbf{q}_n \end{pmatrix}, W = -\begin{pmatrix} R_{\mathcal{I}}^{\mathcal{B}}(m_L(\ddot{\mathbf{x}}_L + g\mathbf{z}_{\mathcal{I}})) \\ J_L\dot{\boldsymbol{\Omega}}_L + S(\boldsymbol{\Omega}_L)J_L\boldsymbol{\Omega}_L \end{pmatrix}, \tag{1.41}$$

where Φ^{+}, N are respectively the Moore-Penrose generalized inverse and the kernel of

$$\Phi = \begin{pmatrix} I & I & \dots & I \\ S(\mathbf{r}_1) & S(\mathbf{r}_2) & \dots & S(\mathbf{r}_n) \end{pmatrix} \tag{1.42}$$

As explained by the authors, the flat vector Λ can be chosen after W to ensure, for instance, that the tensions perform no isometric work. The output of the flatness generated trajectories is then used as an input to the geometric controller presented in [Lee et al., 2013].

The authors then address the case when a cable goes from being slack to taut. They assume there is a discrete change in the velocity of the system, which they model by a perfectly non-elastic collision. In such conditions, at some discrete times, some agents may enter or leave the system leading to different successive flat subsystems. They call this system a differentially-flat hybrid system.

However, the set of flat outputs makes the trajectory planning problem rather untractable. Even with the point-mass load, we cannot plan the trajectory of all the drones, but only the relative positions of $n-1$ of them with respect to the load. Moreover, the position of the first drone is only a result of the trajectories of the load and of the other drones and thus cannot be planned. As a consequence, in order to get an acceptable trajectory for this drone, the trajectory for the load is simple – an ellipse parallel to the ground – and orientations and tensions of the cables as constant as possible resulting in a similar trajectory for the first drone.

1.2.3 Collision avoidance on determined paths

In most cases, when planning trajectories, the agents are considered to be point-mass systems. However, real agents occupy a determined volume and collision avoidance has to be ensured

between neighboring agents. This can be considered either during planning by ensuring that the different paths are sufficiently far away. It can also be considered after the paths have been determined – without taking into account collision avoidance – by adapting the way the agents travel on the path. As explained by LaValle and Hutchinson, these two approaches are the two opposites of a spectrum going from *centralized* to *decoupled* multi-agent motion planning.

Indeed, in [LaValle and Hutchinson, 1998], the authors suggest two algorithms – which are exponentially difficult in time and space and thus not applicable to a lot of agents – to avoid collision on paths and roadmaps – a set of paths on which agents can choose between different paths at crossings – created by a decoupled planner. The problem is discretized in time and at every step Δt, the robots are either move at full speed, forward in the case of paths, forward or backward in the case of roadmaps, or to stop and wait at their current position.

The trajectories are optimized based on a cost function evaluating an energy-like function, which encompasses collision avoidance – for which the cost is either 0 or ∞ – and the accomplishment of the goal – for which the cost is again either 0 or ∞. The authors link it to the field of noncooperative game theory by showing that the minimum of this cost function corresponds to a Nash equilibrium. Indeed, the limit solution is not always optimal and there might be better solutions for cooperative agents.

Along load transportation for multiple agents, [Maza et al., 2010] introduced a global framework for task solving. This framework is built on successive steps. A certain number of tasks are allocated to the agents – which are supposed to move only in straight line. This allocation is equivalent to a trajectory planner without collision avoidance. After this planning step, a framework suggesting sequential path grants and clearances is applied to avoid collisions. The task allocation framework is established on market based algorithms developed by [Smith, 1980]. In a sense, since it is applied to allocate multiple waypoints, it is similar to a multi-agent Travelling Salesman problem. The agents hold auctions on the different tasks, this leads this step to be processed in a decentralized way. The solution to avoid collision during the completion of the different tasks resembles the airports' Air Traffic Control systems: the neighboring helicopters simply exchange sequential path grants and clearances.

Another domain where the paths are pre-planned is the domain of road crossings. The cars and trucks are bound to move on the tracks and know beforehand the direction they will choose while ignoring their neighbors' choices. Several works are conducted nowadays to create automated vehicles driving autonomously. While this topic of research has already some applications in highway traffic control, the problem of automatizing crossroads is still an open and challenging problem. Crossroads are indeed a dangerous bottleneck of traffic at which the agents rarely cooperate.

In [Gregoire et al., 2013], the authors decompose the crossroad management into two subproblems. First, a discrete algorithm is in charge of the prioritization of the vehicles entering the crossroad area. The set of vehicles in the area is represented as an acyclic priority graph: vehicles entering the area are added to the graph as new vertices. The ranking in the updated graph is not made on a first-come, first-served basis but on an optimization of the traffic at the crossroad. Second, a continuous control law is suggested that allows the vehicle a maximal acceleration while being robust to the maximal possible brake of other vehicles in the intersection. The implementation of the two algorithms is however made in a centralized way and needs a total collaboration of all the vehicles.

CHAPTER 2

Trajectory generation for PDE systems, existing works and available tools

Using the solution to a Partial Differential equations as a base for a trajectory planer is a method used by several authors as was shown by the review in the previous chapter. In this chapter, we present first various results on the resolution of the heat equation and of Burgers' equation, notably the concept of flat partial differential equation. Second, we introduce Gevrey functions, a class of functions widely used to generate solutions to flat partial differential equation.

Contents

1.1 Different methods of managing a multi-agent system **19**
1.1.1 Behavioral methods 20
1.1.2 Methods based on Particle Swarm Optimization 22
1.1.3 Potential methods 24
1.1.4 Graph based methods 27
1.1.5 Methods based on Partial Differential Equations 28
1.2 Different problems of collaborative systems **31**
1.2.1 Deployment problems 31
1.2.2 Cooperative transportation of a swinging load 35
1.2.3 Collision avoidance on determined paths 38

2.1 Existing works: an overview

The notion of flatness, that will be presented and used in the second part of this thesis, appeared as a convenient and constructive tool to plan motion for systems governed by ordinary differential equations. This idea was later applied to distributed parameter systems. A distributed parameter system is said to be *flat* if the system is entirely determined by a set of functions called *flat outputs of the system*. A good example can be found in [Laroche et al., 2000] where a one-dimensional rod – represented in figure 2.1 – following the heat equation is studied.

The rod is insulated at one end while heated at the other. The temperature of the rod $\theta(x,t)$ is the solution of the heat equation:

$$\begin{cases} \theta_t(x,t) &= \theta_{xx}(x,t), \qquad x \in [0,1] \\ \theta_x(0,t) &= 0 \\ \theta_x(1,t) &= u(t) \end{cases} \tag{2.1}$$

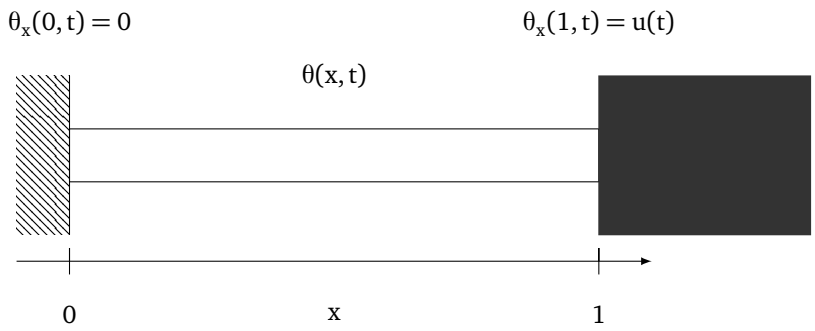


Figure 2.1 – The heated rod of [Laroche et al., 2000]

The authors show that the function $\theta(0,t)$ completely determines the heat of the rod $\theta(x,t)$ and, therefore, gives a way to create the open-loop controller $u(t) = \theta_x(1,t)$. A key notion for this result is the notion of Gevrey function. Gevrey functions – introduced in [Gevrey, 1918] are a way of estimating the divergence of the Taylor expansion of non-analytic functions.

Definition 2.1.1. Gevrey functions ([Laroche et al., 2000]) A smooth function $t \in [0,T] \mapsto y(t)$ is *Gevrey of order* α if:

$$\exists M, R > 0, \forall m \in \mathbb{N}, \sup_{t\in[0,T]} \left|y^{(m)}(t)\right| \leqslant M\frac{(m!)^\alpha}{R^m} \tag{2.2}$$

Gevrey functions are entire if $\alpha < 1$, are analytic if $\alpha = 1$ and have a divergent Taylor expansion if otherwise. A Gevrey function of order $\beta < \alpha$ is also Gevrey of order α. The space of Gevrey function of order α, G^α is stable by scaling, integration, addition, multiplication and composition.

The authors propose the solution in the form of:

$$\theta(x,t) = \sum_{i\geqslant 0} y^{(i)}(t)\frac{x^{2i}}{(2i)!}, \tag{2.3}$$

where $y(t) = \theta(0,t)$. The function $\theta(x,t)$ is a solution of equation (2.1). This solution defines the control:

$$u(t) = \sum_{i\geqslant 1} \frac{y^{(i)}(t)}{(2i-1)!}. \tag{2.4}$$

These formal series may be written with the help of the formal differential operators of infinite order (see e.g. [Rodino, 1993] for a complete introduction to these operators):

$$-C_x = \cosh\left(x\sqrt{\frac{d/dt}{\mu}}\right) \tag{2.5}$$

and:

$$-S_x = \sqrt{\frac{\mu}{d/dt}}\sinh\left(x\sqrt{\frac{d/dt}{\mu}}\right) \tag{2.6}$$

These formal differential operators and their application to the function y have a sense in the meaning of the following theorem:

Theorem 2.1.2. *[Laroche et al., 2000] When* $y(t)$ *is Gevrey of order* $\alpha < 2$*, the formal solution stated in equation (2.3) is Gevrey of order* α *in* t *and order* 1 *in* x *(and in particular the formal control introduced in equation (2.4) is Gevrey of order* α*).*

When $\alpha = 2$*, the same result holds provided* $R > 4$ *where* R *is defined in definition 2.1.1.*

The authors also introduce a motion planning framework and more generally study the linear diffusion equation. A thorough study of flatness for ordinary differential equation as well as for distributed parameter systems can be found in [Rudolph et al., 2003].

As was presented in subsection 1.1.5 – Methods based on Partial Differential Equations, PDEs have been used extensively to represent swarms of agents. In these representations, the solution $u(x,t)$ to the PDE represents the density of agents at point x. While this can be a good method to animate swarms of birds or locusts in computer animation it does not appear to be a good choice for a system of numerable agents having precise objectives.

The work by [Meurer and Krstić, 2011] introduced in subsection 1.2.1 – Deployment problems, suggests another solution: to let the solution $u(x,t)$ to the PDE represent the position of the agent x where x represents a topological – as opposite to geometric – position. The agents are ordered based on an arbitrary choice at time t. However using PDEs for formation deployment is challenging. Control and trajectory generation for PDEs is in many cases still an open problem. We present in this section some recent works on this topic centered on Burgers' equation and flatness-based control of PDEs.

[Frihauf and Krstic, 2011] propose to use a simple linear reaction-advection-diffusion equation of the form:

$$u_t(x,t) = u_{xx}(x,t) + bu_x(x,t) + \lambda u(x,t), x \in [0,1], t \in [0,1] \tag{2.7}$$

to generate two-dimensional deployment paths. Two approaches are proposed to generate the trajectories, either by considering u, b and λ to be real values and to generate two independent solutions for two independent one-dimensional problems or to consider them as complex-valued. In this case, the equation is known as Ginzburg-Landau equation and coupling between the two dimensions is introduced. In the first – real-valued – case, a classification of the basis functions of the non-zero equilibria is made based on the respective values of b and λ. For example in the case $b^2 > 4\lambda$, the basis functions are of the type $(e^{\sigma_0 x}, e^{\sigma_1 x})$. Using the various solutions introduced by this classification, one can generate a wide variety of equilibrium states for the formation such as Lissajous curves. The problem is a bit more difficult in the complex-valued case and only some specific cases are addressed.

The authors use the backstepping for PDEs approach (see e.g. [Krstić and Smyshlyaev, 2008] for a complete course on backstepping designs for the boundary control of PDEs) to create the controls for the anchor agent $u(0,t)$ and for the leader $u(1,t)$. This approach allows the stabilization of the formation through feedback. The feedback needs information on the positions of all the agents. An estimation of these positions can be obtained by an observer knowing only the position of the leader, of its nearest neighbor – in terms of communication topology – and eventually of the anchor agent.

Krstić et al. consider Burgers' equation. Introduced in the thirties by the Dutch physicist Jan Burgers, this equation is a one-dimensional PDEs that is used to model gas dynamics or traffic

flows. For a given viscosity μ, Burgers' viscous equation reads:

$$u_t(x,t) = \mu u_{xx}(x,t) - u_x(x,t)u(x,t), x \in [0,1]. \tag{2.8}$$

When the viscosity μ is null, the equations is called Burgers' inviscid equation. However, this equation has a completely different behavior than the viscous version and we will consider the sole viscous equation in the following. An interesting point of Burgers' equation is its low order together with its nonlinear character that allows shock-like equilibrium profiles. Another key point of this equation is the existence – used by the authors – of Hopf-Cole transformation, a nonlinear transformation allowing to turn Burgers' equation into the heat equation and which can be stated as follows:

Theorem 2.1.3 (Hopf-Cole transformation ([Hopf, 1950] [Cole et al., 1951])). *Let* $u(x,t)$ *be a solution to Burgers' equation:*

$$u_t(x,t) = \mu u_{xx}(x,t) - u_x(x,t)u(x,t), x \in [0,1]. \tag{2.9}$$

then the function v *defined by* $u = -2\mu\frac{v_x}{v}$ *is a solution to the heat equation:*

$$v_t(x,t) = \mu v_{xx}(x,t), x \in [0,1]. \tag{2.10}$$

Proof. We may rewrite equation (2.9) as:

$$u_t = \mu u_{xx} - \frac{1}{2}(u^2)_x \tag{2.11}$$

We may then introduce $\tilde{u}_x = u$. The previous equation then reads:

$$\tilde{u}_{xt} = \mu \tilde{u}_{xxx} - \frac{1}{2}((\tilde{u}_x)^2)_x \tag{2.12}$$

Integrating with respect to x (and omitting the integration constant) leads to:

$$\tilde{u}_t = \mu \tilde{u}_{xx} - \frac{1}{2}(\tilde{u}_x)^2 \tag{2.13}$$

Introducing $v(x,t)$, a strictly positive function on the domain, so that $\tilde{u} = -2\mu \ln v$ transforms Burgers' equation into the heat equation:

$$v_t = \mu v_{xx} \tag{2.14}$$

together with:

$$u = -2\mu\frac{v_x}{v} \tag{2.15}$$

□

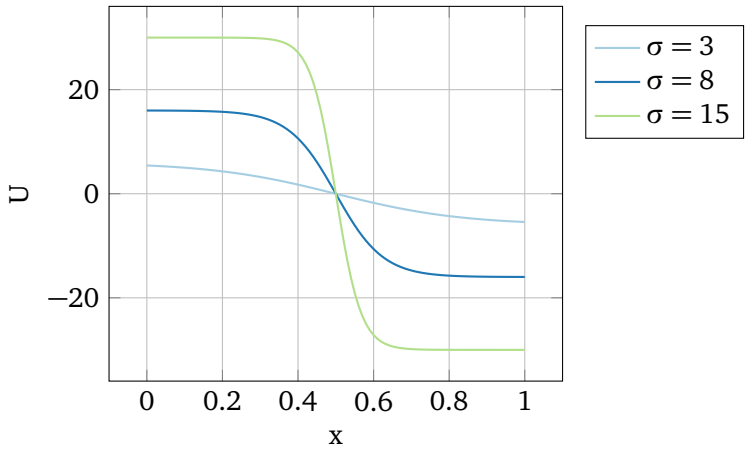


Figure 2.2 – Shock-like equilibrium profiles for Burgers' equation

The authors study the stabilization of "shock-like" equilibrium profiles – as plotted in figure 2.2 for different values of the parameter σ – of the form:

$$U(x) = -2\sigma \tanh(\sigma(x - 1/2)), \sigma \leqslant 0, \tag{2.16}$$

with the use of the two control inputs:

$$u_x(0,t) = \omega_0(t), \qquad u_x(1,t) = \omega_1(t). \tag{2.17}$$

The authors make use of Hopf-Cole transformation to transform the equation into a linear reaction-diffusion equation.They then stabilize the resulting PDE using linear backstepping which induces a nonlinear control for the original equation.

In a companion paper, [Krstić et al., 2009] study trajectory generation and tracking for the same problem. The authors show explicit expression of the controls ω_0 and ω_1 so that the system follows the reference trajectory $u^r(0,t) = b + a \sin \omega t$. However, the open-loop controller is not generally stable along the trajectory. The authors therefore introduce a nonlinear closed-loop controller using the backstepping-based controller introduced in the companion paper.

We may recall here another article by [Meurer and Krstić, 2011] presented in subsection 1.2.1 – Deployment problems which showed a custom version of Burgers' equation to be flat. This enabled the authors to present a framework for the motion planning based on the broader class of Gevrey functions.

There are other solutions to study Burgers' equation. For example, [Gorguis, 2006] applies the Adomian decomposition method and compares its results to those obtained through the Hopf-Cole transformation on some simple illustrative examples. Applying Adomian decomposition method to Burgers' equation consists in rewriting the equation as:

$$u(x,t) = u(x,0) + \mu L^{-1} u_{xx}(x,t) - L^{-1} u_x(x,t) u(x,t), x \in [0,1], t \in [0,1]. \tag{2.18}$$

where $L^{-1}(\cdot) = \int_0^1 (\cdot) dt$ is the formal integration operator. The solution $u(x,t)$ is replaced by the decomposition series

$$u(x,t) = \sum_{n \geqslant 0} u_n(x,t), \tag{2.19}$$

while the nonlinear term uu_x is replaced by Adomain polynomials:

$$uu_x = \sum_{n \geqslant 0} A_n. \tag{2.20}$$

Identifying the components leads to the recursive problem:

$$\begin{cases} u_0(x,t) & = u(x,0) \\ u_{n+1}(x,t) & = L^{-1}(u_n)_{xx} - L^{-1}A_n, n \geqslant 0 \end{cases} \tag{2.21}$$

It is however important to consider the existence of the conditions we are searching for. Indeed, it is not always ensured that controls can be found to match a desired final state. In this work, we consider the system defined by Burgers' equation and the initial state:

$$\begin{cases} u_t(x,t) & = \mu u_{xx}(x,t) - u_x(x,t)u(x,t) \\ u(x,0) & = u_0(x) \end{cases}, \tag{2.22}$$

and we want to know if it is possible to find two controls:

$$\begin{cases} u(0,t) & = v_0(t) \\ u(1,t) & = v_1(t) \end{cases}, \tag{2.23}$$

so that at final time T we can achieve a desired state w(x):

$$u(x,T) = w(x) \tag{2.24}$$

[Glass and Guerrero, 2007] consider the system defined by equations (2.22) to (2.23) and show the exact global controllability to non-zero constant states. That is:

Theorem 2.1.4. *Exact global controllability to non-zero constant states [Glass and Guerrero, 2007] There is a constant $\alpha_0 \geqslant 1$ such that for any $M \in \mathbb{R}\backslash\{0\}$ there exists $\mu_0 > 0$ such that for any $u_0 \in L^\infty([0,1])$, any time $T > \alpha_0/|M|$ and any $\mu \in (0,\mu_0)$ there exist controls v_0^μ and v_1^μ satisfying the following properties:*

- *$\|v_0^\mu\|_\infty$ and $\|v_1^\mu\|_\infty$ are uniformly bounded for $\mu \in (0,\mu_0)$, that is to say, there exists a constant C_{α_0} such that:*

$$\|v_0^\mu\|_\infty + \|v_1^\mu\|_\infty \leqslant C_{\alpha_0}(\|u_0\|_\infty + M). \tag{2.25}$$

- *The solution u of equations (2.22) and (2.23) associated to $v_0 = v_0^\mu$ and $v_1 = v_1^\mu$ satisfies:*

$$u(x,T) = M \tag{2.26}$$

The first point is rather encouraging since the controls are finite. Furthermore, the theorem is shown to hold for $\alpha_0 = 9$ and a corollary is given for $\mu = 1$ that says that the result also holds as long as M is large enough. As a consequence, to find controls to achieve a deployment to a constant state, it can be necessary to consider changing the goal time. Indeed, [Coron, 2007] shows another result using Hopf-Cole transfomation:

Theorem 2.1.5. *[Coron, 2007] For every* $T > 0$*, there exists* $M > 0$ *such that, for every* $C \in \mathbb{R}$ *with* $|C| \geqslant M$*, there exists* $y \in L^2((0,T) \times (0,1))$ *satisfying equation (2.22) with* $u_0(x) = 0$ *and such that:*

$$u_T(x) = C \tag{2.27}$$

It is thus possible in a chosen time to perform transition from a null-state to a constant state as long as this state is not too close to zero.

Indeed, [Guerrero and Imanuvilov, 2007] show that global null controllability for Burgers' equation with two control forces defined by equations (2.22) and (2.23) does not hold, nor does exact controllability. As a consequence, in the following we will try to find controls and final deployment bearing these controllability results in mind.

2.2 Differential flatness and Gevrey functions

2.2.1 Gevrey functions, definition and examples

The class of Gevrey functions Γ_s [Gevrey, 1918] is a class of function naturally appearing in the studies of the heat equation. Gevrey functions of order α are C^∞ functions, which Taylor series – in the case where $\alpha > 1$ – do not converge.

Definition 2.2.1. Gevrey function [Laroche et al., 2000] A function $\varphi \in C^\infty$ over K is Gevrey of order $\alpha \geqslant 1$ if there exist $M > 0$, $R > 0$ so that for every n:

$$\sup_{t \in K} |\varphi^{(n)}(t)| \leqslant M \frac{(n!)^\alpha}{R^n} \tag{2.28}$$

The class of Gevrey functions of order σ, Γ_σ is stable by scaling, integration, derivation, sum, multiplication and composition.

Whereas analytic functions being constant on an open set have to be constant everywhere – these functions are equal to their Taylor expansion and, indeed, all of their derivatives vanish on the set – non-analytic functions – and especially Gevrey functions of order $\sigma > 1$ – can be constant on an open set without being constant everywhere. As such, they are extensively used to design transitions between constant states for differential systems of infinite order.

The stability of the Γ_σ class can also be used in the case of the formal differential operators of infinite order C_x and S_x presented in equations (2.5) and (2.6) on page 42:

Proposition 2.2.2. *[Laroche et al., 2000] When* $\varphi(t)$ *is Gevrey of order* $\sigma < 2$*,* $C_x\varphi(t)$ *is Gevrey of order* σ *in* t *and order* 1 *in* x*.*

When $\sigma = 2$*, the same result holds provided* $R > 4$ *(*R *being the "Gevrey Radius" as defined in definition 2.2.1)*

Proof. The proof is given in [Laroche et al., 2000]. □

Corollary 2.2.3. *Proposition 2.2.2 also holds for* $S_x\varphi(t)$.

Proof. This result may either be proved using a computational proof similar to [Laroche et al., 2000] or make the following statement: Since $\partial_x C_x = ((d/dt)/\mu)S_x$ and since the Gevrey property is stable by scaling and derivation, if $\varphi(t)$ is Gevrey, $C_x\varphi(t)$ is also Gevrey and thus $S_x\varphi(t)$ is Gevrey of order α in t and order 1 in x. □

A common example of basis Gevrey function is the "bump function", used for example in [Laroche et al., 2000; Crépeau and Prieur, 2008; Meurer and Krstić, 2011], depicted in figure 2.3:

$$\varphi_\sigma(t) = \begin{cases} \exp\left(\dfrac{-1}{(t(1-t))^\sigma}\right) & \text{if} \in]0,1[\\ 0 & \text{elsewhere} \end{cases} \tag{2.29}$$

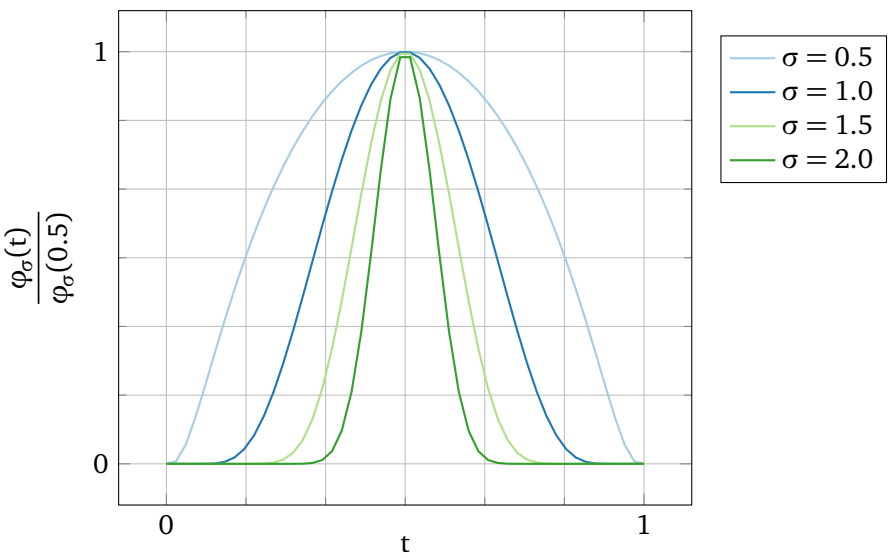


Figure 2.3 – Normalized value of $\varphi_\sigma(t)$ for various value of σ.

This function is Gevrey of order $\sigma = 1 + 1/\sigma$ whatever $\sigma > 0$. According to definition 2.2.1, the following function – obtained as the integration of equation (2.29) – is also Gevrey of order $1 + 1/\sigma$.

$$\Phi_\sigma(t) = \begin{cases} 0 & \text{if } t \leqslant 0 \\ \dfrac{\int_0^t \varphi_\sigma(\tau)d\tau}{\int_0^1 \varphi_\sigma(\tau)d\tau} & \text{if } t \in]0\,;1[\\ 1 & \text{if } t \geqslant 1 \end{cases} \tag{2.30}$$

This function, represented in figure 2.4, allows for a C^∞ finite time transition between two constant states. It will be the base function of our solution. Other functions are commonly in use to perform similar finite time C^∞ transitions such as [Rudolph et al., 2003]:

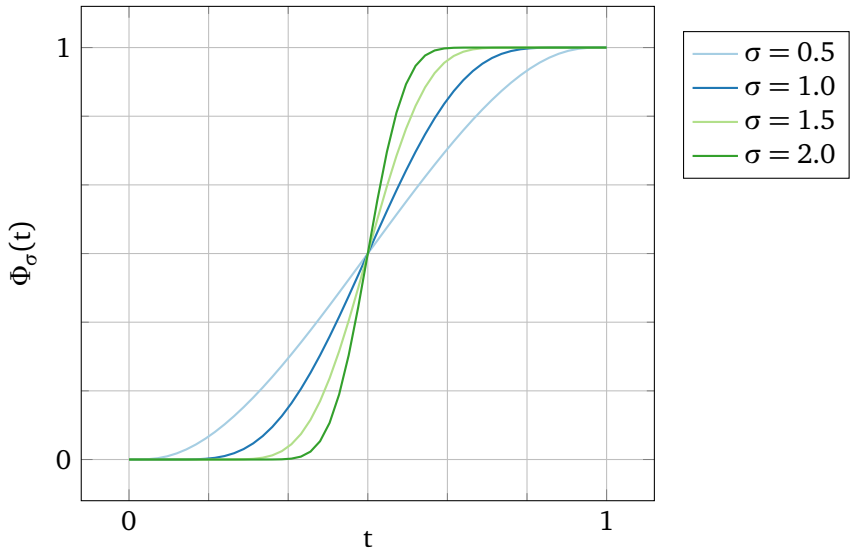


Figure 2.4 – The basis function $\Phi_\sigma(t)$ for various values of σ.

$$\Phi_\sigma(t) = \begin{cases} 0 & \text{if } t \leqslant 0 \\ \frac{1}{2}\left(1 + \tanh \frac{2(2t-1)}{(4t(1-t))^\sigma}\right) & \text{if } t \in]0\,;1[\ , \\ 1 & \text{if } t \geqslant 1 \end{cases} \tag{2.31}$$

or [Martin et al., 2014]:

$$\Phi_\sigma(t) = \begin{cases} 0 & \text{if } t \leqslant 0 \\ \dfrac{\exp\left(-(1-t)^\sigma\right)}{\exp\left(-(1-t)^\sigma\right) + \exp\left(-t^\sigma\right)} & \text{if } t \in]0\,;1[\ , \\ 1 & \text{if } t \geqslant 1 \end{cases} \tag{2.32}$$

These two examples of transition functions are Gevrey of order $1+\frac{1}{\sigma}$ and have various advantages. The decision to choose the function defined in equations (2.29) and (2.30) was motivated by the existence of the efficient algorithm for derivatives computation presented in the following.

2.2.2 Efficient computation of the derivatives of Φ_σ

Inspired by the method presented in [Rudolph et al., 2003] for the computation of their basis function, we present an algorithm to efficiently compute the derivatives of the function we selected. This calculation was already conducted by Pierre Rouchon and can be found in the form of source code on the cdrom example provided with [Rudolph et al., 2003]. Nonetheless, it is of interest to explicitly present this algorithm in this work. We consider the function:

$$\Phi_\sigma(t) = \exp\left(\frac{-1}{(t(1-t))^\sigma}\right) = y_0 \tag{2.33}$$

Where σ is a positive constant. We search for a way to efficiently evaluate, for each value of t:

$$\frac{d^n}{dt^n}\Phi_\sigma(t) = y_n \tag{2.34}$$

In the following, we introduce the two sequences of functions:

$$\begin{cases} a_0 & = (t(1-t))^{\sigma+1} \\ a_{k+1} & = \dfrac{d}{dt}a_k, k \geqslant 0 \end{cases} \tag{2.35}$$

And:

$$\begin{cases} p_0 & = t(1-t) \\ p_{k+1} & = \dfrac{d}{dt}p_k, k \geqslant 0 \end{cases} \tag{2.36}$$

In terms of a_k and p_k, the function y_0 reads

$$y_0 = \exp\left(-\frac{p_0}{a_0}\right) \tag{2.37}$$

Differentiating once this relation reads:

$$a_0^2 y_1 = (a_1 p_0 - p_1 a_0) y_0 \tag{2.38}$$

From the definition of a_k, we find the following relation:

$$a_1 p_0 = (\sigma + 1) p_1 a_0 \tag{2.39}$$

Thus, equation (2.38) reads:

$$a_0 y_1 = \sigma p_1 y_0 \tag{2.40}$$

Taking the n-th derivative of the previous equation leads to:

$$\sum_{k=0}^{n} \binom{n}{k} y_{n+1-k} a_k = \sigma \sum_{k=0}^{n} \binom{n}{k} y_{n-k} p_{k+1} \tag{2.41}$$

The only non-zero values of p_i are for $i \leqslant 2$. Thus, the previous equation reads:

$$\sum_{k=0}^{n} \binom{n}{k} y_{n+1-k} a_k = \sigma p_1 y_n + \sigma n p_2 y_{n-1} \tag{2.42}$$

To find a handy expression of a_k, we take the k-th derivative of equation (2.39):

$$\sum_{i=0}^{k} \binom{k}{i} a_{k+1-i} p_i = (\sigma + 1) \sum_{i=0}^{k} \binom{k}{i} a_{k-i} p_{i+1} \tag{2.43}$$

Keeping the only non-zero values of p_i, this can be written as:

$$p_0 a_{k+1} = (\sigma + 1 - k) p_1 a_k + \left(\sigma + 1 - \frac{k-1}{2}\right) k p_2 a_{k-1} \tag{2.44}$$

This relation is used to efficiently compute the successive derivatives of a_0. Considering equation (2.42), assuming we have all the necessary y_k and a_k for $0 \leqslant k \leqslant n$, we may compute y_{n+1} as:

$$y_{n+1} a_0 = (\sigma p_1 - n a_1) y_n + n \left(\sigma p_2 - \frac{n-1}{2} a_2\right) y_{n-1} - \sum_{k=3}^{n} \binom{n}{k} y_{n+1-k} a_k \tag{2.45}$$

This computation appears efficient considering both the computational effort and the needed memory space. For each new degree only two terms are summed for a_{k+1} and k for y_{k+1} and two new values, a_{k+1} and y_{k+1} are kept in memory.

CHAPTER 3

Formal solutions to the heat equation

This chapter presents our main contribution to the study of the heat equation with controls on both sides. Based on a famous work by Holmgren, we derive the expression of a new formal differential operator of infinite order and study its effect on some simple basis functions. We also study its computational evaluation and convergence. We present finally the Gevrey controls used to generate solutions to the heat equation with controls on both sides allowing finite time transition between two states and present a way to generate them.

Contents

2.1 Existing works: an overview **41**
2.2 Differential flatness and Gevrey functions **47**
2.2.1 Gevrey functions, definition and examples 47
2.2.2 Efficient computation of the derivatives of Φ_σ 49

3.1 Rewriting the heat equation with formal differential operators of infinite order

In a note to the French Academy of Sciences, [Holmgren, 1908] studied the solution $z(x,y)$[1] to the heat equation

$$\partial_x^2 z - \partial_y z = 0. \tag{3.1}$$

on a domain Γ bounded by the arc AB of equation $x = \chi(y)$. Using the change of variable

$$\begin{aligned} \xi &= x - \chi(y) \\ \eta &= y. \end{aligned} \tag{3.2}$$

and introducing the two functions

$$\begin{aligned} (z)_{\xi=0} &= \varphi(\eta) \\ \left(\partial_\xi z\right)_{\xi=0} &= \psi(\eta). \end{aligned} \tag{3.3}$$

Holmgren proposed to study a series expansion of z and to identify the terms and showed that it was absolutely convergent in the case:

$$|\varphi^{(n)}(y)| < \frac{M(2n)!}{\rho^n}, \quad |\psi^{(n)}(y)| < \frac{M(2n)!}{\rho^n} \tag{3.4}$$

[1] We use in the following Holmgren's notations. The y variables is of course a time variable.

and regular if

$$|\varphi^{(n)}(y)| < \frac{M(n(1+\alpha))!}{\rho^n}, \quad |\psi^{(n)}(y)| < \frac{M(n(1+\alpha))!}{\rho^n}, 0 < \alpha < 1 \tag{3.5}$$

and that, in the special case where $\chi(x) = x_0$, this solution could be written:

$$z(x,y) = \sum_{n=0}^{\infty} \frac{\varphi(n)(y)}{(2n)!}(x-x_0)^{2n} + \sum_{n=0}^{\infty} \frac{\psi(n)(y)}{(2n+1)!}(x-x_0)^{2n+1} \tag{3.6}$$

Indeed, if we consider now the heat equation with respect to time and if we introduce the conductivity term μ, the genuine heat equation simply reads:

$$\partial_t \varphi(x,t) = \mu \partial_x^2 \varphi(x,t), \tag{3.7}$$

using operational calculus (a formal transformation similar to Laplace transform) leads to:

$$s\hat{\varphi}(x,s) = \mu \partial_x^2 \hat{\varphi}(x,s), \tag{3.8}$$

where s is equivalent to the partial time derivation operator $D_t = \partial_t$. The resulting equation is an ordinary differential equation in x. Solving this differential equation results in the system:

$$\begin{cases} \widehat{\varphi}(x,s) & = \widehat{C}_x \widehat{\lambda}_1(s) + \widehat{S}_x \widehat{\lambda}_2(s) \\ \widehat{C}_x & = \cosh\left(x\sqrt{\frac{s}{\mu}}\right) \\ \widehat{S}_x & = \sqrt{\frac{\mu}{s}} \sinh\left(x\sqrt{\frac{s}{\mu}}\right) \end{cases} . \tag{3.9}$$

Reverting the formal transformation of D_t into s, we get:

$$\varphi(x,t) = C_x \lambda_1(t) + S_x \lambda_2(t), \tag{3.10}$$

where:

$$\begin{aligned} C_x &= \cosh\left(x\sqrt{\frac{d/dt}{\mu}}\right) \\ S_x &= \sqrt{\frac{\mu}{d/dt}} \sinh\left(x\sqrt{\frac{d/dt}{\mu}}\right) \end{aligned} \tag{3.11}$$

are differential operators of infinite order [Rodino, 1993]. Using the Taylor expansion of cosh and sinh, these operators may be equivalently written as series of the form:

$$\begin{aligned} C_x &= \sum_{k\geqslant 0} \frac{1}{\mu^k} \frac{x^{2k}}{(2k)!} \frac{d^k}{dt^k} \quad , \\ S_x &= \sum_{k\geqslant 0} \frac{1}{\mu^k} \frac{x^{2k+1}}{(2k+1)!} \frac{d^k}{dt^k} \quad . \end{aligned} \tag{3.12}$$

When evaluating these operators in $x = 0$, equation (3.12) reads:

$$\begin{aligned} C_0 &= 1 \\ S_0 &= 0 \quad . \end{aligned} \tag{3.13}$$

Furthermore, differentiating both operators with respect to x leads to:

$$\begin{aligned} \partial_x C_x &= \frac{d/dt}{\mu} S_x \quad , \\ \partial_x S_x &= C_x \quad . \end{aligned} \tag{3.14}$$

If we take the partial derivative of equation (3.10) with respect to x, φ_x reads:

$$\varphi_x(x,t) = \frac{1}{\mu}\frac{d}{dt}(S_x\lambda_1(t)) + C_x\lambda_2(t). \tag{3.15}$$

Evaluating equations (3.10) and (3.15) in $x = 0$ then leads to:

$$\begin{cases} \varphi(0,t) & = \lambda_1(t), \\ \varphi_x(0,t) & = \lambda_2(t). \end{cases} \tag{3.16}$$

We can then replace the functions λ_1 and λ_2 by their respective values in equation (3.10). This leads us to the expression:

$$\varphi(x,t) = C_x\varphi_0(t) + S_x\varphi_{x,0}(t) \tag{3.17}$$

The heat equation is thus said to be flat with $(\varphi_0, \varphi_{x,0})$ being one of the possible flat output where $\varphi_0(t)$ stands for $\varphi(x = 0, t)$ and $\varphi_{x,0}(t)$ stands for $(\partial\varphi/\partial x)(x = 0, t)$. Flatness of differential systems of infinite order is an extension of the concept of flatness for differential systems of finite order which will be used in the second part of this work.

3.2 The heat equation with controls on both sides

3.2.1 Objectives

The flat output of the heat equation and conditions considered in [Laroche et al., 2000] appears as a natural choice for a problem of heat transfer. In the case of a motion planning problem for a multi-agent system based on the heat equation, it seems reasonable to parametrize the trajectories in terms of positions of the agents. Our choice is to find an expression of the trajectories of the whole formation as the output of the trajectories of two agents. This leads us to investigate the expression of the heat equation based on the temperatures at both ends.

3.2.2 Formal derivation

The set of flat outputs used in equation (3.17) may be a reasonable choice in some applications (as for example in [Laroche et al., 2000]) but not in our case. Indeed, we want to control the trajectories of our agents on both sides of the equation.

The idea of our transformation is then to move the natural flat output in equation (3.17) constituted of the heat φ_0 and the flux $\varphi_{x,0}$ on one side to an expression based on the heat on both sides. To do so, we evaluate equation (3.17) in $x = 1$.

$$\varphi_1(t) = C_1\varphi_0(t) + S_1\varphi_{x,0}(t). \tag{3.18}$$

The functions $\varphi_0(t)$ and $\varphi_1(t)$ are our control inputs. In order to express φ in terms of φ_0 and φ_1 we want to express $\varphi_{x,0}$ in term of these functions and to use the obtained expression in equation (3.17). We may formally invert the S_1 operator in equation (3.18) to obtain:

$$\varphi_{x,0}(t) = (S_1)^{-1}\left(\varphi_1(t) - C_1\varphi_0(t)\right). \tag{3.19}$$

Injecting equation (3.19) into equation (3.17) leads to:

$$\varphi(x,t) = \left(C_x - S_x(S_1)^{-1}C_1\right)\varphi_0(t) + S_x(S_1)^{-1}\varphi_1(t). \tag{3.20}$$

The operator $(S_1)^{-1}$ is formally defined by:

$$(S_1)^{-1}\sqrt{\frac{d/dt}{\mu}}\operatorname{csch}\left(\sqrt{\frac{d/dt}{\mu}}\right) \tag{3.21}$$

where csch is the cosecant hyperbolic function. Its series expansion is [Abramowitz and Stegun, 1965, 4.5.65 p. 85]

$$(S_1)^{-1} = -\sum_{k\geqslant 0}\frac{1}{\mu^k}\frac{2(2^{2k-1}-1)B_{2k}}{(2k)!}\frac{d^k}{dt^k} \tag{3.22}$$

where B_{2k} is the 2k-th Bernoulli number. Then, we propose to introduce the formal differential operator of infinite order $T_x = S_x(S_1)^{-1}$ and study its properties. We may first give an explicit expression of this operator:

Proposition 3.2.1. *The differential operator* $T_x = S_x(S_1)^{-1}$ *can be expressed as:*

$$T_x = \sum_{k\geqslant 0}\frac{1}{\mu^k}\frac{2^{2k+1}}{(2k+1)!}B_{2k+1}\left(\frac{1+x}{2}\right)\frac{d^k}{dt^k} \tag{3.23}$$

where $B_k(x)$ *is the* k*-th Bernoulli polynomial defined by [Abramowitz and Stegun, 1965, p. 804 23.1.1]:*

$$\frac{te^{xt}}{e^t-1} = \sum_{k\geqslant 0}B_k(x)\frac{t^k}{k!}, \quad |t| < 2\pi \tag{3.24}$$

Proof. Using the formal series expression of the operator S_x given in equation (3.12), the operator T_x can be written as:

$$T_x = -\sum_{k\geqslant 0}\frac{1}{\mu^k}\frac{x^{2k+1}}{(2k+1)!}\frac{d^k}{dt^k}\left(\sum_{l\geqslant 0}\frac{1}{\mu^l}\frac{2(2^{2l-1}-1)B_{2l}}{(2l)!}\frac{d^l}{dt^l}\right) \tag{3.25}$$

This can be also written as:

$$T_x = -\sum_{k\geqslant 0}\sum_{l\geqslant k}\frac{1}{\mu^l}\frac{x^{2k+1}}{(2k+1)!}\frac{2(2^{2(l-k)-1}-1)B_{2(l-k)}}{(2(l-k))!}\frac{d^l}{dt^l} \tag{3.26}$$

Exchanging the two sum operators in the previous equation leads to:

$$T_x = -\sum_{l\geqslant 0} \frac{1}{\mu^l} \frac{2^{2l+1}}{(2l+1)!} \sum_{k=0}^{l} \binom{2l+1}{2k+1} B_{2(l-k)} \left(\frac{x^{2k+1}}{2^{2k+1}} - \frac{x^{2k+1}}{2^{2l}} \right) \frac{d^l}{dt^l} \tag{3.27}$$

We recognize here two odd parts of the B_{2k+1} Bernoulli polynomial [Abramowitz and Stegun, 1965, p. 804 23.1.2 and 23.1.7]:

$$B_n(x) = \sum_{l=0}^{k} \binom{k}{l} B_{k-l} x^k \tag{3.28}$$

Indeed:

$$\sum_{k=0}^{l} \binom{2l+1}{2k+1} B_{2(l-k)} \left(\frac{x^{2k+1}}{2^{2k+1}} - \frac{x^{2k+1}}{2^{2l}} \right) = \frac{1}{2} \left(B_{2l+1}\left(\frac{x}{2}\right) - B_{2l+1}\left(-\frac{x}{2}\right) - \frac{1}{2^{2l}} \left(B_{2l+1}(x) - B_{2l+1}(-x) \right) \right) \tag{3.29}$$

However, Bernoulli polynomials have the following property [Abramowitz and Stegun, 1965, p. 804 23.1.9]:

$$(-1)^n B_n(-x) = B_n(x) + n x^{n-1}, \quad n \geqslant 0 \tag{3.30}$$

Thus:

$$\begin{aligned} -B_{2l+1}(-x) &= B_{2l+1}(x) + (2l+1)x^{2l} \\ -B_{2l+1}\left(-\frac{x}{2}\right) &= B_{2l+1}\left(\frac{x}{2}\right) + (2l+1)\left(\frac{x}{2}\right)^{2l} \end{aligned}$$

And thus, equation (3.29) reads:

$$\sum_{k=0}^{l} \binom{2l+1}{2k+1} B_{2(l-k)} \left(\frac{x^{2k+1}}{2^{2k+1}} - \frac{x^{2k+1}}{2^{2l}} \right) = B_{2l+1}\left(\frac{x}{2}\right) - \frac{1}{2^{2l}} B_{2l+1}(x) \tag{3.31}$$

However, this can be further simplified using Raabe's multiplication theorem [Abramowitz and Stegun, 1965, p. 804 23.1.10]:

$$B_n(mx) = m^{n-1} \sum_{k=0}^{m-1} B_n\left(x + \frac{k}{m}\right), n \geqslant 0, m \geqslant 1 \tag{3.32}$$

This allows us to write:

$$\begin{aligned} B_{2k+1}(x) &= 2^{2k} \sum_{k=0}^{1} B_{2k+1}\left(\frac{x}{2} + \frac{k}{2}\right) \\ &= 2^{2k} \left(B_{2k+1}\left(\frac{x}{2}\right) + B_{2k+1}\left(\frac{1+x}{2}\right) \right) \end{aligned}$$

which, used in equation (3.31) gives:

$$\sum_{k=0}^{l} \binom{2l+1}{2k+1} B_{2(l-k)} \left(\frac{x^{2k+1}}{2^{2k+1}} - \frac{x^{2k+1}}{2^{2l}} \right) = -B_{2l+1}\left(\frac{1+x}{2}\right) \tag{3.33}$$

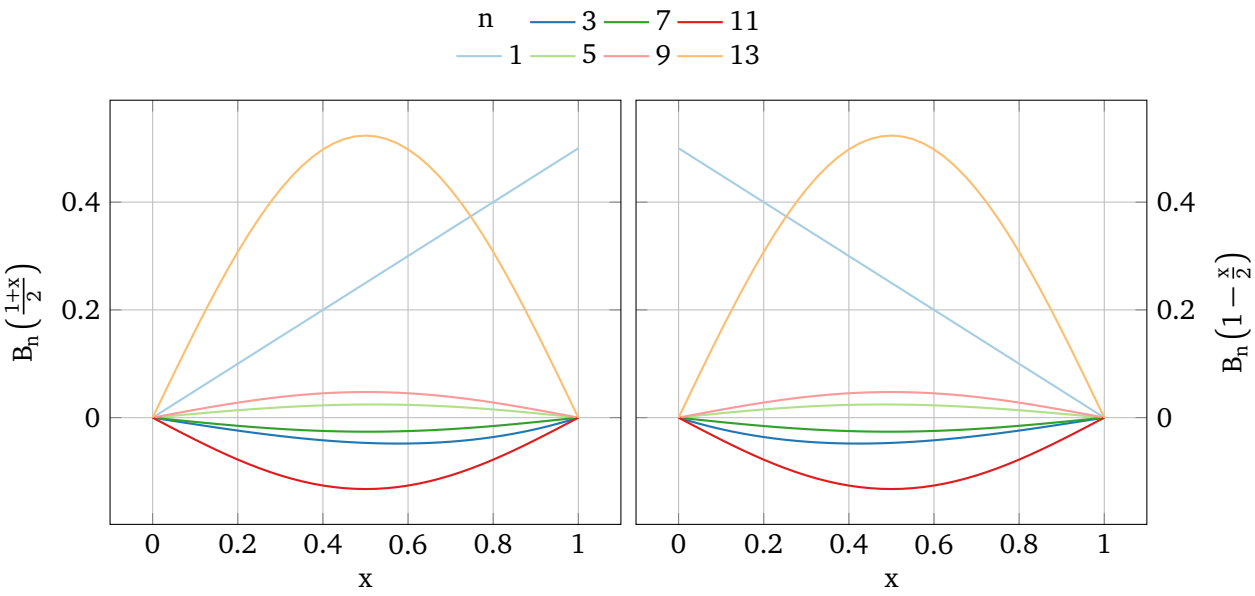


Figure 3.1 – First odd order Bernoulli polynomials used by the T_x operator (left) and by T_{1-x} (right).

Using this result in equation (3.27) leads to:

$$T_x = \sum_{k\geqslant 0} \frac{1}{\mu^k} \frac{2^{2k+1}}{(2k+1)!} B_{2k+1}\left(\frac{1+x}{2}\right) \frac{d^k}{dt^k} \tag{3.34}$$

This ends the proof. □

Bernoulli polynomials are omnipresent in numbers theory and in the study of affiliated functions such as the Riemann Zeta function. They have among other specificities, the remarkable particularity of having a constant number of solutions on the unity axis. Above order one, even order Bernoulli polynomials have exactly two zeros on the unity axis while odd order ones have three at the extremities and in the middle of the unity axis. The first odd order Bernoulli polynomials are represented in figure 3.1.

In equation (3.20), the control on the two sides play similar roles. It is therefore tempting to find a symmetric (in φ_0 and φ_1) expression of the solution. A good guess could, for example, be:

$$\varphi(x,t) = T_{1-x}\varphi_0(t) + T_x\varphi_1(t) \tag{3.35}$$

To prove this hypothesis, we further need in equation (3.20) an expression of $C_x - T_x C_1$. Bearing this in mind, we search for an analytical expression of this formal differential operator and state the following result:

Proposition 3.2.2. *The operator* $T_x C_1$ *can be written under the form:*

$$T_x C_1 = C_x + \sum_{k\geqslant 0} \frac{1}{\mu^k} \frac{2^{2k+1}}{(2k+1)!} B_{2k+1}\left(\frac{x}{2}\right) \frac{d^k}{dt^k} \tag{3.36}$$

Proof. Applying the operator T_x given in proposition 3.2.1 to the operator C_1 as given by equation (3.12), we can state:

$$T_x C_1 = \sum_{k\geqslant 0}\sum_{l\geqslant 0} \frac{1}{\mu^{k+l}} \frac{2^{2k+1}}{(2l)!(2k+1)!} B_{2k+1}\left(\frac{1+x}{2}\right)\frac{d^{k+l}}{dt^{k+l}} \tag{3.37}$$

This expression can be rewritten as:

$$T_x C_1 = \sum_{k\geqslant 0}\sum_{l\geqslant k} \frac{1}{\mu^{l}} \frac{2^{2k+1}}{(2(l-k))!(2k+1)!} B_{2k+1}\left(\frac{1+x}{2}\right)\frac{d^{l}}{dt^{l}} \tag{3.38}$$

Exchanging the two sum operators leads to:

$$T_x C_1 = \sum_{l\geqslant 0} \frac{1}{\mu^{l}} \frac{2^{2l+1}}{(2l+1)!} \left(\sum_{k=0}^{l}\binom{2l+1}{2k+1} B_{2k+1}\left(\frac{1+x}{2}\right)\frac{1}{2^{2(l-k)}}\right)\frac{d^{l}}{dt^{l}} \tag{3.39}$$

Using the expansion property of Bernoulli Polynomials [Abramowitz and Stegun, 1965, p. 804 23.1.7]:

$$B_n(x+h) = \sum_{k=0}^{n}\binom{n}{k} B_k(x)h^{n-k}, n \geqslant 0 \tag{3.40}$$

We can state:

$$\sum_{k=0}^{l}\binom{2l+1}{2k+1} B_{2k+1}\left(\frac{1+x}{2}\right)\frac{1}{2^{2(l-k)}} = \frac{1}{2}\left(B_{2l+1}\left(1+\frac{x}{2}\right) - B_{2l+1}\left(1-\frac{x}{2}\right)\right) \tag{3.41}$$

However, we can deduce from the differences and symmetry properties of Bernoulli polynomials that:

$$\begin{aligned} B_{2l+1}\left(1-\frac{x}{2}\right) &= -B_{2l+1}\left(\frac{x}{2}\right) \\ B_{2l+1}\left(1+\frac{x}{2}\right) &= B_{2l+1}\left(\frac{x}{2}\right) + (2l+1)\left(\frac{x}{2}\right)^{2l} \end{aligned} \tag{3.42}$$

And thus:

$$T_x C_1 = \sum_{l\geqslant 0} \frac{1}{\mu^{l}} \frac{2^{2l+1}}{(2l+1)!} B_{2l+1}\left(\frac{x}{2}\right)\frac{d^{l}}{dt^{l}} + \sum_{l\geqslant 0} \frac{1}{\mu^{l}} \frac{x^{2l}}{(2l)!}\frac{d^{l}}{dt^{l}} \tag{3.43}$$

Which ends the proof. □

Lemma 3.2.3. *Since $B_{2k+1}(x)$ Bernoulli polynomials are odd around $x = \frac{1}{2}$, we can write ([Abramowitz and Stegun, 1965, p. 804 23.1.8]):*

$$B_{2l+1}\left(1-\frac{x}{2}\right) = -B_{2l+1}\left(\frac{x}{2}\right) \tag{3.44}$$

And equation (3.43) can thus be written as:

$$C_x - T_x C_1 = \sum_{l\geqslant 0} \frac{1}{\mu^{l}} \frac{2^{2l+1}}{(2l+1)!} B_{2l+1}\left(1-\frac{x}{2}\right)\frac{d^{l}}{dt^{l}} \tag{3.45}$$

Or, equivalently:

$$C_x - T_x C_1 = T_{1-x} \tag{3.46}$$

The results of proposition 3.2.1 and lemma 3.2.3 allow us to rewrite equation (3.20) in the asserted form and to state our first major theorem:

Theorem 3.2.4. *The function* $\varphi(x,t)$ *formally defined by:*

$$\varphi(x,t) = T_{1-x}\varphi_0(t) + T_x\varphi_1(t) \tag{3.47}$$

where

$$T_x = S_x(S_1)^{-1} = \sum_{l\geqslant 0} \frac{1}{\mu^l}\frac{2^{2l+1}}{(2l+1)!}B_{2l+1}\left(\frac{1+x}{2}\right)\frac{d^l}{dt^l} \tag{3.48}$$

is a solution to the heat equation:

$$\varphi_t(x,t) = \mu^2\varphi_{xx}(x,t) \tag{3.49}$$

with the boundary conditions:

$$\begin{cases} \varphi(0,t) = & \varphi_0(t) \\ \varphi(1,t) = & \varphi_1(t) \end{cases} \tag{3.50}$$

The meaning of this formal definition and, notably the convergence in the usual sense of the series is investigated in the following. As will be indicated, this definition can be extended beyond the usual sense with the help of other summation processes. We first start by verifying two results on the formal differential operator T_x:

Lemma 3.2.5. *The formal differential operator* T_x *verifies:*

$$\begin{aligned} T_0 &= 0 \\ T_1 &= 1 \end{aligned} \tag{3.51}$$

Proof. While these results seem obvious from the definition of T_x as $T_x = S_x(S_1)^{-1}$, we show that the explicit formulation of T_x given in proposition 3.2.1 also leads to this result. Indeed, we have in $x = 0$ the relation:

$$T_0 = \sum_{k\geqslant 0} \frac{1}{\mu^k}\frac{2^{2k+1}}{(2k+1)!}B_{2k+1}\left(\frac{1}{2}\right)\frac{d^k}{dt^k} \tag{3.52}$$

and we know that Bernoulli polynomials of odd order are null at 1/2:

$$B_{2n+1}\left(\frac{1}{2}\right) = 0 \tag{3.53}$$

In $x = 1$, we have equivalently:

$$T_1 = \sum_{k\geqslant 0} \frac{1}{\mu^k}\frac{2^{2k+1}}{(2k+1)!}B_{2k+1}(1)\frac{d^k}{dt^k} \tag{3.54}$$

and we know from [Abramowitz and Stegun, 1965, p. 804 23.1.2] that:

$$B_n(0) = B_n \tag{3.55}$$
$$B_n(1) = (-1)^n B_n \tag{3.56}$$
$$\tag{3.57}$$

and that, all odd Bernoulli numbers but the first are zero [Abramowitz and Stegun, 1965, p. 804 23.1.3]:

$$B_{2n+1} = \begin{cases} -\frac{1}{2} & k = 0 \\ 0 & k > 0 \end{cases} \tag{3.58}$$

Thus the announced results. □

Lemma 3.2.6. *The formal differential operator* T_x *verifies:*

$$\frac{\partial^2}{\partial x^2} T_x = \frac{1}{\mu} T_x \frac{d}{dt} \tag{3.59}$$

Proof. This result could again be considered as a consequence of the definition of T_x as $T_x = S_x(S_1)^{-1}$ since S_x verifies this property. The expression of T_x in proposition 3.2.1 is however consistent since Bernoulli polynomials have the following property [Abramowitz and Stegun, 1965, p. 804 23.1.5]:

$$B_n'(x) = nB_{n-1}, n \geqslant 1 \tag{3.60}$$

This property makes, by definition, Bernoulli polynomials an Appell sequence. Applying this property to the definition of T_x reads:

$$\frac{\partial^2}{\partial x^2} T_x = \sum_{k \geqslant 1} \frac{1}{\mu^k} \frac{2^{2k-1}}{(2k-1)!} B_{2k-1}\left(\frac{1+x}{2}\right) \frac{d^k}{dt^k} \tag{3.61}$$

Changing k for $k+1$ gives the announced result. □

Proof of theorem 3.2.4. The announced result is a direct consequence of lemmas 3.2.5 and 3.2.6. □

The result provided in theorem 3.2.4 however does not provide a proof of convergence of the series or a method to construct adapted controls. We therefore study some of these properties in the following section.

3.2.3 Computational implementation

Practical use of the T_x operator requires evaluation of Bernoulli polynomials up to a high order. Solutions to compute these numbers have been long investigated. It was the subject of one of the first computer algorithms written by Ada Lovelace for Charles Babbage Analytical Engine (to be found in [Menabrea and Lovelace, 1843, see Note G]. Since then, more efficient algorithms

have appeared. For example, the author of [Harvey, 2010] presents an algorithm that was used to compute B_k for $k = 10^8$.

In our implementation we rely on Akiyama-Tanigawa algorithm which was conjectured in [Akiyama and Tanigawa, 2001]. This algorithm, presented in algorithm 3.1, can be seen as a special version of "Pascal triangle" which was introduced to compute binomial coefficient.

```
Data: n                                                    ▷ positive integer
function AKIYAMA-TANIGAWA ALGORITHM:
    for i ← 0, n do
        a_i ← 1/(i+1)
        for j ← i, 1 do
            a_{j-1} ← j(a_{j-1} − a_j)
    return a_0                                             ▷ a_0 is B_n
```

Algorithm 3.1 – The Akiyama-Tanigawa algorithm for Bernoulli number calculation.

However, a basic implementation of this algorithm rapidly diverges due to the "low" precision of the computer usual storage for numbers: `double`. In this representation, the numbers stored occupy 64 bit of which 52 bit are used for the mantissa (the precision). This is not enough to evaluate Bernoulli numbers with Akiyama-Tanigawa past the few first numbers. For this reason, the implementation of the various evaluation presented in this chapter is performed using the GNU MPFR library. MPFR stands for Multiple Precision Floating-Point Reliably. This is a C library for arbitrary precision computation. Using this library, it is possible to specify the length of the mantissa. In the following, most of the computations are performed with a precision of 512 bit. This level of precision was chosen to be far above the minimal precision necessary for our computations. Every evaluation in this chapter was performed in less than a second. Hence, the use of this library does not represent a significant drawback on this account.

3.3 Various properties of the T_x operator

3.3.1 Polynomial states and controls

The case of polynomial functions is of particular interest to see that the operator T_x is convergent for some functions. Indeed, since the number of terms in the expression of T_x is then finite, it is ensured that the operator T_x is defined. We may thus state the following proposition:

Proposition 3.3.1. *Let $\mathbb{R}^N[t]$ be the space of polynomial of degree N with real coefficients. The application:*

$$T_x : \mathbb{R}^N[t] \longrightarrow \mathbb{R}^N[t] \times \mathbb{R}^{2N+1}[x] \tag{3.62}$$

defines an injective application.

Proof. This result is a direct consequence of the definition of T_x and the fact that its restriction to $\mathbb{R}^N[t]$ reads:

$$T_x = \sum_{k=0}^{N} \frac{1}{\mu^k} \frac{2^{2k+1}}{(2k+1)!} B_{2k+1}\left(\frac{1+x}{2}\right) \frac{d^k}{dt^k} \tag{3.63}$$

And has thus a finite number of terms. The term of highest order in x comes from $B_{2N+1}\left(\frac{1+x}{2}\right)$ which is of order $2N+1$. □

The operator T_x being defined, we want to explicitly construct solutions to some special cases. Hence, let f be a polynomial in $\mathbb{R}[x]$ such that:

$$f(x) = \sum_{l=0}^{2N-1} p_l \frac{x^l}{l!} = \sum_{l=0}^{2N-1} q_l \frac{(1-x)^l}{l!}, p_l, q_l \in \mathbb{R} \tag{3.64}$$

We want to find controls φ_0 and φ_1 such that:

$$(T_{1-x}\varphi_0 + T_x\varphi_1)(t=0) = f(x) \tag{3.65}$$

That is, we want to create a solution to the heat equation with controls on both sides so that the initial state of the solution is f. Differentiating $\varphi = T_{1-x}\varphi_0 + T_x\varphi_1$ according to x leads to:

$$\begin{aligned} \left(\frac{\partial^{2k}}{\partial x^{2k}}\varphi\right)(x=0, t=0) = f^{(2k)}(0) \\ \left(\frac{\partial^{2k}}{\partial x^{2k}}\varphi\right)(x=1, t=0) = f^{(2k)}(1) \end{aligned} \tag{3.66}$$

Using lemma 3.2.6, this leads to the equations:

$$\begin{aligned} \frac{1}{\mu^k}\left(T_{1-x}\varphi_0^{(k)} + T_x\varphi_1^{(k)}\right)(x=0, t=0) = f^{(2k)}(0) \\ \frac{1}{\mu^k}\left(T_{1-x}\varphi_0^{(k)} + T_x\varphi_1^{(k)}\right)(x=1, t=0) = f^{(2k)}(1) \end{aligned} \tag{3.67}$$

Since $T_0 = 0$ and $T_1 = 1$, this is equivalent to:

$$\begin{aligned} \varphi_0^{(k)}(0) = \mu^k p_{2k} \\ \varphi_1^{(k)}(0) = \mu^k q_{2k} \end{aligned} \tag{3.68}$$

Initially, we assume that the controls are entire functions. As such, they match their Taylor series expansion. The previous equations then uniquely define two polynomial controls:

$$\begin{aligned} \varphi_0(t) = \sum_{k=0}^{N-1} \mu^k p_{2k} \frac{t^k}{k!} \\ \varphi_1(t) = \sum_{k=0}^{N-1} \mu^k q_{2k} \frac{t^k}{k!} \end{aligned} \tag{3.69}$$

The advantage of flat systems is that the flat outputs entirely define the system. Therefore, φ is entirely defined by φ_0 and φ_1 and we have an efficient way to find a formal expression of the solution. Indeed:

$$T_x\varphi_1(t) = \sum_{k=0}^{N-1}\sum_{l=0}^{N-1-k} \frac{2^{2k+1}}{(2k+1)!} B_{2k+1}\left(\frac{1+x}{2}\right)\mu^l q_{2(k+l)}\frac{t^l}{l!} \tag{3.70}$$

Exchanging the two sums leads to:

$$T_x\varphi_1(t) = \sum_{l=0}^{N-1} \mu^l \frac{t^l}{l!} \sum_{k=0}^{N-1-l} \frac{2^{2k+1}}{(2k+1)!} B_{2k+1}\left(\frac{1+x}{2}\right) q_{2(k+l)} \tag{3.71}$$

As a direct consequence of equation (3.64), we have:

$$(-1)^k q_k = \sum_{l=0}^{2N-1-k} \frac{p_{l+k}}{l!} \tag{3.72}$$

And:

$$T_x\varphi_1(t) = \sum_{l=0}^{N-1} \mu^l \frac{t^l}{l!} \sum_{k=0}^{N-1-l} \sum_{m=2(k+l)}^{2N-1} \frac{2^{2k+1}}{(2k+1)!} B_{2k+1}\left(\frac{1+x}{2}\right) \frac{p_m}{(m-2(k+l))!} \tag{3.73}$$

After inverting the two inner sums, the equation reads:

$$T_x\varphi_1(t) = \sum_{l=0}^{N-1} \mu^l \frac{t^l}{l!} \sum_{m=2l}^{2N-1} \frac{p_m 2^{m+1-2l}}{(m+1-2l)!} \sum_{k=0}^{\frac{m}{2}-l} \binom{m+1-2l}{2k+1} \frac{1}{2^{m-2(k+l)}} B_{2k+1}\left(\frac{1+x}{2}\right) \tag{3.74}$$

The inner sum can be identified – as was done in the previous section – to as the odd part of the Bernoulli polynomial $B_{m+1-2l}\left(1+\frac{x}{2}\right)$ and thus:

$$T_x\varphi_1(t) = \sum_{l=0}^{N-1} \mu^l \frac{t^l}{l!} \sum_{m=0}^{2N-2l-1} \frac{p_{m+2l} 2^{m+1}}{(m+1)!} \frac{1}{2}\left(B_{m+1}\left(1+\frac{x}{2}\right) - B_{m+1}\left(1-\frac{x}{2}\right)\right) \tag{3.75}$$

And using the fact that ([Abramowitz and Stegun, 1965, p. 804 23.1.6 and 23.1.8]):

$$\begin{aligned} B_{m+1}\left(1+\frac{x}{2}\right) &= B_{m+1}\left(\frac{x}{2}\right) + (m+1)\left(\frac{x}{2}\right)^m \\ B_{m+1}\left(1-\frac{x}{2}\right) &= (-1)^{m+1} B_{m+1}\left(\frac{x}{2}\right) \end{aligned} \tag{3.76}$$

And dividing the sum according to the parity of m, we have:

$$T_x\varphi_1(t) = \sum_{l=0}^{N-1} \mu^l \frac{t^l}{l!} \sum_{m=0}^{2N-1-2l} p_{m+2l} \frac{x^m}{m!} + \sum_{l=0}^{N-1} \mu^l \frac{t^l}{l!} \sum_{m=0}^{N-l-1} \frac{p_{2m+2l}}{(2m+1)!} 2^{2m+1} B_{2m+1}\left(\frac{x}{2}\right) \tag{3.77}$$

Where we recognize the second term in terms of derivatives of f and the second as $-T_{1-x}\varphi_0(t)$ (since $B_{2m+1}\left(\frac{x}{2}\right) = -B_{2m+1}\left(1-\frac{x}{2}\right)$), and thus:

$$T_{1-x}\varphi_0(t) + T_x\varphi_1(t) = \sum_{l=0}^{N-1} \mu^l \frac{t^l}{l!} f^{(2l)}(x) \tag{3.78}$$

This relation allows us to state our next result:

Theorem 3.3.2. *Given a function* f *of the form:*

$$f(x) = \sum_{k=0}^{2N-1} p_k \frac{x^k}{k!} = \sum_{k=0}^{2N-1} q_k \frac{(1-x)^k}{k!} \tag{3.79}$$

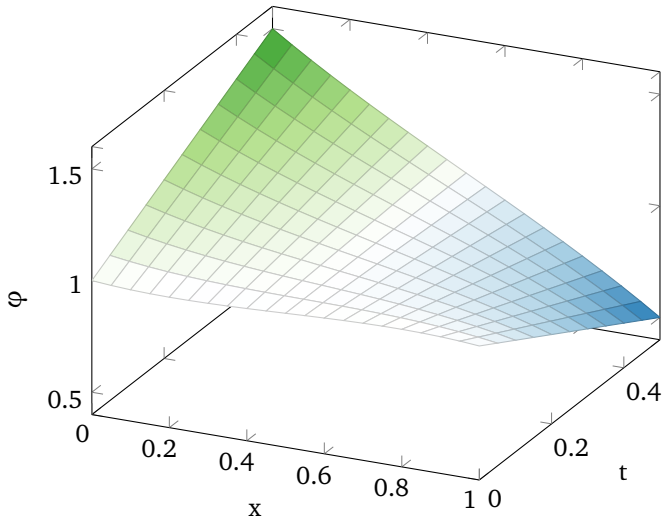


Figure 3.2 – Evolution of a polynomial state ($f(x) = 1 - \frac{1}{6}x + x^2 - 2x^3$) where the polynomial controls ($\varphi_0(t) = 1 + \mu t$, $\varphi_0(t) = 1 - \mu t$) have been built after theorem 3.3.2 ($\mu = 1.0$), 3d view.

The two controls defined by:

$$\varphi_0(t) = \sum_{k=0}^{N-1} p_{2k}\mu^k \frac{t^k}{k!} \tag{3.80}$$

And:

$$\varphi_1(t) = \sum_{k=0}^{N-1} q_{2k}\mu^k \frac{t^k}{k!} \tag{3.81}$$

Define a solution:

$$T_{1-x}\varphi_0(t) + T_x\varphi_1(t) = \varphi(x,t) \tag{3.82}$$

Which verifies:

$$\varphi(x,0) = f(x) \tag{3.83}$$

And which can be expressed as:

$$\varphi(x,t) = \sum_{l=0}^{N-1} \mu^l \frac{t^l}{l!} f^{(2l)}(x) \tag{3.84}$$

An illustration of this solution, evaluated based on equation (3.82), is given in figures 3.2 and 3.3. On low order polynomials the difference between solutions evaluated based on equation (3.82) and equation (3.84) is computationally insignificant. Indeed, only the very first Bernoulli numbers have to be evaluated, this is done without any problems. The computational efficiency and precision of equation (3.82) in term of convergence speed will be discussed in section 3.3.4.2. Since non constant polynomials are diverging, the controls, and thus the solution to the heat equation created by this solution are bound to diverge with time unless the initial state is linear (in which case the controls are constant).

This theorem can also be used to construct a solution having f as final state at $t = 1$ as stated in the following corollary:

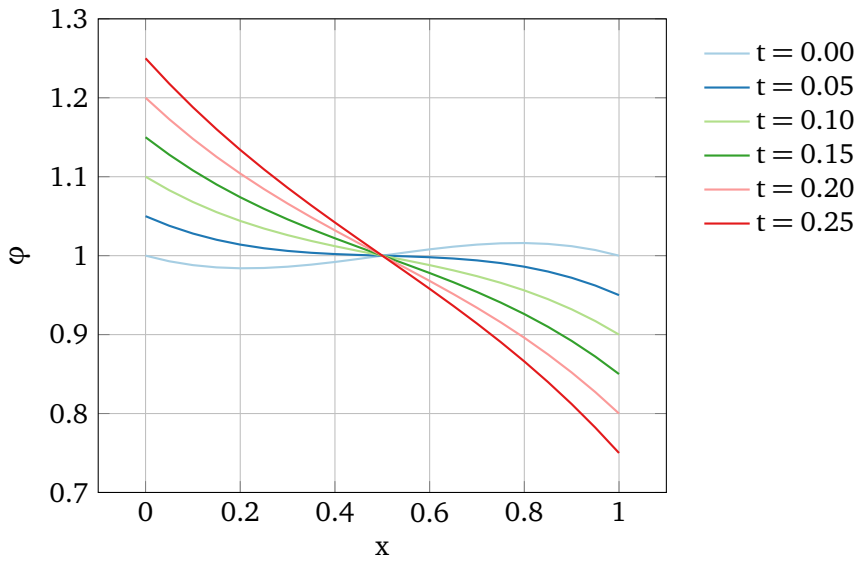


Figure 3.3 – Evolution of a polynomial state ($f(x) = 1 - \frac{1}{6}x + x^2 - 2x^3$) where the polynomial controls ($\varphi_0(t) = 1 + \mu t$, $\varphi_0(t) = 1 + \mu t$) have been built after theorem 3.3.2 ($\mu = 1.0$), cross-sectional view.

Corollary 3.3.3. *Given the same function f, the two controls defined by:*

$$\varphi_0(t) = \sum_{k=0}^{N-1} p_{2k}\mu^k \frac{(t-1)^k}{k!} \tag{3.85}$$

And:

$$\varphi_1(t) = \sum_{k=0}^{N-1} q_{2k}\mu^k \frac{(t-1)^k}{k!} \tag{3.86}$$

Define a solution:

$$T_{1-x}\varphi_0(t) + T_x\varphi_1(t) = \varphi(x,t) \tag{3.87}$$

Which verifies:

$$\varphi(x,1) = f(x) \tag{3.88}$$

And which can be expressed as:

$$\varphi(x,t) = \sum_{l=0}^{N-1} \mu^l \frac{(t-1)^l}{l!} f^{(2l)}(x) \tag{3.89}$$

Proof. The proof is similar to the proof of theorem 3.3.2. □

3.3.2 Application to the Weierstrass approximation theorem

The preceding result allows us to give a generalization of our result. Indeed, it is known from:

Theorem 3.3.4. *Weierstrass approximation theorem, (see e.g. [Rudin, 1976, p. 159 theorem 7.26])*
If f *is a continuous real function on* $[a, b]$*, there exists a sequence of real polynomials* P_n *such that*

$$\lim_{n\to\infty} P_n(x) = f(x) \tag{3.90}$$

uniformly on $[a, b]$.

Moreover, there exist explicit methods to constructs these polynomials such as, for example, Bernstein polynomials. This allows us to state another result:

Proposition 3.3.5. *Let* $f(x)$ *be a continuous real function over* $[0, 1]$*. It is possible to construct two sequences of real functions* $\varphi_{0,n}(t)$ *and* $\varphi_{1,n}(t)$ *such that:*

$$\forall \varepsilon > 0, \exists N \geqslant 0, \forall n \geqslant N, \sup_{x\in[0,1]} |f(x) - \varphi_n(x, 0)| < \varepsilon \tag{3.91}$$

Where

$$\varphi_n(x, t) = T_{1-x}\varphi_{0,n}(t) + T_x\varphi_{1,n}(t) \tag{3.92}$$

is a solution to the heat equation with

$$\varphi_n(0, t) = \varphi_{0,n}(t) \tag{3.93}$$
$$\varphi_n(1, t) = \varphi_{1,n}(t) \tag{3.94}$$
$$\tag{3.95}$$

Proof. Through Weierstrass approximation theorem we know that there exists a sequence of polynomials f_n such as:

$$\forall \varepsilon > 0, \exists N \geqslant 0, \forall n \geqslant N, \sup_{x\in[0,1]} |f(x) - f_n(x, 0)| < \varepsilon \tag{3.96}$$

Such a sequence can be constructed, for example using Bernstein polynomials. From theorem 3.3.2, we can construct two sequences of functions $\varphi_{0,n}$, $\varphi_{1,n}$ from f_n that give the wanted result. □

This result, however, does not ensure that the $\varphi_{0,n}$ and $\varphi_{1,n}$ functions converge in the usual sense nor that the solution $\varphi_n(x, t)$ does. Therefore we will study in the next subsection the problem of convergence of T_x for arbitrary functions.

3.3.3 Convergence of the T_x operator

As can be seen from figure 3.1, the maxima of Bernoulli polynomials increase rapidly. The convergence of the T_x polynomials, is established for polynomial functions because they require a finite number of Bernoulli polynomials. This convergence has to be studied for other classes of functions. A study of the absolute convergence of the operator and properties on the maxima of Bernoulli polynomials on the unity axis allow us to state the following result.

Proposition 3.3.6. *Let* $a, b > 0$, $R < \mu\pi^2$ *be three strictly positive coefficients and* $\varphi_0(x)$, $\varphi_1(x)$ *two functions over* $[0,1]$ *with the following property:*

$$\sup_{t\in[0,1]} |\varphi_0^{(k)}(t)| \leqslant aR^k \tag{3.97}$$

$$\sup_{t\in[0,1]} |\varphi_1^{(k)}(t)| \leqslant bR^k \tag{3.98}$$

Then, the solution to the heat equation defined in equation (3.47) is convergent. Furthermore, the resulting solution to the heat equation is Gevrey of order 0 *in both* x *and* t.

Proof. To prove this assertion, we search for an upper bound for the general term of $\varphi(x,t)$:

$$|c_k(x,t)| = \frac{1}{\mu^k}\frac{2^{2k+1}}{(2k+1)!}\left|B_{2k+1}\left(\frac{1+x}{2}\right)\varphi_1^{(k)}(t) + B_{2k+1}\left(1-\frac{x}{2}\right)\varphi_0^{(k)}(t)\right| \tag{3.99}$$

$$\leqslant \frac{1}{\mu^k}\frac{2^{2k+1}}{(2k+1)!}\left(\left|B_{2k+1}\left(\frac{1+x}{2}\right)\varphi_1^{(k)}(t)\right| + \left|B_{2k+1}\left(1-\frac{x}{2}\right)\varphi_0^{(k)}(t)\right|\right) \tag{3.100}$$

$$\tag{3.101}$$

[Lehmer, 1940] determines that for $x \in [0,1]$, Bernoulli polynomials of odd order are uniformly bounded by:

$$\sup_{x\in[0,1]} |B_{2k+1}(x)| < \frac{2(2k+1)!}{(2\pi)^{2k+1}}, \quad k \geqslant 0 \tag{3.102}$$

This allows us to rewrite the bound as:

$$|c_k(x,t)| \leqslant \frac{1}{\mu^k}\frac{2}{\pi^{2k+1}}\left(\left|\varphi_1^{(k)}(t)\right| + \left|\varphi_0^{(k)}(t)\right|\right) \tag{3.103}$$

$$\tag{3.104}$$

Using the bound of the derivatives given in equations (3.97) and (3.98), we can bound the general term by:

$$|c_k(x,t)| \leqslant \frac{2}{\pi}\frac{R^k}{\mu^k\pi^{2k}}(a+b) \tag{3.105}$$

$$\tag{3.106}$$

Applying Cauchy's criterion test to this general term gives:

$$\sqrt[k]{|c_k(x,t)|} \leqslant \sqrt[k]{\frac{2(a+b)}{\pi}}\frac{R}{\mu\pi^2} \tag{3.107}$$

$$\tag{3.108}$$

And thus:

$$\limsup_{k\to\infty} \sqrt[k]{|c_k(x,t)|} \leqslant \frac{R}{\mu\pi^2} < 1 \tag{3.109}$$

$$\tag{3.110}$$

which gives the announced convergence. To show that this solution is Gevrey, we derive the general term of $T_0\varphi_1(t)$ for any given order n and m:

$$\frac{\partial^{n+m}}{\partial^n x\,\partial^m t}c_k(x,t) = \frac{1}{\mu^k}\frac{2^{2k+1}}{(2k+1)!}\left(\frac{\partial^n}{\partial^n x}B_{2k+1}\left(\frac{1+x}{2}\right)\right)\varphi_1^{(k+m)}(t) \tag{3.111}$$

$$= \frac{1}{\mu^k}\frac{2^{2k+1-n}}{(2k+1-n)!}B_{2k+1-n}\left(\frac{1+x}{2}\right)\varphi_1^{(k+m)}(t) \tag{3.112}$$

where we used the fact that the Bernoulli polynomials form an Appell sequence. For any order, we bound the Bernoulli polynomials by the following:

$$\sup_{x\in[0,1]}|B_k(x)| < \frac{2k!}{(2\pi)^k}\zeta(2), \quad k \geqslant 0 \tag{3.113}$$

where ζ is the Riemann zeta function. We may then bound the general term by:

$$\left|\frac{\partial^{n+m}}{\partial^n x\,\partial^m t}c_k(x,t)\right| \leqslant \frac{1}{\mu^k}\frac{2}{\pi^{2k+1-n}}\zeta(2)bR^{k+m} \tag{3.114}$$

□

Cauchy's criterion apply as well and there exists a constant M such that:

$$\left|\frac{\partial^{n+m}}{\partial^n x\,\partial^m t}T_x\varphi_1(t)\right| \leqslant M\pi^n R^m \tag{3.115}$$

A similar proof can be given for the second term. The sum of these two terms is also Gevrey of the same order, this finishes the proof.

Remark. The given bound on φ_0 and φ_1 is equivalent to say that this functions are Gevrey of order 0 with radius $R < 1/(\mu\pi^2)$. The proof developed for proposition 3.3.6 however does not seem to extend well to Gevrey functions of strictly positive order.

3.3.4 Application of T_x to exponential functions

3.3.4.1 Convergent case

An example of non-polynomial function respecting the conditions of proposition 3.3.6 is $\varphi(t) = \exp(\lambda t)$ with $|\lambda| < \mu\pi^2$. Consider two linear combinations:

$$\varphi_0(t) = \sum_{i=0}^{n} a_i e^{\lambda_i t} \tag{3.116}$$

$$\varphi_1(t) = \sum_{i=0}^{n} b_i e^{\gamma_i t} \tag{3.117}$$

With $|\lambda_i|, |\gamma_i| < \mu\pi^2$. The heat equation being a linear equation, we can consider a single term (thus the index i will be omitted) of this linear combinations. On the one hand, we have:

$$T_x\varphi_1(t) = be^{\gamma t}\sum_{k\geqslant 0}\frac{\gamma^k}{\mu^k}\frac{2^{2k+1}}{(2k+1)!}B_{2k+1}\left(\frac{1+x}{2}\right) \tag{3.118}$$

Expanding the Bernoulli polynomial around 1/2, one gets:

$$T_x\varphi_1(t) = be^{\gamma t}\sum_{k\geqslant 0}\frac{\gamma^k}{\mu^k}\frac{2^{2k+1}}{(2k+1)!}\sum_{l=0}^{2k+1}\binom{2k+1}{l}B_l\left(\frac{1}{2}\right)\left(\frac{x}{2}\right)^{2k+1-l} \tag{3.119}$$

However we know that([Abramowitz and Stegun, 1965, p. 805 23.1.21]):

$$B_n\left(\frac{1}{2}\right) = -(1-2^{1-n})B_n, \quad n\geqslant 0 \tag{3.120}$$

Which implies that:

$$B_{2n+1}\left(\frac{1}{2}\right) = 0, \quad n\geqslant 0 \tag{3.121}$$

And thus:

$$T_x\varphi_1(t) = -be^{\gamma t}\sum_{k\geqslant 0}\frac{\gamma^k}{\mu^k}\sum_{l=0}^{k}\frac{(2^{2l}-2)B_{2l}}{(2(k-l)+1)!(2l)!}x^{2k+1-2l} \tag{3.122}$$

Inverting the two sums leads to:

$$T_x\varphi_1(t) = -be^{\gamma t}\sum_{l\geqslant 0}\frac{\gamma^l}{\mu^l}\frac{(2^{2l}-2)B_{2l}}{(2l)!}\sum_{k\geqslant 0}\frac{\gamma^k}{\mu^k}\frac{x^{2k+1}}{(2k+1)!} \tag{3.123}$$

Where we recognize the series expansion of the hyperbolic sine and cosine sinh and csch which is convergent for $|\sqrt{\gamma/\mu}| < \pi$. We may therefore write the previous equation as:

$$T_x\varphi_1(t) = -be^{\gamma t}\operatorname{csch}\left(\sqrt{\frac{\gamma}{\mu}}\right)\sinh\left(\sqrt{\frac{\gamma}{\mu}}x\right) \tag{3.124}$$

Similarly, we have:

$$T_{1-x}\varphi_0(t) = -ae^{\lambda t}\operatorname{csch}\left(\sqrt{\frac{\lambda}{\mu}}\right)\sinh\left(\sqrt{\frac{\lambda}{\mu}}(1-x)\right) \tag{3.125}$$

And then:

$$\varphi(x,t) = be^{\gamma t}\sinh\left(\sqrt{\frac{\gamma}{\mu}}x\right)\operatorname{csch}\left(\sqrt{\frac{\gamma}{\mu}}\right) - ae^{\lambda t}\sinh\left(\sqrt{\frac{\lambda}{\mu}}(1-x)\right)\operatorname{csch}\left(\sqrt{\frac{\lambda}{\mu}}\right) \tag{3.126}$$

If we set $\gamma = \lambda$ and $c = -a = b\exp-\sqrt{\gamma/\mu}$, we find the following obvious solution to the heat equation:

$$\varphi(x,t) = c\exp\left(\gamma t + \sqrt{\frac{\gamma}{\mu}}x\right) \tag{3.127}$$

$$\varphi_0(t) = c\exp\left(\gamma t\right) \tag{3.128}$$

$$\varphi_1(t) = c\exp\left(\gamma t + \sqrt{\frac{\gamma}{\mu}}\right) \tag{3.129}$$

Furthermore, when changing γ for $-\gamma$ ($\gamma > 0$), the hyperbolic functions transform in their circular equivalents and leads to the two following solutions to the heat equation, either

$$\begin{cases} \varphi(x,t) & = ce^{-\gamma t}\cos\left(\sqrt{\frac{\gamma}{\mu}}x\right) \\ \varphi_0(t) & = ce^{-\gamma t} \\ \varphi_1(t) & = ce^{-\gamma t}\cos\left(\sqrt{\frac{\gamma}{\mu}}\right) \end{cases} \tag{3.130}$$

Or:

$$\begin{cases} \varphi(x,t) & = ce^{-\gamma t}\sin\left(\sqrt{\frac{\gamma}{\mu}}x\right) \\ \varphi_0(t) & = 0 \\ \varphi_1(t) & = ce^{-\gamma t}\sin\left(\sqrt{\frac{\gamma}{\mu}}\right) \end{cases} \tag{3.131}$$

The condition $\sqrt{\gamma/\mu} < \pi$ required for the resulting $T_x\varphi_0$ function to be convergent implies that the unity axis can contain only less than half a period of the cosine (resp. sine) function. It is therefore of interest to study the possibility to use greater values of γ.

3.3.4.2 Numerical study of the convergence

In the previously presented solutions to the heat equation, it seems reasonable to consider values of γ such that $|\gamma| \geqslant \mu\pi^2$. Indeed, the corresponding solutions do exist ! To understand the problem of divergence of the operator T_x, we define the general term:

$$a_k(x) = \frac{1}{\mu^k}\frac{2^{2k+1}}{(2k+1)!}B_{2k+1}\left(\frac{1+x}{2}\right) \tag{3.132}$$

The functions a_k are plotted in figure 3.4 for the first values of k. It confirms the geometric decrease of the function. We introduce the function $t_\gamma(x)$ defined as:

$$t_\gamma : x \mapsto \sum_{k\geqslant 0}\gamma^k a_k(x), |\gamma| < \mu\pi^2 \tag{3.133}$$

We may interpret $t_\gamma(x)$ as:

$$t_\gamma : x \mapsto \lim_{t\to 0}\left(\exp^{-\gamma t} T_x \exp^{\gamma t}\right) \tag{3.134}$$

In the case of positive γ, it can be seen that the series defining the function $t_\gamma(x)$ is an alternating series. Indeed, [Abramowitz and Stegun, 1965, p. 805 23.1.14] states that $(-1)^{k+1}B_{2k+1}(x) > 0, n \geqslant 1, x \in (0, \frac{1}{2})$. Together with [Abramowitz and Stegun, 1965, p. 804 23.1.8] which gives $B_{2k+1}\left(\frac{1+x}{2}\right) = -B_{2k+1}\left(\frac{1-x}{2}\right)$, we may stated that $B_{2k+1}\left(\frac{1+x}{2}\right)$ is of the sign of $(-1)^k$. Then an approximation of the convergence of the series is given by:

$$|t_\gamma(x) - \sum_{k=0}^{n-1}\gamma^k a_k(x)| \leqslant \gamma^n |a_n(x)| \tag{3.135}$$

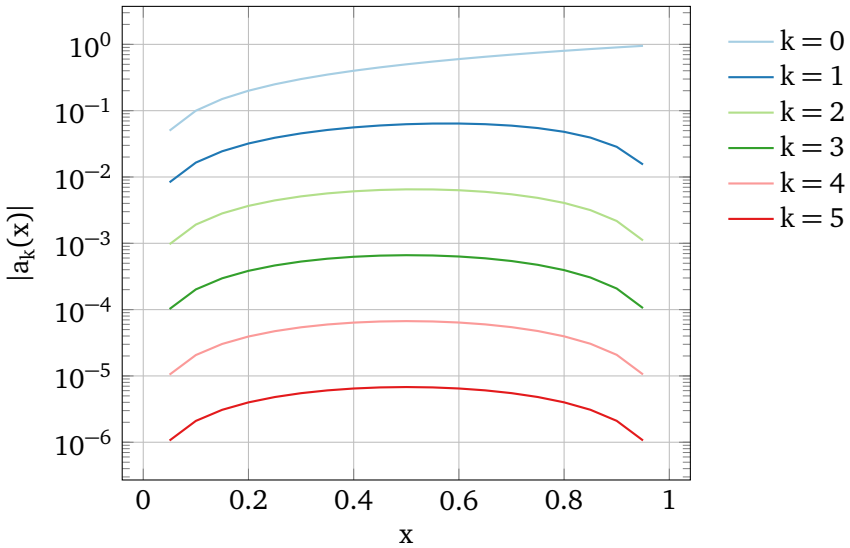


Figure 3.4 – Numerical evaluation of the first a_k functions ($\mu = 1$).

which gives us the following approximate upper bound of the absolute error ([Abramowitz and Stegun, 1965, p. 805 23.1.14])

$$|t_\gamma(x) - \sum_{k=0}^{n-1} \gamma^k a_k(x)| \leqslant \frac{2}{\pi}\left(\frac{\gamma}{\mu\pi^2}\right)^n \left(\frac{1}{1-2^{-2n}}\right) \tag{3.136}$$

Hence, the convergence of the series is in this case geometric. To get an approximate value of t_γ with an error of at least ε, we have to estimate at most n_ε terms where n_ε verifies:

$$n_\varepsilon \geqslant \frac{\ln \varepsilon\pi - \ln 2}{\ln \frac{\gamma}{\mu\pi^2}} \tag{3.137}$$

The evolution of the error is represented in figure 3.5. As was foreseen from the error evaluation, the closer γ is to π^2, the slower the convergence is. With γ really close to its upper bound it even becomes computationally unpractical to get an appropriate evaluation of $t_\gamma(x)$.

For negative values of γ, we bound the terms by the geometric approximation $\frac{2}{\pi}\left(\frac{\gamma}{\mu\pi^2}\right)^n$. This gives the error bound:

$$|t_\gamma(x) - \sum_{k=0}^{n-1} \gamma^k a_k(x)| \leqslant \frac{2}{\pi}\left(\frac{|\gamma|}{\mu\pi^2}\right)^n \left(\frac{1}{1-\frac{|\gamma|}{\mu\pi^2}}\right) \tag{3.138}$$

The decrease of this upper bound of the error appears to be somewhat slower – by a factor $1/(1-\frac{|\gamma|}{\mu\pi^2})$ – than in the positive case. This decrease is nevertheless in the same range. The function $t_\gamma(x)$ is then evaluated for $n = 40$ for various positive and negative values of γ and various values of the viscosity μ. The results are represented in figure 3.6. The functions are shared according to the sign of γ on both side of the $t_0(x) = x$ function. Functions with negative γ are above t_0 and are concave while function with positive γ are below t_0 and are convex.

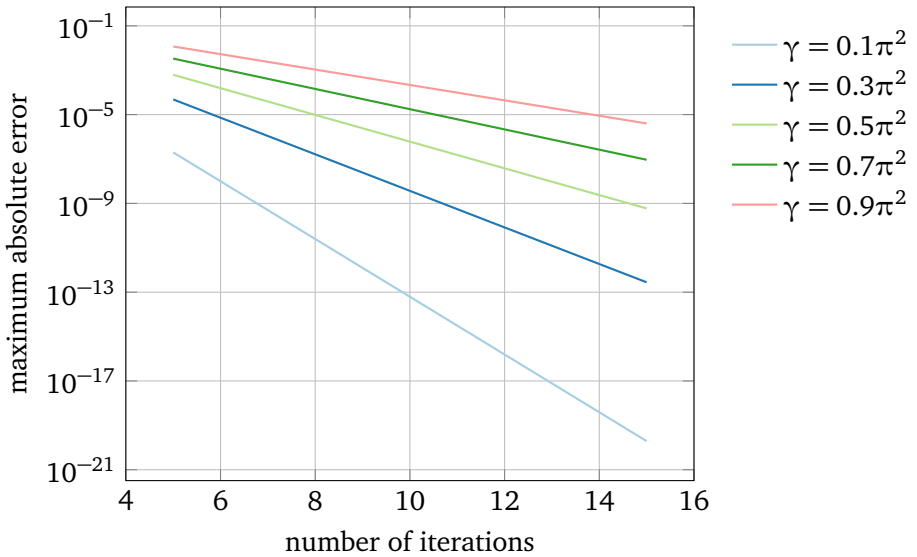


Figure 3.5 – Numerical evaluation of the error ε according to the number of terms calculated for chosen values of γ ($\mu = 1$).

With increasing viscosity – which corresponds by a change of variable to a decrease in γ – the t_γ functions get closer to the identity. However, usual summation for values of $\gamma/\mu > \pi^2$ diverges. For such values, other summation processes should be used [Malgrange and Ramis, 1992; Lutz et al., 1999; Meurer, 2005]. However, such studies are outside of the scope of the present work.

Remark. [Nörlund, 1922] proved that the (only) zero of $B_{2n}(x)$ between 0 and $\frac{1}{2}$ (more precisely, the only zero of $B_{2n}(x)$ between 0 and $\frac{1}{4}$ which is the only lying between 0 and $\frac{1}{2}$) tends to $\frac{1}{4}$ with n. [Ostrowski, 1960] improves this result by showing that this zero convergences monotonically. By symmetry, the zero of $B_{2n}(x)$ on the other half of the unit axis tends to $\frac{3}{4}$. As a consequence, recalling that $B'_{2n+1}(x) = (2n+1)B_{2n}(x)$, the position of the maximum of $B_{2n+1}\left(\frac{1+x}{2}\right)$ tends to $\frac{1}{2}$.

3.3.5 The operator T_x and product of functions

3.3.5.1 Product with exponential functions

Based on sections 3.3.1 and 3.3.4, our first goal is to create, for any real continuous function $f(x)$ over $[0,1]$ two sequences of real functions $\varphi_{0,n}(t)$ and $\varphi_{1,n}$ driving the heat equation from state $f(x)$ at $t = 0$ to zero in infinite time. For this purpose, we first show the following result:

Proposition 3.3.7. *Consider the following function:*

$$\varphi_{1,n}(t) = \frac{t^n}{n!}e^{-\gamma t}, n > 0, \mu\pi^2 > \gamma > 0 \tag{3.139}$$

Then, the function:

$$(x,t) \in [0,1] \times \mathbb{R}^+ \mapsto T_x\varphi_{1,n}(t) \tag{3.140}$$

is defined and converges to zero:

$$\lim_{t\to\infty} T_x\varphi_{1,n}(x,t) = 0 \tag{3.141}$$

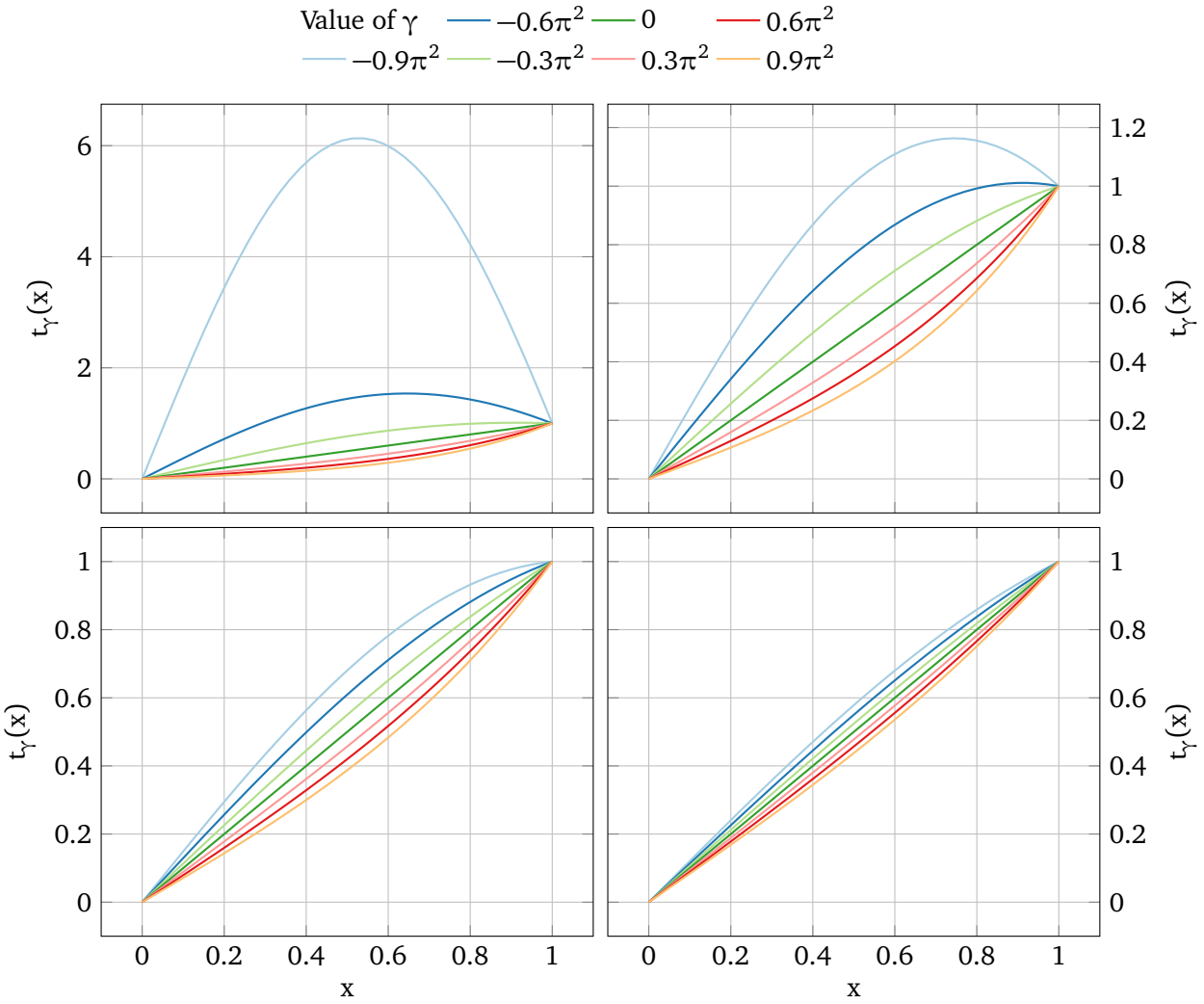


Figure 3.6 – Numerical evaluation of $t_\gamma(x)$ for various values of γ and various viscosities (from top to bottom, from left to right respectively $\mu = 1.0$, $\mu = 2.0$, $\mu = 4.0$, $\mu = 8.0$).

Proof. Applying the formal differential operator of infinite order T_x to this function reads:

$$T_x\varphi_{1,n}(t) = \sum_{k\geqslant 0} \frac{1}{\mu^k} \frac{2^{2k+1}}{(2k+1)!} B_{2k+1}\left(\frac{1+x}{2}\right) \frac{d^k}{dt^k}\left(\frac{t^n}{n!}e^{-\gamma t}\right) \tag{3.142}$$

Applying the formula for the derivative of products reads:

$$\begin{aligned} T_x\varphi_{1,n}(x,t) = &\sum_{k=0}^{n} \frac{1}{\mu^k} \frac{2^{2k+1}}{(2k+1)!} B_{2k+1}\left(\frac{1+x}{2}\right) \sum_{l=0}^{k} \binom{k}{l} \frac{t^{n-l}}{(n-l)!} (-\gamma)^{k-l} e^{-\gamma t} \\ &+ \sum_{k>n} \frac{1}{\mu^k} \frac{2^{2k+1}}{(2k+1)!} B_{2k+1}\left(\frac{1+x}{2}\right) \sum_{l=0}^{n} \binom{k}{l} \frac{t^{n-l}}{(n-l)!} (-\gamma)^{k-l} e^{-\gamma t} \end{aligned} \tag{3.143}$$

Inverting the sums leads to:

$$T_x\varphi_{1,n}(x,t) = e^{-\gamma t} \sum_{l=0}^{n} \frac{t^{n-l}}{(n-l)!} \sum_{k\geqslant l} \frac{(-\gamma)^{k-l}}{\mu^k} \frac{2^{2k+1}}{(2k+1)!} B_{2k+1}\left(\frac{1+x}{2}\right) \binom{k}{l} \tag{3.144}$$

To bound the general term

$$d_{k,l}(x) = \frac{(-\gamma)^{k-l}}{\mu^k} \frac{2^{2k+1}}{(2k+1)!} B_{2k+1}\left(\frac{1+x}{2}\right) \binom{k}{l} \tag{3.145}$$

of the inner sum, we use the bound introduced in equation (3.102) and the following bound for the binomial product:

$$\binom{k}{l} \leqslant \frac{k^l}{l!} \tag{3.146}$$

Then the general term $d_{k,l}(x)$ is bounded by:

$$|d_{k,l}(x)| \leqslant \frac{\gamma^k}{\mu^k} \frac{1}{(\pi)^{2k}} \frac{k^l}{l!\gamma^l} \tag{3.147}$$

Applying Cauchy's criterion test to this general term gives:

$$\sqrt[k]{|d_{k,l}(x)|} \leqslant \frac{\gamma}{\mu\pi^2} \sqrt[k]{\frac{k^l}{l!\gamma^l}} \tag{3.148}$$

And thus:

$$\limsup_{k\to\infty} \sqrt[k]{|d_{k,l}(x)|} \leqslant \frac{\gamma}{\mu\pi^2} < 1 \tag{3.149}$$

Then the inner sum $\sum_{k\geqslant l} d_{k,l}(x)$ converges. We call this limit $d_l(x)$. We have then the following expression:

$$T_x\varphi_{1,n}(x,t) = e^{-\gamma t} \sum_{l=0}^{n} \frac{t^{n-l}}{(n-l)!} d_l(x) \tag{3.150}$$

This expression is clearly defined for all n and converges to zero (γ is chosen strictly positive) with respect to time. This gives the announced result. □

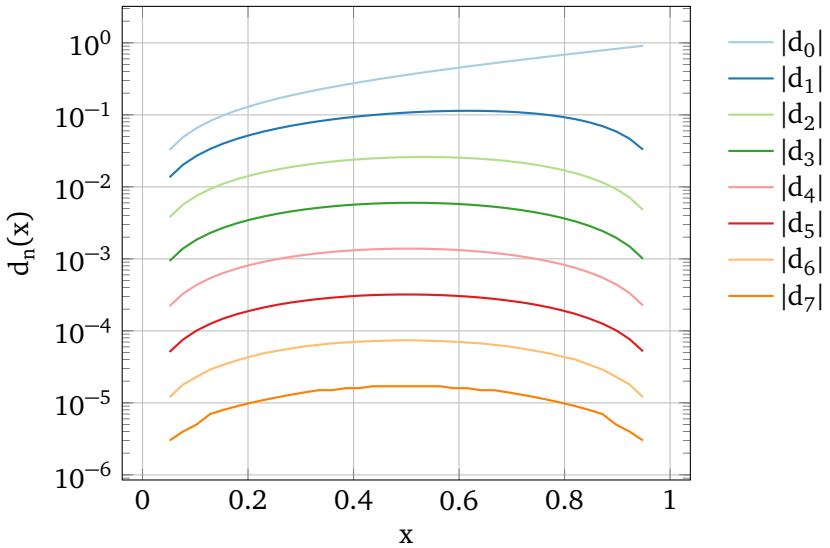


Figure 3.7 – The first $d_n(x)$ functions ($\gamma = 1.0$, $\mu = 1.0$).

At time $t = 0$, we have:

$$T_x\varphi_{1,n}(x,0) = d_n(x) \tag{3.151}$$

By definition, $d_0(x) = t_\gamma(x)$. The function is depicted in figure 3.7. Despite their similar shape, the function d_n are not directly linked to the general term a_n depicted in figure 3.4, d_6 is of the order of a_4. In future works, it could be interesting to prove that the successive d_n functions are linearly independent. This would allow to construct approximating polynomials matching any starting state. This is not the chosen solution since [Laroche et al., 2000] suggest a solution based on the function introduced in equation (2.30) allowing finite-time transitions. We present this method in the following.

3.3.5.2 Product with the non-analytic Φ_σ function

Using non-analytic functions such as the Gevrey functions introduced in section 2.2.1 allows the transition to be performed in finite time. Therefore we present the following result:

Proposition 3.3.8. *Consider the following function:*

$$\varphi_{1,n}(t) = \frac{t^n}{n!}(1 - \Phi_\sigma(t)), n > 0 \tag{3.152}$$

where Φ_σ is the Gevrey function defined in equation (2.30) on page 48. Then, assuming this definition has a meaning, the function, :

$$(x,t) \in [0,1] \times \mathbb{R}^+ \mapsto T_x\varphi_{1,n}(t) \tag{3.153}$$

verifies:

$$\begin{aligned} T_x\varphi_{1,n}(x,0) &= T_x\frac{t^n}{n!}(x,1) \\ T_x\varphi_{1,n}(x,1) &= 0 \end{aligned} \tag{3.154}$$

Proof. The function $\Phi_\sigma(t)$ is differentially flat in $t = 0$ and $t = 1$. In these points, the derivatives of $\varphi_{1,n}(t)$ verify:

$$\lim_{t\to 0} \frac{d^k}{dt^k}\varphi_{1,n}(t) = \frac{d^k}{dt^k}\frac{t^n}{n!} \tag{3.155}$$

And:

$$\lim_{t\to 1} \frac{d^k}{dt^k}\varphi_{1,n}(t) = 1 \tag{3.156}$$

□

However, it is not ensured that the function $T_x\varphi_{1,n}$ exists over since the convergence of the series can not be established by simple analytic means. Numerical simulations tend to accredit the hypotheses that these series, taken in the usual sense, are divergent in non-trivial cases. In a similar case, [Laroche et al., 2000] suggest to use a least-term summation. In the following, we assume that a suitable summation scheme exist that give a meaning to the operator T_x. We can then give a generalization of theorem 3.3.2:

Theorem 3.3.9. *Given a function* f *of the form:*

$$f(x) = \sum_{k=0}^{2N-1} p_k \frac{x^k}{k!} = \sum_{k=0}^{2N-1} q_k \frac{(1-x)^k}{k!} \tag{3.157}$$

The two controls defined by:

$$\varphi_0(t) = (1-\Phi_\sigma(t)) \sum_{k=0}^{N-1} p_{2k}\mu^k \frac{t^k}{k!} \tag{3.158}$$

And:

$$\varphi_1(t) = (1-\Phi_\sigma(t)) \sum_{k=0}^{N-1} q_{2k}\mu^k \frac{t^k}{k!} \tag{3.159}$$

Define a solution:

$$T_{1-x}\varphi_0(t) + T_x\varphi_1(t) = \varphi(x,t) \tag{3.160}$$

Which verifies:

$$\begin{cases} \varphi(x,0) & = f(x) \\ \varphi(x,1) & = 0 \end{cases} \tag{3.161}$$

Proof. This result is a direct consequence of theorem 3.3.2 and proposition 3.3.8. □

Following corollary 3.3.3, we can state this useful corollary:

Corollary 3.3.10. *Given the same function* f*, the two controls defined by:*

$$\varphi_0(t) = \Phi_\sigma(t) \sum_{k=0}^{N-1} p_{2k}\mu^k \frac{(t-1)^k}{k!} \tag{3.162}$$

And:

$$\varphi_1(t) = \Phi_\sigma(t) \sum_{k=0}^{N-1} q_{2k} \mu^k \frac{(t-1)^k}{k!} \tag{3.163}$$

Define a solution:

$$T_{1-x}\varphi_0(t) + T_x\varphi_1(t) = \varphi(x,t) \tag{3.164}$$

Which verifies:

$$\begin{cases} \varphi(x,0) & = 0 \\ \varphi(x,1) & = f(x) \end{cases} \tag{3.165}$$

The formal differential operator of infinite order T_x is linear. This enables us to state this final corollary which will be the base of the functions used in the frame of our motion-planning framework:

Corollary 3.3.11. *Given two function* f_0 *and* f_1 *of the form:*

$$\begin{aligned} f_0(x) &= \sum_{k=0}^{2N-1} p_k \frac{x^k}{k!} = \sum_{k=0}^{2N-1} q_k \frac{(1-x)^k}{k!} \\ f_1(x) &= \sum_{k=0}^{2N-1} r_k \frac{x^k}{k!} = \sum_{k=0}^{2N-1} s_k \frac{(1-x)^k}{k!} \end{aligned} \tag{3.166}$$

The two controls defined by:

$$\varphi_0(t) = (1-\Phi_\sigma(t)) \sum_{k=0}^{N-1} p_{2k} \mu^k \frac{t^k}{k!} + \Phi_\sigma(t) \sum_{k=0}^{N-1} r_{2k} \mu^k \frac{(t-1)^k}{k!} \tag{3.167}$$

And:

$$\varphi_1(t) = (1-\Phi_\sigma(t)) \sum_{k=0}^{N-1} q_{2k} \mu^k \frac{t^k}{k!} + \Phi_\sigma(t) \sum_{k=0}^{N-1} s_{2k} \mu^k \frac{(t-1)^k}{k!} \tag{3.168}$$

Define a solution:

$$T_{1-x}\varphi_0(t) + T_x\varphi_1(t) = \varphi(x,t) \tag{3.169}$$

Which verifies:

$$\begin{cases} \varphi(x,0) & = f_0(x) \\ \varphi(x,1) & = f_1(x) \end{cases} \tag{3.170}$$

Following theorems 3.3.2 and 3.3.2, we may write the solution to the heat equation given in the previous corollary as:

$$\varphi(x,t) = (1-\Phi_\sigma(t)) \sum_{l=0}^{N-1} \mu^l \frac{t^l}{l!} f_0^{(2l)}(x) + \Phi_\sigma(t) \sum_{l=0}^{N-1} \mu^l \frac{t^l}{l!} f_1^{(2l)}(x) + \Sigma(x,t,\Phi_\sigma^{(1)},\Phi_\sigma^{(2)},\dots) \tag{3.171}$$

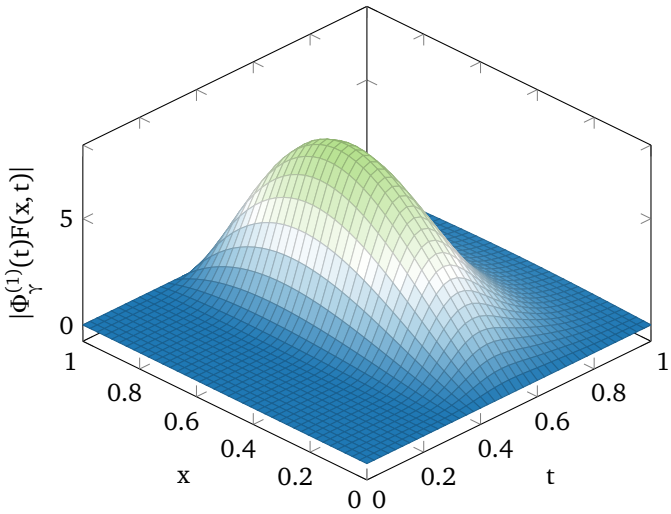


Figure 3.8 – Numerical evaluation of $|\Phi_\sigma^{(1)}(t)\Sigma(x,t)|$ $(\gamma = 1.5, \mu = 1)$.

Where Σ is a function containing the remaining of the successive derivatives of $\Phi_\sigma^{(1)}$. However, the two first derivatives are linked by equation (2.38), thus the previous equation may be written as:

$$\varphi(x,t) = (1-\Phi_\sigma(t))\sum_{l=0}^{N-1}\mu^l\frac{t^l}{l!}f_0^{(2l)}(x) + \Phi_\sigma(t)\sum_{l=0}^{N-1}\mu^l\frac{t^l}{l!}f_1^{(2l)}(x) + \Phi_\sigma^{(1)}(t)\Sigma(x,t) \tag{3.172}$$

The rest term $\Phi_\sigma^{(1)}(t)\Sigma(x,t)$ may be evaluated by comparing the result of a finite element computation of the heat equation and the first explicit term of the formal computation suggested previously. This allows to find the rest depicted in figure 3.8. In the following, we assume that computing the solution to the heat equation by means of finite differences is equivalent to least term summation of the formal solution suggested previously.

In the next chapter, we will see how the previous formal approach and this assumption allow to build trajectories for a multi-agent system. We will explain how we manage to respect geometric constraints either for the heat equation or for Burgers' equation.

3.4 The Hopf-Cole transformation and Gevrey functions

In the proposition 3.3.6 on page 68 we prove that the solutions to the heat equation generated by the T_x operator from Gevrey functions of order 0 in t are Gevrey of order 0 in both t and x. Even if we were not able to extend this result to Gevrey functions of greater order, it is of interest to study the effect of the Hopf-Cole transformation to Gevrey functions.

Proposition 3.4.1. *If $\varphi(x,t)$ is a Gevrey of order respectively α and 1 in t and x and if there exists $c > 0$ such as $\forall x \in [0,1], \forall t \in [0,1], \varphi(x,t) > c$ then $u = \varphi_x/\varphi$ is Gevrey of order respectively α and 1 in t and x.*

Proof. Let $v(x,t) = \ln\varphi(x,t)$. Since the logarithm function ln is analytic on $[c,\infty[$, it is Gevrey of order 1. By composition, the function $v(x,t)$ is Gevrey and has same order as φ as long as

the orders are greater than (or equal to) 1 (see on this the various results of composition of Gevrey functions given in [Yamanaka, 1989]). Since $u = v_x$, and since Gevrey orders are stable by derivation, u is Gevrey and has same order as v and thus, as φ. □

This results proves that strictly positive solutions to the heat equation generated by the operator T_x from Gevrey functions of order 0 in t result, after applying the Hopf-Cole transform, in analytic solutions to Burgers' equation.

CHAPTER 4

PDE-based motion planning framework

In the previous chapter, we presented a formal way to construct a solution to the heat equation with control on both side. In this chapter we present the framework we created to generate acceptable trajectories. In a first section we create solutions to the heat equation. In a second section, we present how to use the aforementioned method and Hopf-Cole transformation to generate solutions to Burgers' equation. Various aspects are discussed on how to generate trajectories for a multi-agent system constituted of leader and follower agents based on this solution.

Contents

3.1 Rewriting the heat equation with formal differential operators of infinite order . **53**
3.2 The heat equation with controls on both sides **55**
3.2.1 Objectives 55
3.2.2 Formal derivation 55
3.2.3 Computational implementation 61
3.3 Various properties of the T_x operator **62**
3.3.1 Polynomial states and controls 62
3.3.2 Application to the Weierstrass approximation theorem 66
3.3.3 Convergence of the T_x operator 67
3.3.4 Application of T_x to exponential functions 69
3.3.4.1 Convergent case 69
3.3.4.2 Numerical study of the convergence 71
3.3.5 The operator T_x and product of functions 73
3.3.5.1 Product with exponential functions 73
3.3.5.2 Product with the non-analytic Φ_σ function 76
3.4 The Hopf-Cole transformation and Gevrey functions **79**

4.1 Generating solutions to the heat equation

In the previous chapter, corollary 3.3.11 gives an explicit way to solve a transition in finite time between two polynomial states f_0 and f_1. This trajectory is a solution of the heat equation and is unidimensional. Suppose a state in dimension d described by $f_i = (f_i^0, \dots, f_i^{d-1}), i \geqslant 0$ where

$f_i = f(T_i)$. The times T_i are strictly increasing with respect to i and describe the time of the successive imposed states. As a matter of simplification, we consider in the following the only state at $T_0 = 0$ and $T_1 = 1$. A transition between two successive states f_i and f_{i+1} is described by d equations similar to those described in corollary 3.3.11.

Considering a single dimension, and following the formalism used in the controls and states described in corollary 3.3.11, the states:

$$\begin{aligned} f_0(x) &= \sum_{k=0}^{2N-1} p_k \frac{x^k}{k!} \\ f_1(x) &= \sum_{k=0}^{2N-1} r_k \frac{x^k}{k!} \end{aligned} \tag{4.1}$$

represent 4N degrees of freedom. The controls are strictly equivalent to these states. Our goal is to create trajectories for N agents. The trajectory of each of these agents is described by the trajectory of a point α_i of the heat equation $\varphi(\alpha_i, t)$. In this case, the positions of the agents at time t_0 and t_1 are equivalent to 2N of these degrees of freedom. The remaining 2N degrees of freedom may be used in different ways. In the following, we suggest to use them as waypoints for the left-most and right-most agents.

We write $\mathbf{r_0}$ (resp. $\mathbf{s_0}$, $\mathbf{p_0}$, $\mathbf{p_1}$, $\mathbf{q_0}$ and $\mathbf{q_1}$) the vector (r_{2i}) (resp. (s_{2i}), (p_{2i}), (p_{2i+1}), (q_{2i}) and (q_{2i+1})), F_0 and F_1 the upper-triangular matrices of generic terms $1/(2(j-i))!$ and $1/(2(j-i)+1)!$. Based on equation (3.166) we get the relations:

$$\begin{cases} \mathbf{r_0} = F_0\mathbf{p_0} + F_1\mathbf{p_1} \\ \mathbf{s_0} = F_0\mathbf{q_0} + F_1\mathbf{q_1} \end{cases} \tag{4.2}$$

Let the respective index of every agent be $i/(N-1), 0 \leq i < N$. Let $\mathbf{d^s} = (d_i^s)$ (resp. $\mathbf{d^f} = (d_i^f)$) be the initial (resp. final) formation, so that $f_0(i/(N-1)) = d_i^s$ (resp. $f_1(i/(N-1)) = d_i^f$). We denote by A the invertible Vandermonde matrix of generic term $((i/N-1)^{2l})_{0\leq i,l<N}$ and J_0 (resp. J_1) the diagonal matrix of generic term $(1/(2i)!)$ (resp. $(1/(2i+1)!)$) and H the diagonal non invertible matrix of generic term $(i/(N-1))$. We have:

$$\begin{cases} \mathbf{d^s} = AJ_0\mathbf{p_0} + HAJ_1\mathbf{p_1} \\ \mathbf{d^f} = AJ_0\mathbf{q_0} + HAJ_1\mathbf{q_1} \end{cases} \tag{4.3}$$

This sets the 2N equations for the positions. Introducing the matrices $Q = J_0^{-1}A^{-1}$, which is invertible and $R = HAJ_1$ (which is not), we get the following relations between odd and even coefficients:

$$\begin{cases} \mathbf{p_0} = Q\left(\mathbf{d^s} - R\mathbf{p_1}\right) \\ \mathbf{q_0} = Q\left(\mathbf{d^f} - R\mathbf{q_1}\right) \end{cases} \tag{4.4}$$

Therefore, the even coefficients of the states f_i depend on the chosen positions and on the odd coefficients. This odd coefficients are determined as follows.

We task the left trajectory (resp. right trajectory) with going, in this order, through the successive points $(e_i^0)_{0\leq i<N}$ (resp. $(e_i^1)_{0\leq i<N}$) at time $(t_i^0)_{0\leq i<N}$ (resp. $(t_i^1)_{0\leq i<N}$). Since the

positions of the left- and right-most agents at time $t = 0$ and $t = 1$ are already known, we take $0 < t_i^0, t_i^1 < 1$. Let $t_i^0 = t_i^1 = (i+1)/(N+1)$. Let P be the anti-diagonal matrix of generic term 1 (with $PP = I$). Let $\mathbf{\Phi}$ be the diagonal matrix of generic term $(\Phi_\gamma(t_i^0))_{0\le i<N}$. Since $\Phi_\gamma(1-t) = 1-\Phi_\gamma(t)$, we have $(I-\mathbf{\Phi}) = P\mathbf{\Phi}P$. Let V (resp. T) be the Vandermonde matrix of generic term $((t_i^0)^j)_{0\le i,j<N}$ (resp. $((t_i^0-1)^j)_{0\le i,j<N}$). Let B be the diagonal matrix of generic term $((-1)^i)_{0\le i<N}$ (notice we have $BB = I$). Since $t_i^j = 1-t_{N-1-i}^j, j \in \{0,1\}$, we have $V = PTB$. Let G be the diagonal matrix of generic term $(1/(i!))_{0\le i<N}$. We get, using the definition of the controls in equations (3.167) and (3.168):

$$\begin{cases} \mathbf{e^0} &= P\mathbf{\Phi}PVG\mathbf{p_0} + \mathbf{\Phi}PVGB\mathbf{q_0} \\ \mathbf{e^1} &= P\mathbf{\Phi}PVG\mathbf{r_0} + \mathbf{\Phi}PVGB\mathbf{s_0} \end{cases} \tag{4.5}$$

Using relation equation (4.2) in the latter and writing $D = P\mathbf{\Phi}PVG$ and $E = \mathbf{\Phi}PVGB$ (which are both invertible matrices), we get:

$$\begin{cases} \mathbf{e^0} &= D\mathbf{p_0} + E\mathbf{q_0} \\ \mathbf{e^1} &= D(F_0\mathbf{p_0} + F_1\mathbf{p_1}) + E(F_0\mathbf{q_0} + F_1\mathbf{q_1}) \end{cases} \tag{4.6}$$

Using equations equation (4.4) in the previous equations, writing $\hat{Q} = Q^{-1}D^{-1}$ and $\tilde{Q} = Q^{-1}F_0^{-1}D^{-1}$ – which are both invertible – we get the global system:

$$\begin{cases} \mathbf{p_0} &= Q\left(\mathbf{d^s} - R\mathbf{p_1}\right) \\ \mathbf{q_0} &= Q\left(\mathbf{d^f} - R\mathbf{q_1}\right) \\ \hat{Q}\mathbf{e^0} &= \mathbf{d^s} - R\mathbf{p_1} + K(\mathbf{d^f} - R\mathbf{q_1}) \\ \tilde{Q}\mathbf{e^1} &= \mathbf{d^s} - R\mathbf{p_1} + \tilde{Q}DF_1\mathbf{p_1} + L_0(\mathbf{d^f} - R\mathbf{q_1}) + L_1\mathbf{q_1} \end{cases} \tag{4.7}$$

Where $K = \hat{Q}EQ$, $L_0 = \tilde{Q}EF_0Q$ and $L_1 = \tilde{Q}EF_1$, are invertible matrices. Subtracting the third line to the fourth, we get:

$$\begin{cases} \mathbf{p_0} &= Q\left(\mathbf{d^s} - R\mathbf{p_1}\right), \mathbf{q_0} = Q\left(\mathbf{d^f} - R\mathbf{q_1}\right) \\ \hat{Q}\mathbf{e^0} &= \mathbf{d^s} - R\mathbf{p_1} + K(\mathbf{d^f} - R\mathbf{q_1}) \\ \tilde{Q}DF_1\mathbf{p_1} &= \tilde{Q}\mathbf{e^1} - \hat{Q}\mathbf{e^0} + (K - L_0)\mathbf{d^f} + ((L_0 - K)R - L_1)\mathbf{q_1} \end{cases} \tag{4.8}$$

Then, injecting the fourth line in the third, we get an expression of $\mathbf{q_1}$ depending only on $\mathbf{e^0}$, $\mathbf{e^1}$, $\mathbf{d^s}$ and $\mathbf{d^f}$. Assuming the matrix $RF_1^{-1}D^{-1}\tilde{Q}^{-1}((L_0-K)-L_1)+KR$ is invertible, we get an exact expression of $\mathbf{q_1}$. Using the three other equations, we get the three other vectors. The four remaining vectors $\mathbf{r_0}$, $\mathbf{r_1}$, $\mathbf{s_0}$ and $\mathbf{s_1}$ are found using equation (4.2).

Therefore, it is possible, using equation (4.8), to construct a set of coefficient for the controls. The computed solution to the heat equation matches at the different time steps the chosen positions of the agents and the waypoints of the left-most and right-most agent. This is illustrated by figure 4.1 for four agents and the consequent four waypoints. All the trajectories pass, as expected, through the given waypoints, up to the computational inaccuracy. It appears that the smaller the "stiffness" γ of the basis transition function is, the higher is the "aggressiveness" of the trajectory. The corresponding solution for a chosen value of γ is depicted in figure 4.2. It matches the given start and end formations and gives as a result four smooth trajectories for the four agents.

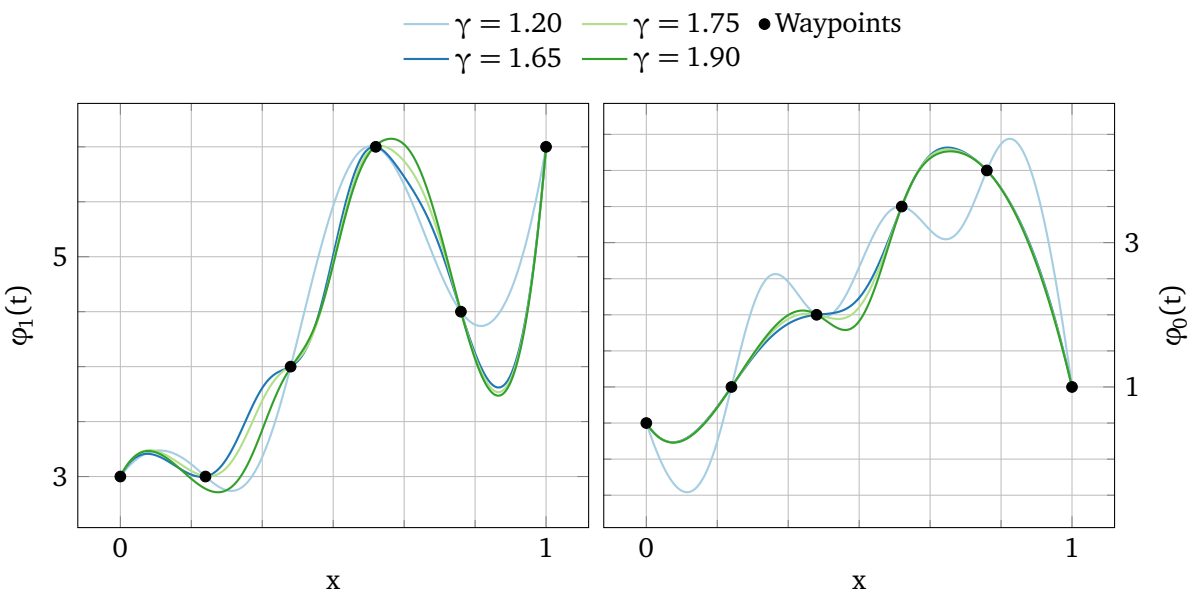


Figure 4.1 – Example of controls for defined waypoints for various values of γ ($\mu = 1.0$).

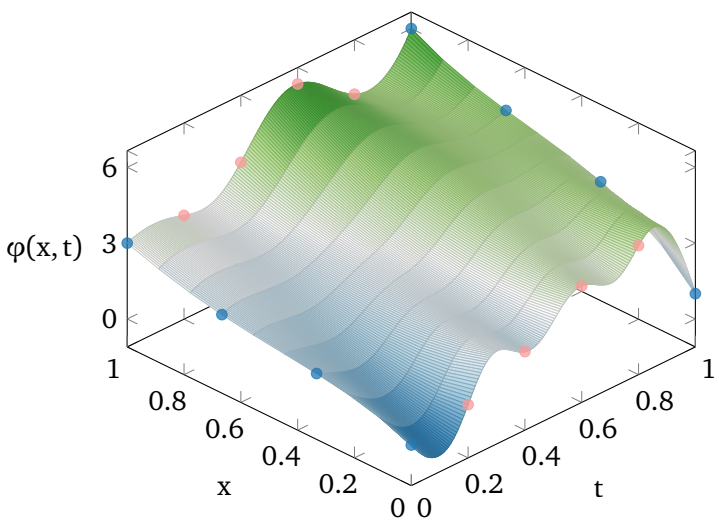


Figure 4.2 – Numerical resolution of the heat equation ($\gamma = 1.2$, $\mu = 1.0$) for given start and end states (depicted by blue dots) and given waypoints (depicted by pink dots) .

4.2 Generating solutions to Burgers' equation

In the previous section, we showed how solutions to the heat equation can be produced. These solutions use half of the available degrees of freedom to so that the left-most and right-most agent pass through preset waypoints. In this section, we show how this method can be adapted to construct solutions to Burgers' equation.

4.2.1 Optimization of the trajectories

It is possible to turn a solution θ to the heat equation constructed in the previous sections into a solution φ to Burgers' equation. This is done thanks to Hopf-Cole transformation that was recalled and proved in theorem 2.1.3:

$$\varphi = -2\mu \frac{\theta_x}{\theta} \tag{4.9}$$

Creating an appropriate solution is thus a two-sided problem. On the one hand, the solution to the heat equation θ has to be of strictly constant sign. We may see in the sample controls depicted in figure 4.1 that adding waypoints create some "overshoot" between these waypoints. This can lead the solution to cross the origin. On the other hand, the solutions to Burgers' equation obtained from the Hopf-Cole transformation should follow a determined set of conditions. This two-sided problem is solved by means of optimization.

Formally, we may define our goals as follows. First, we want the solution to the heat equation to be strictly positive. We define c, the minimum of $\varphi(x,t)$ on the unity square (e.g. $\mathcal{A} = \{(x,t), 0 \leqslant x, t \leqslant 1\}$). Thanks to the minimum principle (see e.g. [Logan, 2008, part. 7.2.1, p. 353]), this minimum is located on the border of the unity square (e.g. $\mathcal{A} - \mathring{\mathcal{A}}$ where $\mathring{\mathcal{A}}$ is the interior of $\mathcal{A}$: $\mathring{\mathcal{A}} = \{(x,t), 0 < x, t < 1\}$). This minimum can be found as:

$$c = \min\left\{\inf_{0\leqslant x\leqslant 1} f_0(x), \inf_{0\leqslant x\leqslant 1} f_1(x), \inf_{0\leqslant t\leqslant 1} \varphi_0(t), \inf_{0\leqslant t\leqslant 1} \varphi_1(t)\right\}, \tag{4.10}$$

Second, if the first condition is verified, *i.e.* if $c > 0$, we may evaluate the solution to Burgers' equation using the Hopf-Cole transformation. This solution $u(x,t)$ on $\mathcal{A}$ is evaluated using a finite difference scheme. Doing this, we want the function $u(x,t)$ to meet specific requirements. These requirements can depend on the objectives of the system. In the case of trajectories in space, these objectives may encompass collision avoidance. In the following, we will minimize the lengths of the paths of the leader and the anchor. This length is evaluated as:

$$\langle u\rangle(x) = \int_0^1 |u_t(x,s)|ds - |u(x,1) - u(x,0)|. \tag{4.11}$$

The overall cost function is then defined as:

$$\langle \mathbf{f}_0, \mathbf{f}_1\rangle = \begin{cases} c & \text{if } c \leqslant 0 \\ 1/(\langle u\rangle(0) + \langle u\rangle(1)) & \text{if } c > 0 \end{cases}. \tag{4.12}$$

A convenient way to find extrema of certain functions is to use Particle Swarms Optimization (short **PSO**, see subsection 1.1.2 – Methods based on Particle Swarm Optimization). An advantage of PSO over other optimization frameworks, is that it does not need derivations of the cost function. Therefore, this process is adapted to optimization in high dimension or with noisy data. A simple description of the PSO algorithm can be given in two steps as follows:

Initialization: A set of n particles – called the swarm – is created with random positions. These positions are uniformly distributed over the solution space $[\mathbf{x}_{\min}, \mathbf{x}_{\max}]^n$. The speeds of the particles are randomly initialized. This initialization is done with value uniformly distributed over $[-|\mathbf{x}_{\max} - \mathbf{x}_{\min}|, |\mathbf{x}_{\max} - \mathbf{x}_{\min}|]^n$. The local best position of each particle is initialized to the particle's initial position while the global best particle is searched among them.

Evolution: At each step of the evolution, the speed of each particle is updated based on its current speed and position and on the knowledge of the best local and global particles. The particles are then moved according to the new speed and evaluated.

It is believed that the swarm will converge to the global optimum if it exists or, if not, to a local optimum. However this convergence depends on the various parameters that can be chosen and the swarm can, in some cases, diverge or oscillate. The choice of these parameters has thus to be performed with care whereas with a certain freedom.

Inspired by the way the coefficients for the state representations f_0 and f_1 were found, we search for a way to find these coefficients using the Hopf-Cole transformation. We may write, for each of the agent of index x_k,:

$$\frac{u(t_j, x_k)}{-2\mu} f_j(x_k) = f_j'(x_k) \tag{4.13}$$

We search for an appropriate matrix M_j so that:

$$\mathbf{p}_1^j = M_j \mathbf{p}_0^j, \tag{4.14}$$

Where $\mathbf{p}_i^j$ corresponds either to $\mathbf{p}_i$ or $\mathbf{q}_i$ in the preceding. We write U_j the diagonal matrix of the scaled positions $(-u(x_k^j, t_j)/2\mu)_{0 \leqslant k < N}$ of the N agents at time t_j. We write A_j the matrix of generic term $((x_k)^{2l}/(2l)!)_{0 \leq k < N, 0 \leq l < N}$ and B_j the matrix of generic term $((x_k)^{2l+1}/(2l+1)!)_{0 \leq k < N, 0 \leq l < N}$. D is the derivation matrix :

$$D = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ \vdots & & & \ddots & 1 \\ 0 & \cdots & \cdots & \cdots & 0 \end{pmatrix} \tag{4.15}$$

Equation (4.13) can be written as:

$$U_j(A_j \mathbf{p}_0^j + B_j \mathbf{p}_1^j) = B_j D \mathbf{p}_0^j + A_j \mathbf{p}_1^j, \tag{4.16}$$

which can be put in the form

$$(U_j B_j - A_j)\mathbf{p}_1^j = (B_j D - U_j A_j)\mathbf{p}_0^j, \tag{4.17}$$

During the optimization process, the particles are chosen in the N-dimensional space of $\mathbf{p}_0^j$ (or 2N-dimensional space of $\mathbf{p}_0^j$ and $\mathbf{p}_0^{j+1}$ as will be explained in the following). The odd coefficient are then computed based on the positions of the agents and on the particles. Algorithm 1.1 is then adapted and applied to this system. As a result, it outputs a possibly optimal set of coefficients for the state representation f_0 and f_1. This is adapted as explained in the following subsections.

4.2.2 Leaders and followers

In our framework, we distinguish two types of agents: *leaders* and *followers*. The agents introduced in the previous section for which a start and an end state can be imposed are *leaders*. Their positions are determined and chosen at all the desired transition states. As a consequence of the adopted formalism, a function f_i in equation (3.166) represents a formation of N leaders.

In addition to these leaders, another class of agents is introduced: followers. This class is inspired by the agents used in [Meurer and Krstić, 2011]. Their positions in the start and end states are a result of the position of the leaders. The number of followers is not limited by the representation chosen in equation (3.166). These followers are described by indices chosen between the indices of the leading agents. This choice can represent a specific topology.

The status of leaders and followers can be changed between successive deployments and the number of leaders is not fixed. A system of three leaders and two followers during a deployment can be turned in any system with two to five leaders in the following deployment. However, changing the organization of the system impacts the complexity and the efficiency of the optimization problem.

4.2.3 Transition between successive steps

Let there be three successive states of a formation of N leaders and M followers at time t_0, t_1 and t_2. These three formations are described by sets of N ordered positions s_0, s_1 and s_2. The first transition from s_0 to s_1 is obtained by optimizing the functions f_0 and f_1 of the form given in equation (3.166) on page 78 representing respectively s_0 and s_1.

To create the transition from state s_1 to state s_2, two approaches are considered. The first solution consists in evaluating only the best possible representation f_2 of the state s_2 by keeping the representation f_1 of the state s_1. This reduces the size of the optimization space by a factor two and is compatible with the use of followers. Since the representation of the state is kept, the position of the followers is coherent. However, the resulting solution is not the best possible solution and can result in longer trajectories.

The second solution consists in evaluating again the representation f_1 and f_2 of the states s_1 and s_2. This solution leads to better optimized trajectories at the cost of a wider optimization space. Moreover, as the representation of the start state changes, the starting positions of the followers is not coherent with their ending position at the previous step. An error is induced that has to be corrected by the closed-loop controllers of the followers.

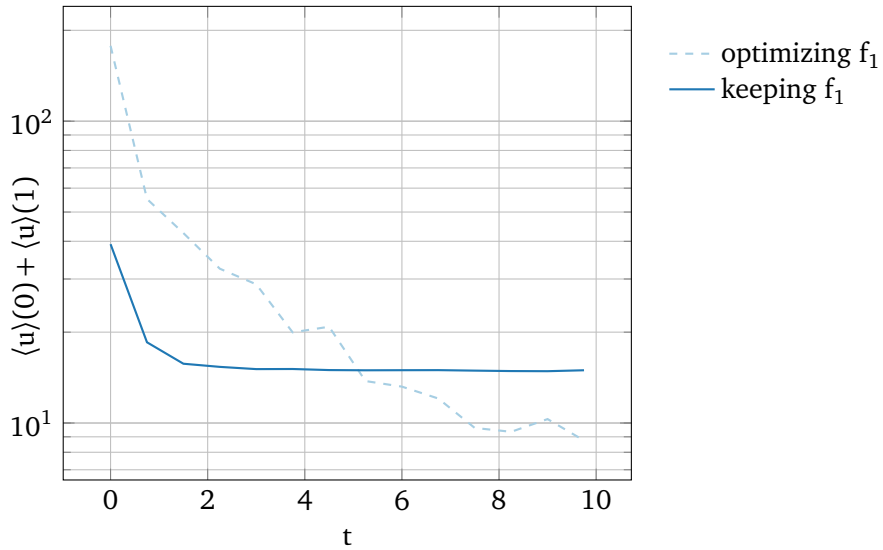


Figure 4.3 – Time evolution of the cost for a team of four leaders (average on 50 runs with fixed points, $\gamma = 1.65, \mu = 1.0$).

The difference in optimization time and efficiency between these two solutions is represented in figure 4.3 for a team of four leaders. In our configuration, the optimization of both the start and the final representations (depicted in light blue, dashed) gives better results than optimizing the only final representation (depicted in dark blue), provided the optimizer has enough time. In the case where a solution is needed in the shortest possible time, keeping the representation f_1 is the best solution. Advantages and drawbacks of these two methods are gathered in table 4.1

	keeping f_i	optimizing f_i
resolution speed	+	-
optimized	-	+
adapted to followers	+	-

Table 4.1 – Comparison of the two suggested methods for state representation optimization.

These methods are illustrated in figure 4.4 on the next page where the final state of the second deployment is the start state of the first deployment ($s_2 = s_0$). Three possible return methods are presented. The first deployment from s_0 to s_1 is drawn in light blue. As it is the first deployment, both the start and end formation representations have been optimized. This results in short and smooth paths.

For the way back, the first possibility, drawn in dark bluein figure 4.4 on the facing page, has been obtained by optimizing again both the start and end representations. This results in equally short and smooth paths whereas different from the direct way. This is a sign of the nonlinearity of Burgers' equation as the trajectories between two points is not the same depending on the sense of travel. The second possibility, drawn in light green, was obtained by keeping the representation of the final formation of the previous deployment as the representation of the new start formation and optimizing the representation of the new final formation. This results in a path longer than in the previous case but was obtained in notably less time. A last

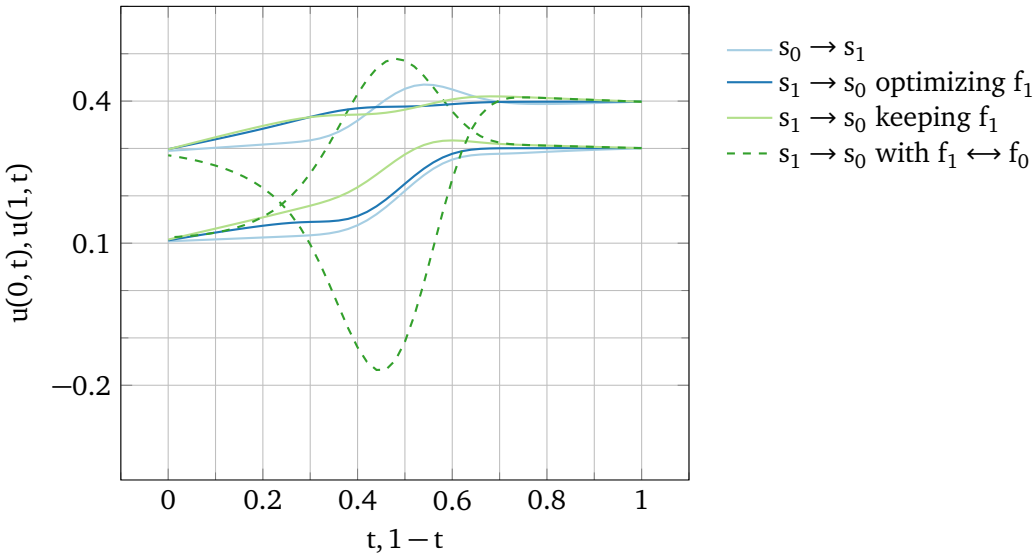


Figure 4.4 – Three return possibilities ($\gamma = 1.65, \mu = 1.0$).

possibility, that is not ensured to work, is given as the dark green dashed line. This is obtained by only exchanging the role of the previous start and final formation representation. Due to the definitions of the controls in equations (3.167) and (3.168) (and notably due to the choice of powers of $t-1$ instead of powers of $1-t$), the systems is not invariant by the change of variable $t \leftarrow 1-t$. In the case where the resulting solution to the heat equation is strictly positive, it is still possible to compute a solution to Burgers' equation that matches the given start and end formation. However, this solution undergoes no optimization process and the resulting solution, as in the case depicted in figure 4.4, is not optimal at all.

4.3 Combining solutions to create trajectories

The generation of solutions presented in the previous sections was presented in only one dimension. Trajectories in more dimensions are created by combining solutions created in one dimension into solutions for a higher dimensional space.

As a matter of illustration, an example in two dimensions with three leaders and eighteen followers is created. First, two solutions on the two dimensions are computed. They are created to ensure the transition between the starting formation and the final formation given in table 4.2. They result in the coefficients given in table 4.3. The resulting trajectories, for each dimension, are presented in the two subfigures of figure 4.5. Then, these trajectories are combined into the two-dimensional trajectories depicted in figure 4.6. The thick solid lines represent the trajectories of the three leaders while the other thin lines represent, horizontally, the state of the formation at time t and, vertically, the trajectories of the agents. In the right subplot of figure 4.5, a contraction of the formation is observed. This does not lead to collisions in the final trajectory due to the distance among the agents in the other dimension. However, a more careful optimization could be needed to ensure the absence of collision due to the non-zero size of the agents.

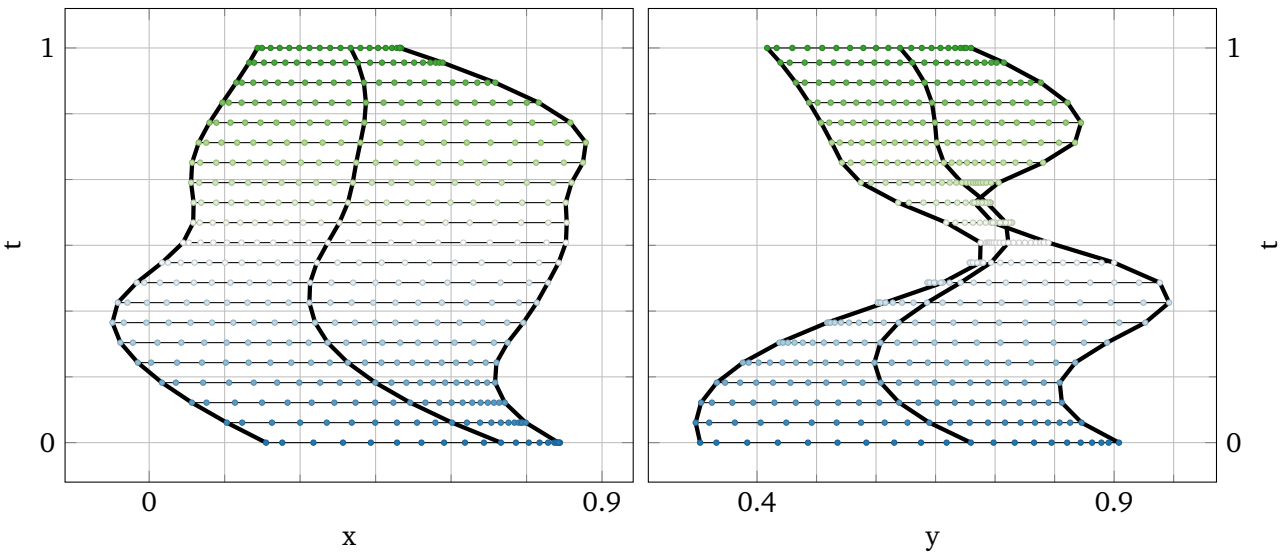


Figure 4.5 – Two solutions to Burgers' equation for 3 leading agents (thick solid lines) and 18 follower agents ($\gamma = 1.5, \mu = 1.0$).

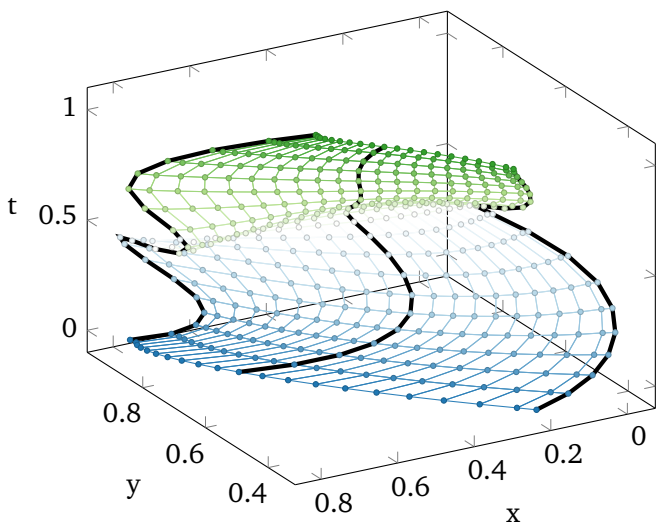


Figure 4.6 – Trajectories in a plane of 3 leading agents (thick solid lines) and 18 follower agents ($\gamma = 1.5, \mu = 1.0$).

index of the agent	0	0.5	1
starting formation	$(0.2, 0.3)$	$(0.7, 0.7)$	$(0.8, 0.9)$
final formation	$(0.2, 0.4)$	$(0.4, 0.6)$	$(0.5, 0.7)$

Table 4.2 – Starting and Final formations for the leaders used in the simulation.

First dimension: x

	0	1	2	3	4	5
p_i	66.623	−6.662	−43.940	57.574	57.518	−172.555
r_i	48.545	19.418	13.635	−28.815	−115.037	172.555

Second dimension: y

	0	1	2	3	4	5
p_i	10.318	−1.548	−4.235	1.593	17.517	−30.274
r_i	7.396	3.328	1.071	−3.973	−12.757	30.274

Table 4.3 – Coefficients (rounded to the third decimal) describing the formations for both dimensions.

The paths in the (x, y) plane of the three leaders and of two chosen followers are shown in figure 4.7. Whereas some of these paths intersect, this is not the case of the trajectories. This can be visually checked in figure 4.6. This representation shows that there are no collisions between the different agents.

4.4 Conclusion and perspectives

In this first part, we presented, first, a formal method based on the flatness of the heat equation to create solutions to the heat equation. This method is applied, using an optimization process and the Hopf-Cole transformation, to create solutions to Burgers' equation. These solutions are used as the base of a framework to plan trajectories for a multi-agent system.

This framework allows to create solutions for a set of agents, either leaders (for which a final position can be precisely set) or followers (for which the final position is a result of the final positions of the leaders), evolving in a space of arbitrary dimension. The problem of successive deployments are considered and computational efficiency of optimizing these transitions is studied.

The framework appears efficient. However, more work would be necessary to ensure obstacle and collision avoidance. These objectives could be, in future works, added to the cost function used during the optimization process. The tracking of these trajectories is supposed to be performed by closed-loop controllers implemented on the various agents. Such a problem will be considered in the following part.

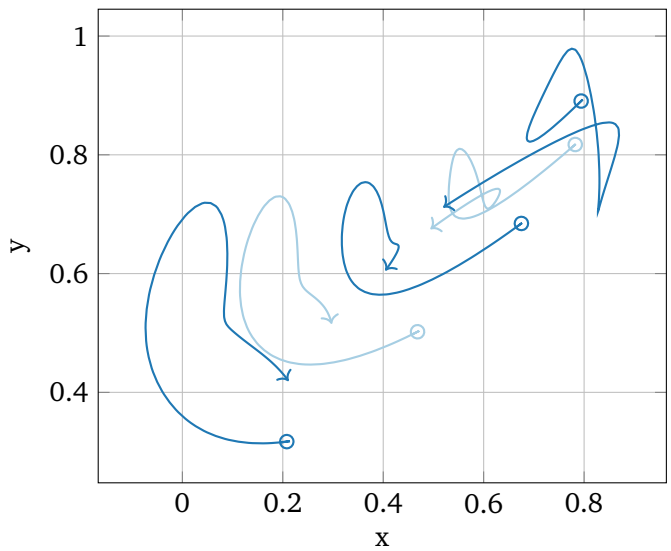


Figure 4.7 – Trajectories of the three leaders (dark blue) and of two chosen followers (light blue) in the (x, y) plane. Starting positions are indicated by circles, final positions by arrows ($\gamma = 1.5, \mu = 1.0$).

Part II

Modeling and control of a trirotor UAV

CHAPTER 5

Unmanned Aerial Vehicles: a brief review

This chapter is dedicated to a brief review on aerial vehicles with an emphasis on control of Unmanned Aerial Vehicles (**UAV**s). In the first section we review some basics on flight dynamics. We then present various classes of aerostats and aerodynes and give a short overview on fixed-wing, rotary-wing, and convertible aircrafts as well as present various models. We devote the second section to the study of the quadrotor UAV. This family of UAVs is the most spread class of small scale multirotor UAVs. Its configuration approaches the structure of the UAV studied in this thesis. We present a brief survey of works conducted to control quadrotors and to perform load transportation with them. In the last section, we review the existing works that have been conducted on multirotor UAVs equipped with tilting rotors.

Contents

4.1 Generating solutions to the heat equation **81**
4.2 Generating solutions to Burgers' equation **85**
4.2.1 Optimization of the trajectories 85
4.2.2 Leaders and followers 87
4.2.3 Transition between successive steps 87
4.3 Combining solutions to create trajectories **89**
4.4 Conclusion and perspectives **91**

5.1 Flight dynamics

5.1.1 Basics of flight

The world of man-made aerial vehicles is separated since the beginning into two categories, aerostats and aerodynes. Aerostats are lighter-than-air aircrafts. They derive the necessary lift to fly from buoyancy. They are called aerostats since they can fly without airflow on their bodies. Aerodynes, on the contrary, are heavier-than-air aircrafts. They derive the necessary lift from a dynamic air flow.

5.1.1.1 Aerostats

There are two ways for an aerostat to get buoyancy. The first way is by filling the aerostat's envelope with warm air. Warm air is indeed lighter than the – fresher – ambient air. The aerostat

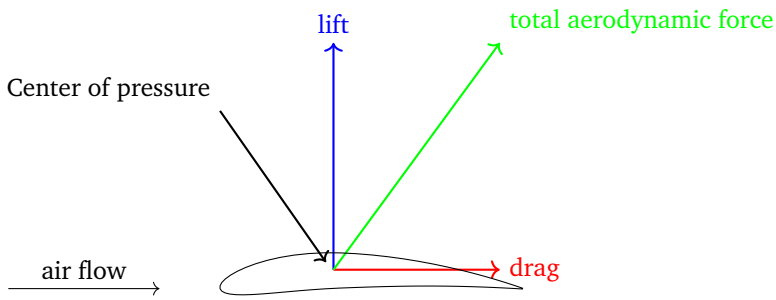


Figure 5.1 – Aerodynamic forces: lift and drag on a foil (NACA4612)

needs to carry a burner and propellant to heat the gas, either to control the flight altitude or to compensate heat losses. The first recorded manned untethered free flight in history was conducted by Jean-François Pilâtre de Rozier and François d'Arlandes on November 21, 1783. They flew a distance of about 9 km in about twenty-five minutes taking off in front of king Louis XVI from the Château de la Muette, now near the Porte de la Muette, Paris, and landing on the Butte aux Cailles, now in the south-eastern XIIIth arrondissement of Paris, at a maximum estimated altitude of thousand meters.

The other solution was developed at the exact same time. It consists in filling a closed envelope with a lighter than air gas. On December 1, 1783 , Jacques Charles and Nicolas-Louis Robert made a two hours flight covering thirty-six kilometers from the Tuileries Garden in the center of Paris to Nesles-la-Vallée, north-west from Paris. Their balloon was inflated with hydrogen obtained by the reaction of sulphuric acid on iron. Other lifting gases can either be helium, ammonia, methane or coal gas – which has been historically widely used for that purpose. Gas balloons face several problems e.g. the very high price of helium, the extreme inflammability of hydrogen, methane or coal gas, the permeability of usual envelopes to hydrogen and helium.

Theses two classes of aircrafts are called balloons and are designated respectively as hot air and gas balloons. Balloons have several advantages such as very long endurance or very high ceiling – indeed, in 2002, a balloon reached the altitude of fifty-three kilometers [Yamagami et al., 2004], the highest altitude ever reached by an atmospheric vehicle. However they lack controllability and move with the surrounding air flow. The lack of controllability can be partially solved by adding propellers and adopting an aerodynamic envelope creating rigid airships, such as zeppelins, equipped with a rigid structural framework, semi-rigid airships, which have a partial structural framework, or blimps, which have no structural framework.

5.1.1.2 Aerodynes

Aerodynes are heavier-than-air vehicles. In order to produce lift they have to create aerodynamic forces to compensate their weight. Indeed a fluid, either liquid or gaseous, flowing past a surface exerts a force on the surface, respectively called hydrodynamic and aerodynamic forces. These forces are decomposed, as illustrated in figure 5.1, in a normal component, the lift, and a parallel component, the drag. The depicted air flow illustrates the velocity and orientation of the flow at infinity, that is without the influence of the airfoil.

A wing or a blade of a propeller produces therefore aerodynamic lift and drag. These forces greatly depend on the geometry of the foil – the cross section of the wing or blade. They also

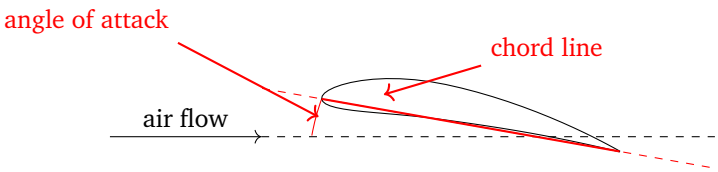


Figure 5.2 – Geometry of an airfoil (NACA4612)

depend on the angle – called the angle of attack – between the oncoming flow and the chord line – the imaginary line joining the leading and trailing edges of the foil as depicted in figure 5.2. Works are conducted to propose mathematical models of these forces, for example by [Pucci et al., 2011].

A simple explanation of the origin of the aerodynamic force can be found in considering a finite volume of air. The trajectory of this element is deflected along the wing. As such, the celerity of the element is changed, hence an acceleration does occur. This acceleration is caused by a force which, summed over the wing, gives the total aerodynamic force. This force is applied to the center of pressure. The position of the center of pressure evolves with the flight parameters such as the angle of attack and the air velocity.

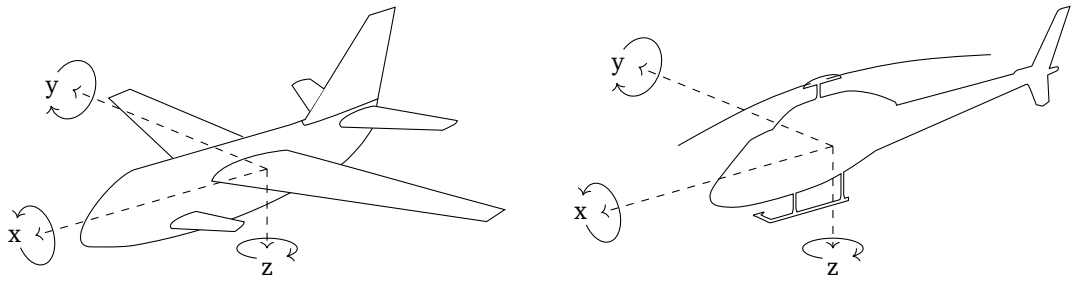
As can be deduced from the previous model, a symmetric airfoil will produce zero lift when parallel to the airflow. The drag and the lift of the airfoil will then evolve with the angle of attack. The lift and drag forces induced by the airfoil are studied through wind tunnel experimentations or, increasingly, through numerical simulations. Each foil has a best angle of attack, the critical angle of attack, where the produced lift is maximum. As such, the profile of the foil is chosen to match its typical service conditions. The critical angle of attack depends however on the relative air speed. It is said to be critical since, past this limit, the lift rapidly decreases because of the greater flow separation induced by the foil. This situation, called stall, is to be avoided in usual flight conditions and was the cause of many accidents such as the flight AF447 Rio-Paris in 2009.

There are two ways to induce air flow on a wing – independently of wind – to produce lift. One can either have wings fixed to the aircraft's body and move the whole aircraft or let the wings move with respect to the body. Aircrafts belonging to the first category are called *fixed-wing* aircrafts while the others are called *rotary-wing* aircrafts.

5.1.2 The roll-pitch-yaw convention and aircrafts' centers

In the following, we are led to describe the position and attitude of an aircraft. We call the three rotation angles described in figure 5.3 respectively roll (about the body longitudinal axis, chosen to be the x axis), pitch (about the body lateral axis, chosen to be the y axis) and yaw (about the body vertical axis, chosen to be the z axis). Mathematical model of aerodynes commonly use rotation matrices to describe the attitude. It is the choice we made in this work. The sequence of rotations chosen is the $x-y-z$ sequence, meaning the attitude is obtained first by the roll angle, then by the pitch angle and then by the yaw. Since we chose a z axis pointing downwards, this convention is usually called the *North-east-down* convention.

Usually, the origin of the body-fixed frame is chosen at the center of mass of the aircraft. The center of mass and the center of pressure usually do not coincide. However the aircrafts are commonly designed so that, in normal conditions, these two points are on the symmetry



(a) On an airplane. (b) On a helicopter.

Figure 5.3 – The roll-pitch-yaw convention.

plane of the aircraft so that the plane remains stable around the roll and the yaw axes. Before the advent of automatic control, the aircrafts were also designed so that the aircraft remained stable around the pitching axis. The inherent stability of such aircrafts is needed for civilian applications where the aircrafts need to remain steerable even in the case of autopilot failure. However, inherent stability can be a disadvantage for military aircrafts. They usually need a greater maneuverability brought by their inherent instability.

The pitch angle should however not be confused with the angle of attack. The pitch angle is defined with respect to the Earth while the angle of attack is defined with respect to the air flow. Indeed, planes can do acrobatics such as looping without exceeding the critical angle of attack.

5.1.3 Fixed-wing aircrafts

One usually distinguishes three classes of fixed-wing aircrafts: kites, gliders and aeroplanes. Kites are aircrafts tethered by one or more rope lines to a point. The rope lines are essential for the kite, they provide control and traction forces, for example as a power source for kite surfing or to lift an observer or payload like a human observer or meteorological instruments. However, kites require a sufficient wind which is not always available. Gliders are aircrafts whose free flight does not depend on an engine. To take-off, they need either to be towed by another aeroplane, by a car, or by a stationary winch rapidly winding a cable. Some gliders can take off using a retractable engine which is not used during the flight.

Aeroplanes are powered aircrafts. They use the thrust from a jet-engine, a propeller or a rocket to gain forward speed and experience aerodynamic lift. Aeroplanes can experience gliding flight, either as a test of their flight properties during design period, as a normal operation for example for rocket powered aeroplanes[1] or after a technical failure such as fuel exhaustion[2] or the loss for any other reason of the thrust.

An example of the structure of an aircraft is shown in figure 5.4. Every aircraft is built around a body (in red) called the fuselage [3] and wings (in grey). Throughout the history of aeroplanes,

[1] Such as the World War II German interceptor Messerschmidt Me 163 or the two reusable spacecrafts, the US American Space Shuttle and the Soviet Buran.

[2] As did occurr to a Boeing 767 of Air Canada, nicknamed “Gimli Glider” on July 23, 1983 which landed after gliding about 70 km at Gimli.

[3] Some planes such as the North American F-82 Twin Mustang, which saw service in the United Stated Air Force from 1946 to 1953, were built around a twin fuselage.

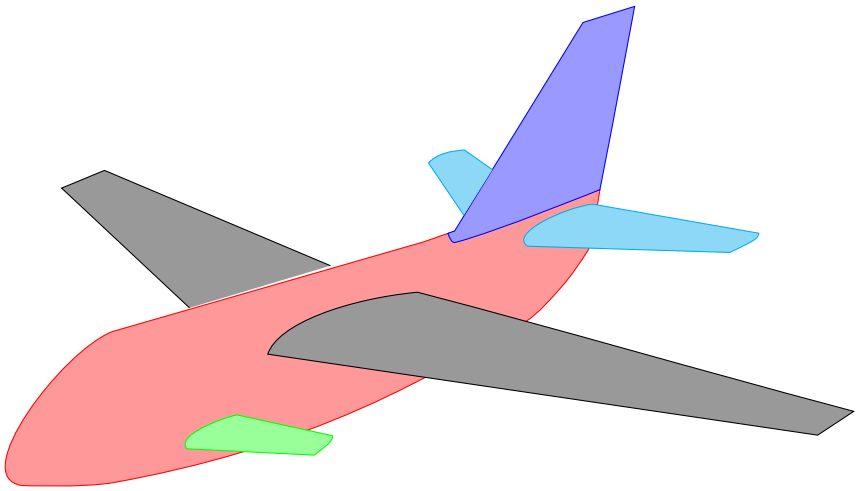


Figure 5.4 – The different elements of an aeroplane.

there have been several wing configurations. Designs have been used with one or more wings which can be mounted at various heights of the fuselage. Furthermore many different wing designs have been used. Additional small lifting surfaces are usually added either at the rear of the aircraft, called the tailplane (in light blue), or at the front, called the canard plane (in green). They are used as horizontal stabilizers. A vertical structure, called the fin (in blue) provides stability in rotation around the vertical axis. Fin and tailplane form the empennage. The wing, as well as the other additional surfaces embeds movable surfaces used as controls.

The propulsion of aeroplanes can be done through different methods. At the beginning of aviation history, the only available motors were steam engines. Clement Ader powered his Éole with such low-power engines and is reported to have taken off in 1890 for an uncontrolled flight on a distance of about thirty to fifty meters. The development of inner combustion engines combined with the development of piston engines allowed the Wright brothers to be the first pilots of humankind. Their Whight Flyer was a lightweight controlled glider equipped with a straight-four pistons engine achieving 9 kW. Since then, the development of new propulsion technologies such as turboprops, in the family of shaft engines, or, in the family of reaction engines, turbines, ramjets and rockets, made possible the development of a broad spectrum of airplanes. Each of these technologies have dedicated use for specific aircrafts and applications.

The increase in power of the engines led to the advent of post-stall – or supermaneuverable – combat aircrafts. Enjoying a high thrust to weight ratio and the ability to vector this thrust, these aircrafts can perform maneuvers such as Pugachev's Cobra[4], the Kulbit[5] or Herbst maneuver[6]. These air combat maneuvers do not rely on the plane's control surfaces nor on the aerodynamic forces described previously[7]. Instead, the execution of these maneuvers heavily relies on thrust vectoring. Therefore, while performing these maneuvers, the dynamics of the airplane changes considerably.

[4]In this maneuver, an airplane in slow level flight rapidly rises its nose far above the critical angle of attack, pitch and angle of attack reach values of about 90 to 120 degrees, before lowering the nose back and resuming normal flight.

[5]A very tight looping starting as Pugachev's Cobra.

[6]A very tight U-turn at a very high angle of attack.

[7]If we do not consider the internal aerodynamics of the engines

Aircraft name	max. take-off weight	max. speed	range	service ceiling
MiG-31	46 200 kg	Mach 2.83	3300 km	24 200 m
Concorde	185 100 kg	Mach 2.23	6200 km	18 300 m
Airbus A380	575 000 kg	Mach 0.96	15 700 km	13 000 m

Table 5.1 – Typical characteristics of emblematic fixed-wing aircrafts

Fixed-wing aircrafts have several advantages. They are energy efficient for displacement. Indeed, they can even fly without motors: kites and gliders are un-powered fixed-wing aircrafts. On a powered fixed-wing aircraft, the motors have, in normal mode, simply to compensate the drag force. This is necessary to maintain a sufficient speed and to experience enough lift. The ratio between lift and drag is called the lift-to-drag ratio. A higher ratio at usual conditions is usually sought as it reduces fuel consumptions and improves performances such as climb performance and glider performance. Fixed-wings aircrafts can therefore experience high take-off load, high speed, high endurance and high cruising altitude. Some information is gathered in table 5.1 for some emblematic airplanes showing typical values of these characteristics.

Nevertheless, fixed-wing aircrafts also have drawbacks. They can, in normal operational mode, only move in the direction imposed by the wings and the motors. They are, hence, unable to translate in the two other directions. Furthermore, since it is usually needed to avoid stall, a minimum speed has to be observed with respect to the wind and thus, planes can't usually hover. Because of this minimum speed, planes also usually need a rather long runway for take-off and landing.

5.1.4 Rotary-wing aircrafts

Besides fixed-wing aircrafts, another way to produce lift is using a rotary wing. As in the case of airplanes, rotary wing aircrafts can take several forms. The most widespread form of rotary-wing aircrafts is helicopters. However the first successful flight of a stable rotary-wing aircraft was accomplished on an autogyro, created by Juan de la Cierva. An autogyro relies on a propeller providing forward thrust and on the lift induced by the autorotation of an unpowered and free-spinning rotor. Rotors are made of blades which are long airfoils. As for propellers, the blades of a usual rotor are linked at one end to a central rotating mast. Autorotation is the state of a rotor spinning as a consequence of the air flow moving through the said rotor.

The success of Cierva's design is due to him understanding the problem of dissymetry of lift. This effect appears when the airflow respective to the aircraft body is not zero – typically in forward flight or under harsh wind conditions. In such a case, the tip of the blade advancing toward the airflow has a speed greater than the tip of the opposite – retreating – blade as illustrated by figure 5.5. Since, in the usual service domain, the blades are designed to produce more lift when the speed of the airflow increases, this leads to the advancing blade producing more lift than the retreating one. This dissymmetry of lift is avoided by letting the blade "flap". The blades are designed to be flexible, they lift and twist up when advancing – the angle of attack decreases thus producing less lift – and down when retreating – the angle of attack increases thus producing more lift. This can also be done by increasing the pitch of the blades differentially on the cycle. However, when the air speed is very high, two problems occur. On the one hand,

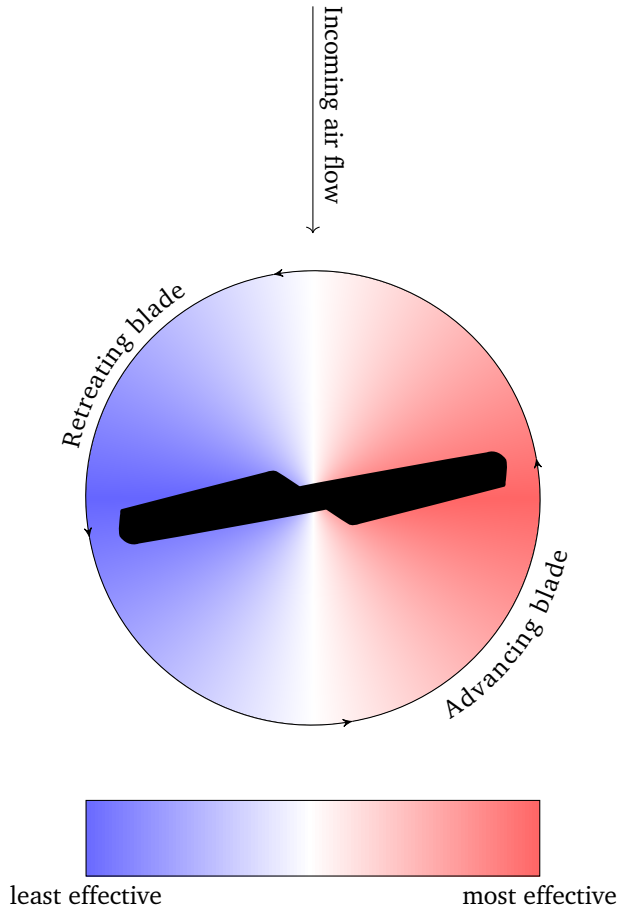
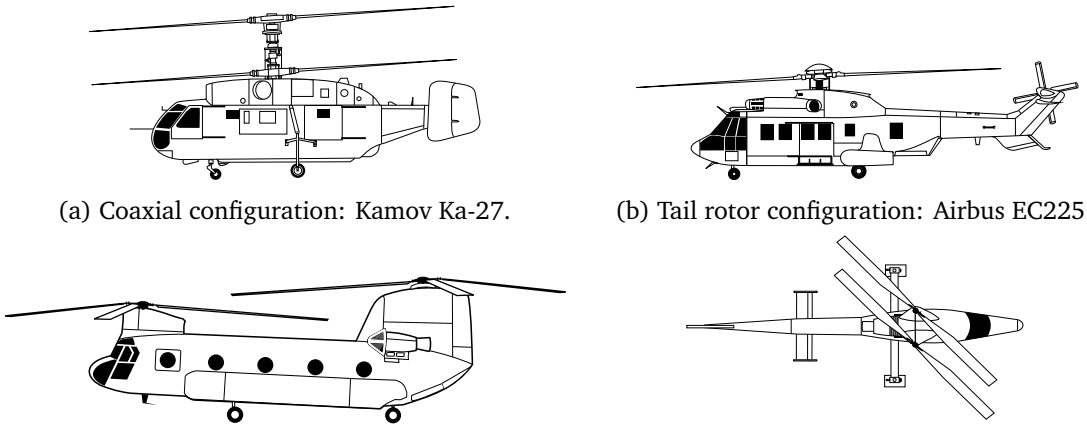


Figure 5.5 – Dissymetry of lift. The blade is more effective in the red area than in the blue area.

the speed of the advancing blade can attain or exceed the speed of sound, the aerodynamic capacity of the blade then radically change. On the other hand, if the blade has to flap highly to compensate, the angle of attack of the retreating blade can exceed the critical angle of attack. Then, the retreating blade do not produce lift anymore. This is known as retreating blade stall and is one of the main limiting factors to the speed of rotating wing aircrafts and especially helicopters[8].

Juan de la Cierva's first successful autogyro, the C.4 resembled the aircrafts of that time. It was built on the fuselage of a contemporary monoplane with a tractor propeller on which a free-spinning rotor was mounted. Autogyros were developed as a means for airplanes to fly at very low speed. The roll and the pitch of autogyros are controlled by rotor cyclic-tilting while yaw control is ensured by a rudder. Since the main rotor is freely rotating, it does not transmit torque to the aircraft body. Thanks to its secondary propeller and to its inclined main rotor, an autogyro can travel at level flight with nearly zero pitch. However, a forward minimal speed is needed both to ensure the autorotation of the main rotor and therefore a minimal lift. This minimal speed is also needed to allow the control of the yaw. Unpowered autogyros have also been developed, under the name of rotor kites. They are usually towed into the air by cars or

[8]Autogyro and compound helicopters often include stub wings producing most of the lift. They can therefore exceed this limit



(a) Coaxial configuration: Kamov Ka-27.

(b) Tail rotor configuration: Airbus EC225.

(c) Tandem configuration: Boeing CH-47 Chinook.

(d) Synchropter configuration: Kaman K-MAX.

Figure 5.6 – Various helicopter configurations (aircrafts not to scale). Ka-27 orthographical image (excerpt) by FOX 52 is licensed under CC BY-SA 4.0, K-1200 orthographical image (excerpt) by FOX 52 is licensed under CC BY-SA 3.0, EC 225 Line Drawing (excerpt) and CH-47 Line Drawing (excerpt) by Jetijones are licensed under CC BY 3.0)

ships[9] to give them the necessary forward speed.

Soon after the development of autogyros, the idea of powering the main rotor arose. This allows the rotor to ensure lift even without forward speed. Thus the aircraft gains the ability to hover. This led in 1936 to the maiden flight of the first helicopter, the German Focke-Wulf Fw 61. It had, just as the first autogyros did, the shape of an airplane but was equipped with two horizontal counter-rotating rotors at the end of two outriggers on each side of the fuselage. Indeed, rotors and propellers create a torque due to aerodynamic effects: the sum of the aerodynamic drag on the blades is zero but the moment is important. This torque is transmitted to the aircraft body in most configurations. While airplanes and autogyros can use control surfaces – namely the rudder – since they always have a minimum forward speed, helicopters are designed to hover. Without air flow on the aircraft body, the control surfaces are ineffective and the aerodynamic torque of the rotors, when transmitted to the body, has to be compensated by an anti-torque design feature to allow the control of the aircraft over the yaw axis.

The most widespread anti-torque solution is the addition of a small tail rotor as for the Airbus EC225 illustrated in figure 5.6b. This design is well suited for small helicopters but the tail rotor is a severe weakness of these helicopters. To overcome this weakness, while preserving the overall configuration of the helicopters, solutions have been developed using a fan. This fan can either be in place of the tail rotor or inside the tail. In this last configuration, the yaw control is ensured by the air flow at the end of the tail.

Similar to the Focke-Wulf Fw 61, some helicopters made the choice of having two counter-rotating main rotors in a configuration called tandem. This configuration allows greater lift capacities, for example the CH-47 Chinook depicted in figure 5.6c. Most of today's recreational or professional Unmanned Aerial vehicles make use of this configuration and feature two (quadrotor

[9] The best known autogyro is the German Focke-Achgelis Fa 330 Bachstelze which was used as an observation platform by German U-boats during World War II.

configuration) or three (hexrotor configuration) pairs of rotors in tandem. If the two main rotors share the same rotation axis, the rotors are said to be coaxial, such as for the Kamov Ka-27 of figure 5.6a. Coaxial rotors have a complex mechanical design in comparison to tandem helicopters and complex aerodynamics due to the downwash of the upper rotor on the lower rotor. However, they are rather compact compared to their lift capacities. Some models, such as the Kaman K-MAX depicted in figure 5.6d, have intermeshing rotors. Their rotors are on two different but very close axes. That way, the mechanical design of the rotors is almost as simple as for a tandem helicopter and the design is as compact as for a coaxial helicopter. However, since the two rotors are intermeshed, the helicopter needs an accurate synchronisation mechanism of the two rotors. This unusual configuration led to giving these aircrafts the name of "synchropter". Another implemented solution is to use tip-jets. Indeed, the transmission of power by a shaft between the engine – located in the helicopter's fuselage – and the rotor places torque on the main body. In tip-jets helicopters, the rotor is self-powered. While other designs have been studied or used in prototypes, the only tip-jet helicopter to have been put in production, the French SNCASO Djinn, used a compressor to inject compressed air in the hollow blades. The air stream puts the rotor in movement when ejected at the tip nozzles.

Along with the usual "flat" rotor shape, cycloidal rotors have also been developed. These rotors resemble the shape of paddle wheels – although different in concept. These rotors are used in aircrafts called "cyclogyros" which belong to the family of helicopters. The rotors rotate around a horizontal axis and provide both thrust and lift. In order to achieve this, since the rotor has a rotational symmetric shape, the axis of rotation and the axis of the rotor are distinct. Due to the difficulty of the mechanical design, no prototype has been known to be successful until recently [Tanaka et al., 2007; Adams et al., 2013].

In order to take advantage of both the autogyro and the helicopter concepts, some prototypes – called compound helicopters – have been designed. They rely on a main rotor which can be powered during take-off and landing – otherwise autorotating – and on propellers or jets to provide forward thrust and compensate the main rotor's torque. This design allows the aircraft to both hover and be very effective in forward speed. The X3 by Eurocopter for example, reached the speed of $472\,\mathrm{km\,h^{-1}}$ in level flight on June 7, 2013.

When compared to fixed-wing aircrafts, rotary-wing aircrafts have some drawbacks. As can be seen from the data gathered in table 5.2, they are less efficient in level flight, have a lower maximum speed, are particularly noisy, have a limited range, have limited load capacity and a limited ceiling. However, rotary-wing aircrafts, especially helicopters, have some great advantages. Indeed autogyro have a low minimal speed and have low take-off and landing requirements. Helicopters on the contrary are the only heavier-than-air aircraft able to hover[10]. They are particularly agile as they can move laterally, vertically and backwards. As such, they are able to take-off from and to land on particularly small platforms and can be used in a large choice of applications.

[10] Some airplanes such as the Soviet Antonov An-2 have a very slow stall speed and can even fly backwards in mild headwind. It can't however be considered as hovering since this situation is strongly dependent on the wind conditions.

Aircraft name	max. take-off weight	max. speed	range	service ceiling
Kamov Ka-27	12 000 kg	270 km h^{-1}	980 km	5000 m
Airbus EC225	11 200 kg	275 km h^{-1}	857 km	5900 m
Boeing CH-47	22 680 kg	315 km h^{-1}	741 km	5640 m
Kaman K-MAX	5443 kg	185 km h^{-1}	495 km	8875 m
Bell-Boeing MV-22B	27 400 kg	565 km h^{-1}	1527 km	7620 m

Table 5.2 – Typical characteristics of rotary-wings and convertible aircrafts

5.1.5 Convertible airplanes and other VTOL aircrafts

By the end of World War II, most of German airfields had been bombed and were unsuitable for airplanes to take-off and land. The Germans – who had already developed the first helicopters – put the concept of powered-lift[11] together to overcome these difficulties. These aircrafts were to take-off and land vertically, thus gaining the ability of a helicopter and the name of Vertical Take-Off and Landing (**VTOL**) aircrafts, a class of aircrafts which contains both such powered-lift aircrafts and rotorcrafts as the helicopter. After taking-off, those were then thought to convert into airplanes and enjoy the advantages in speed, payload and ceiling of airplanes. The German engineers then designed projects for two different aircrafts, the Focke-Achgelis Fa 269 and the Focke-Wulf Triebflügel. The first was thought to be a conventional airplane fitted with two tilting rotors driven by a fixed engine. The latter was supposed to take off sitting on its tail and then be tilted down in level flight. It had a rather unconventional design: the fuselage was inside a propeller powered by ramjets[12].

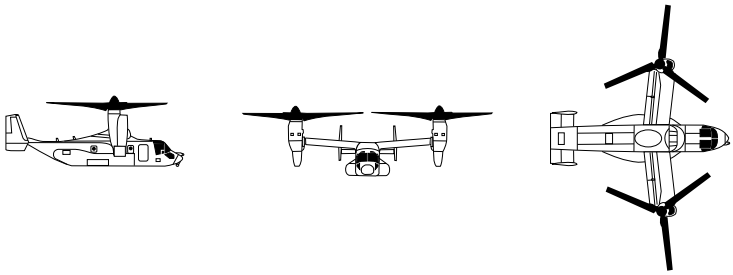
These two aircrafts laid the bases of two designs of powered-lift, the tiltrotor and the tailsitter. No tailsitter ever entered production partially due to the hazardous transition between the vertical mode and the horizontal mode. This transition has been solved and performed by the Bell Boeing V-22 Osprey. It is the only tiltrotor yet to have reached production and is extensively used by the United States Marine Corps as a replacement to the medium-lift tandem rotor helicopter Boeing Vertol CH-46 Sea Knight (a helicopter similar to the CH-47 of figure 5.6c).

As illustrated in figure 5.7a, the V-22 (here depicted in its version for the US Marine Corps MV-22) can have its rotors tilted vertically and then behaves like a helicopter. It can also have its rotors tilted horizontally and behaves then as an airplane as illustrated in figure 5.7b. Usually, for take-off and landing, the rotors are tilted vertically and the V-22 can then ascend vertically as a helicopter. However, if the payload is too important, the rotors can be tilted to smaller angles. The rotors then provide the aircraft with forward speed: the wings produce lift and the aircraft takes-off over a short distance. However, due to the large size of the rotors, the V-22 cannot take-off as a normal airplane with its rotors tilted horizontally. The aircraft is thus bound to operate the transition between the helicopter mode and the airplane mode. In helicopter mode, the aircraft is controlled using helicopter mechanics in the rotors while in airplane mode,

[11] Which is, according to the Part 1 Sec. 1.1 of the Code of Federal Regulations of the US Federal Aviation Administration, a heavier-than-air aircraft capable of vertical takeoff, vertical landing, and low speed flight that depends principally on engine-driven lift devices or engine thrust for lift during these flight regimes and on nonrotating airfoil(s) for lift during horizontal flight.

[12] Thanks to the ramjets the propeller was not linked to the fuselage. There was thus no need for anti-torque control.

the aircraft is controlled using the control surfaces. During the transition between the aircraft modes, the two control modes gradually replace each other. The rotation of the nacelle – the housing of the engine and rotor – is not used as a control.



(a) Bell Boeing V-22 Osprey: helicopter configuration.

(b) Bell Boeing V-22 Osprey: airplane configuration.

Figure 5.7 – The Bell Boeing V-22 Osprey tiltrotor. (MV-22 Osprey Line Drawing by Jetijones is licensed under CC BY 3.0)

The hybrid characteristics of convertible airplanes are nowadays considered by other operators for example to establish offshore gas and oil platforms further than the operational range of current helicopters. Due to the lower noise produced by convertible aircrafts in airplane mode, they are also studied for business connections over urban areas. Indeed, a civil aircraft based on the technologies developed for the V-22 Osprey is currently being developed by the Anglo-Italian helicopter company AgustaWestland under the name AW609.

Other powered-lift VTOL (or STOL for Short Take-Off and Landing) designs include tiltwing aircrafts which are similar to tiltrotors but where the wings tilt together with the nacelles. This is (in the drawings) more efficient in helicopter mode than the tiltrotors – since with tiltrotors, part of the thrust is lost on the wing – but has never made it beyond the prototype. Aircrafts equipped with vectorable nozzles – where the thrust of a jet engine is ducted downwards – are represented by the Harrier ground-attack aircraft family. Other designs include additional engines such as the Soviet prototype Kamov Ka-22 which had propellers in both orientations or the aircraft carrier-based Soviet Yakovlev Yak-38 which had two additional vertical turbojet engines along with vectorable nozzles.

5.2 A usual model, the quadrotor

As early as the First World War, military engineers have tried to make aircrafts fly without pilots. Charles Kettering developed in 1918 for the US Army an “aerial torpedo” prototype designed to

fly autonomously as far as 120 km. This torpedo was a simple and cheap airplane guided along a straight line by a gyroscope. The distance to the target was evaluated by the servants and converted into a number of engine cycles. When over the target, the wings were dropped and the fuselage of the torpedo fell to the ground. This idea was later used by the Germans during World War II to design their extensively used flying bomb V1 and led to the development of cruise missiles – unmanned airplanes the only purpose of which is to hit a target and explode.

The improvements in technology – e.g. in electronics, sensors, avionics, communication, engines, automatic control, global positioning (thanks to the GPS satellite constellation), batteries – led engineers to use unmanned aircrafts for other purposes than flying bombs. Unmanned aircrafts, first used by military operators, gradually replaced surveillance "spy" airplanes, bombers and are now thought to become the future of air superiority fighters. At the same time , they are being developed to be put on the civilian market where they could be used for various tasks, such as, sport cinematography, building diagnoses (e.g. thermal, structural analyses) or in a possibly near future to become delivery drones.

The very first unmanned aircrafts, the various flying torpedos, were to fly without external guidance as advanced technologies for remote control weren't available. Now, thanks to the various data channels available, most of the UAVs are remotely controlled by human operators[13]. However, autopilot systems exist that give UAVs a certain degree of autonomy and work is conducted to enhance this autonomy.

While the first UAVs used technologies stemming from the "real world" of aircrafts (e.g. similar sizes, similar engines, similar flight dynamics), the major improvements over the past years achieved in the areas of batteries, Brushless Direct Current electric motors (**BLDC** motors) – which are by far lighter than brushed motors of equivalent torque – and microelectronics led to the developments of new small scale models. Furthermore, the development of BLDC motors driving propellers generating more thrust than their own weight made it possible to build small scale electric helicopters.

Over the past years several multirotor small-scale unmanned helicopter models appeared and were put on the civilian – professional and recreational – market. Multirotor UAVs use propellers without mechanical controls – as they exist on real-scale helicopters – and are controlled simply by the rotational speed of their motors. The simplest and most widespread design includes two pairs of tandem rotors and are called quadrotors[14].

This section is dedicated to a survey of the control of quadrotors. In a first section we review existing work in the control of quadrotors and the application of these results to trajectory tracking. A second subsection is dedicated to the study of load transportation using quadrotors. In the third section we review a specific case of slung load: the inverted pendulum.

5.2.1 Design and basic model of a quadrotor helicopter

The quadrotor helicopters studied in this review are small scale helicopters equipped, as illustrated in figure 5.8, with two pairs of counter-rotating propellers with fixed direction of rotation. These rotors are usually set at the end of perpendicular arms of equal lengths. The clockwise and counter-clockwise rotors are commonly located on the same axes. One of these axes is

[13] In fact,mainly due to legal issues.

[14] The designations quadcopter and quadrotor helicopters also appear.

arbitrarily chosen to be the $x_{\mathcal{B}}$ axis, the other being $y_{\mathcal{B}}$. The quadrotor are usually designed so that the center of mass lies as close as possible to the geometrical center O of the quadrotor or at least on the $Oz_{\mathcal{B}}$ axis.

Each rotor produces a thrust and a drag moment which depend on the angular velocity. It is often considered (e.g. [Pounds et al., 2002; Bouabdallah et al., 2004; Tayebi and McGilvray, 2006; Alexis et al., 2010; Mellinger and Kumar, 2011; Mellinger et al., 2012; Thorel and d'Andréa-Novel, 2014]) that thrust and drag are proportional to the square of the angular velocity of the propeller and that they are along the rotation axis $z_{\mathcal{B}}$. However, other authors use more sophisticated models including aerodynamic effects such as the dissymmetry of lift (e.g. [Martin and Salaun, 2010; Omari et al., 2013]) induced by the quadrotor displacement.

As illustrated in figure 5.8a, the control of the quadrotor on its vertical axis is done by setting an appropriate equal thrust on all propellers. The yaw control of the quadrotor is performed based on the aerodynamic drag produced by each pair of rotors. As illustrated in figure 5.8b, each pair rotating in the same direction produces a torque rotating the frame in the opposite direction. Roll and pitch control – coupled with a necessary displacement – is achieved by balancing the thrusts of a selected pair of rotors as illustrated in figures 5.8c and 5.8d. Therefore, the quadrotor is underactuated: it cannot move along its $x_{\mathcal{B}}$ and $y_{\mathcal{B}}$ axes and a rotation around those axes implies a translation.

Quadrotors usually embed an inertial measurement unit (**IMU**), an electronic device evaluating the quadrotors orientation and acceleration. This data is used by an on-board microprocessor to set appropriate angular velocity to the rotors. Other sensors can be used such as barometers, ultra-sound range sensors, GPS, or external position tracking systems.

As expressed, for example in [Pounds et al., 2002], a simple model for the translational and rotational dynamics of a quadrotor is:

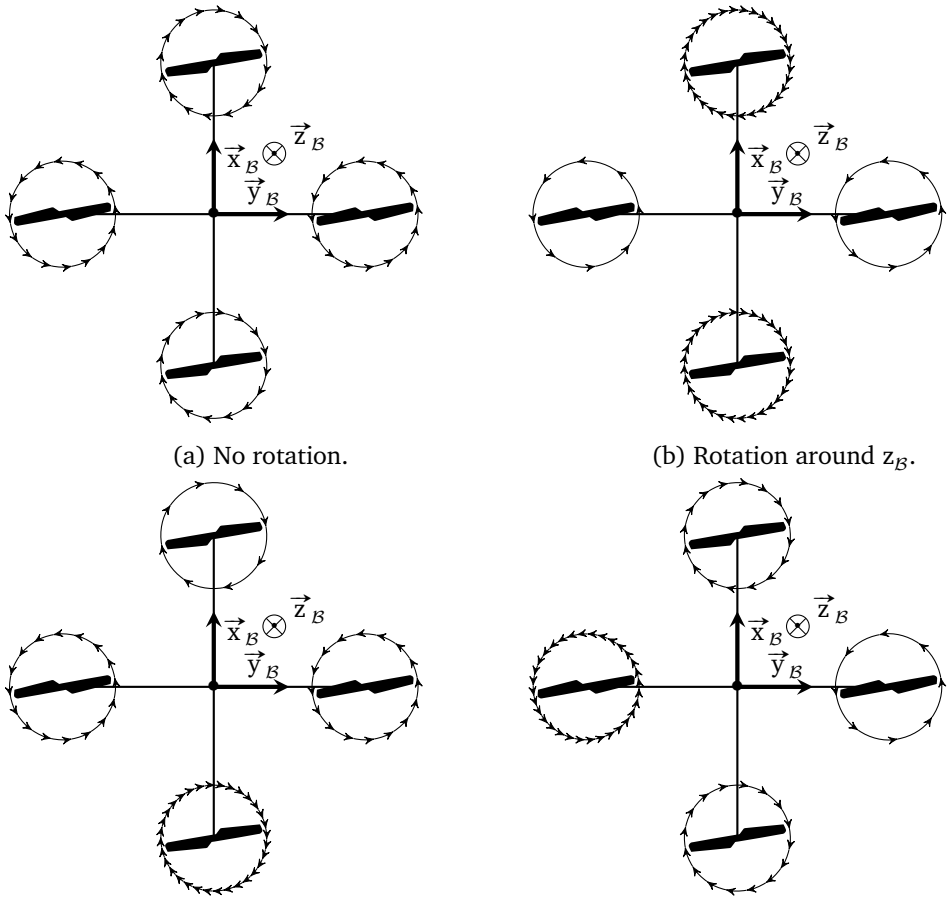
$$\begin{aligned} m\dot{\vec{v}} + m\mathbf{S}(\vec{\Omega})\vec{v} &= \vec{T} + m\mathbf{R}_{\mathcal{I}}^{\mathcal{B}}\vec{g} \\ \mathbf{J}\dot{\vec{\Omega}} + \mathbf{S}(\vec{\Omega})\mathbf{J}\vec{\Omega} &= \vec{\Gamma} \end{aligned} \quad , \tag{5.1}$$

where $\vec{T}$ is the total thrust produced by the quadrotors propellers and $\vec{\Gamma}$ the total torque produced. The tables 5.3 and 5.4 summarize the notations used throughout this work. Furthermore, the following kinematic relations hold:

$$\begin{aligned} \frac{d}{dt}\vec{r} &= \mathbf{R}_{\mathcal{B}}^{\mathcal{I}}\vec{v} \\ \frac{d}{dt}\mathbf{R}_{\mathcal{B}}^{\mathcal{I}} &= \mathbf{R}_{\mathcal{B}}^{\mathcal{I}}\mathbf{S}(\vec{\Omega}) \end{aligned} \quad . \tag{5.2}$$

The model presented in equation (5.1) contains only the forces stemming from the rotors. Indeed, on can point out that no additional term , for example accounting for the aerodynamic forces acting on the quadrotor body – notably the drag – is present. This is usually justified by the low speed of the quadrotor.

Of course, other designs exist. Some models – similar in shape to the quadrotor – use four pairs of counter-rotating coaxial rotors (see e.g. [Chamseddine et al., 2014]) instead of single rotors. The direction of the global aerodynamic drag of coaxial propellers is defined by the



(a) No rotation.

(b) Rotation around $z_{\mathcal{B}}$.

(c) Rotation around $y_{\mathcal{B}}$.

(d) Rotation around $x_{\mathcal{B}}$.

Figure 5.8 – Control of a quadrotor. Arrows' density indicates the angular velocity of the rotor.

faster rotating one and can thus be set by an appropriate choice of respective angular velocity. Therefore, while they can be controlled by similar tools, the controls have to be adapted.

5.2.2 Control and applications for a quadrotor UAV

5.2.2.1 Control and trajectory tracking

In [Pounds et al., 2002], which dates back to the beginning of quadrotor UAVs, the authors suppose that when the quadrotor is at rest, the thrust and the drag produced by each rotor is along $\vec{z}_{\mathcal{B}}$ and are proportional to the square of the angular velocity:

$$\begin{cases} \vec{T}_i = \alpha\omega_i^2 \vec{z}_{\mathcal{B}} \\ \vec{\Gamma}_i = \varkappa\omega_i^2 \vec{z}_{\mathcal{B}} \end{cases} \tag{5.3}$$

where α and $\varkappa$ are aerodynamic constants depending on the propeller. However, in case of angular displacement of the quadrotor, the authors add a gyroscopic torque applied by the rotation of the rotor to the frame:

$\mathcal{I} = \{O_{\mathcal{I}}, \vec{x}_{\mathcal{I}}, \vec{y}_{\mathcal{I}}, \vec{z}_{\mathcal{I}}\}$	An inertial reference frame
$\mathcal{B} = \{O_{\mathcal{B}}, \vec{x}_{\mathcal{B}}, \vec{y}_{\mathcal{B}}, \vec{z}_{\mathcal{B}}\}$	A North-East-Down body-fixed frame
$\vec{r} = x\vec{x}_{\mathcal{I}} + y\vec{y}_{\mathcal{I}} + z\vec{z}_{\mathcal{I}} = \{x, y, z\}$	Position of $O_{\mathcal{B}}$ expressed in the inertial reference frame
$\mathbf{R}_{\mathcal{I}}^{\mathcal{B}}$	Rotation matrix from frame $\mathcal{I}$ to frame $\mathcal{B}$
$\vec{\xi} = \{\varphi, \theta, \psi\}$	A set of Euler angles describing the quadrotor attitude
$\vec{\Omega} = \mathbf{T}\vec{\xi}$	Angular velocity of the drone expressed in $\mathcal{B}$
$\mathbf{S}(\vec{\Omega})$	The skew-symmetric matrix of Ω
$\vec{v} = \mathbf{R}_{\mathcal{I}}^{\mathcal{B}}\dot{\vec{r}}$	Velocity of the drone expressed in its body-fixed frame

Table 5.3 – Notations used for kinematic data.

m	Mass of the quadrotor
l	Length of an arm
$\mathbf{J}$	Inertia matrix of the drone
$\mathbf{J}_r$	Inertia matrix of a rotor
$\vec{T}_i$	Thrust generated by rotor i
$\vec{T}$	Total thrust generated by the quadrotor i
$\vec{\Gamma}_i$	Torque generated by rotor i
$\vec{\Gamma}$	Total torque generated by the rotors in $O_{\mathcal{B}}$
ω_i	Angular velocity of rotor i

Table 5.4 – Notations used for dynamic data.

$$\vec{\Gamma}_{g,i} = (-1)^i \omega_i \mathbf{J}_r \mathbf{S}(\vec{\Omega})\vec{z}_{\mathcal{B}}. \tag{5.4}$$

The authors also take into consideration the dissymetry of lift induced by the displacement of the drone and add the additional torque:

$$\vec{\Gamma}_{a,i} = (-1)^i \omega_i \gamma_a \mathbf{S}(\vec{v})\vec{z}_{\mathcal{B}}, \tag{5.5}$$

where γ_a is a constant based on the geometry of the propeller. The terms of equation (5.3) and these additional terms give the global thrust and torque as:

$$\begin{pmatrix} \vec{\Gamma} \\ T \end{pmatrix} = \begin{pmatrix} \sum_i \vec{\Gamma}_{g,i} + \vec{\Gamma}_{a,i} \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & l & -l \\ l & -l & 0 & 0 \\ \varkappa/\alpha & \varkappa/\alpha & -\varkappa/\alpha & -\varkappa/\alpha \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{pmatrix} \tag{5.6}$$

Nevertheless, the authors linearize the model described by equations (5.1), (5.2) and (5.6) around hovering as:

$$\begin{cases} \dot{\vec{r}} = \vec{v} \\ \dot{\vec{v}} = g \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \vec{\xi} \\ \dot{\vec{\xi}} = \vec{\Omega} \\ \dot{\vec{\Omega}} = \mathbf{J}^{-1} \begin{pmatrix} 0 & 0 & 1 & -1 \\ 1 & -1 & 0 & 0 \\ \varkappa/\alpha & \varkappa/\alpha & -\varkappa/\alpha & -\varkappa/\alpha \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{pmatrix} \end{cases}, \tag{5.7}$$

with the additional condition:

$$T_1 + T_2 + T_3 + T_4 = mg \tag{5.8}$$

meaning that the total thrust shall compensate the gravity. One can note that the model of equation (5.7) forms two cascade of four integrators for the lateral dynamics, a zero dynamics for the vertical dynamics and a second integrator for the yaw dynamics. An inner loop is suggested to control the dynamics of the propellers. The outer loop is then controlled by a simple proportional feedback loop. This however does not allow the quadrotor to fly. Indeed, the chosen DC motors, batteries and controllers are not able to lift the aircraft. It is however an interesting first attempt to control such a rotorcraft.

On a similar near-hover model, [Bouabdallah et al., 2004], showed that a PID controller for the orientation stabilization behaves well in ideal conditions. While no integral term was needed in simulations, this term was included for the real system to damp out a steady-state error. The authors also show the importance of a closed-loop speed control on each rotor.

The authors apply a Linear Quadratic control using a linearization around each attitude state. While in simulation Sage-Eisenberg method shows better results than Pearson method and in average better than the PID, the results for the experiments were in advantage of the PID controller.

Besides PID, the authors in [Bouabdallah and Siegwart, 2005] also applied backstepping and sliding mode techniques on the quadrotor to control the attitude of their quadrotor. Both the techniques could be used for position stabilisation as was demonstrated by the authors in simulations but this couldn't be performed due to the lack of an accurate position feedback. A combination between PID and backstepping, the so-called Integral Backstepping was then proposed for a slightly more complete model in [Bouabdallah and Siegwart, 2007].

A similar approach is used in [Mellinger et al., 2012]. Three simple PID controllers are suggested for attitude control, hover control and 3D path following. To obtain the hover controller – and thus the 3D path following controller – the model is linearized near hovering. This still allows the quadrotor to perform aggressive attitudes and trajectories, through iterative learning.

[Cowling et al., 2007] use the flatness of the $\{x, y, z, \psi\}$ variables – which can be deduced from equations (5.1) and (5.2) when considering the thrust and torque model of equation (5.3) – to perform trajectory tracking. As in the former, an LQR controller is used. The linearization of the system is done near hovering and thus the generated trajectories have to respect a given

numerical condition on roll and pitch for the linearization to remain approximately valid. The authors however show that such a simple model presents singularities when the drone is in free fall and thus constraining the space of reachable states.

A similar approach is chosen by [Chamseddine et al., 2012]. Trajectories based on Bézier polynomials are generated and used as a feedforward for an LQR controller designed near-hovering. To accommodate the conditions on actuator saturations and to ensure the stability of the linearized model – instability appears between twenty-five and thirty degrees in roll and pitch – the authors suggest to simply tune the final time of the trajectory.

Flatness based trajectory tracking can also be performed with other flat outputs. Indeed [Konz and Rudolph, 2013] avoid the absolute singularity introduced by any Euler angle representation – which occurs at a specific roll, pitch or yaw according to the chosen sequence, independently from the two other angles – by basing their quasi-static feedback controller on a moving reference frame. The singularity is still present but only with respect to the moving reference frame and occurs only when encountering really large tracking errors. This allows the authors to perform loopings and other aggressive maneuvers. Analogous geometric algorithms are proposed and successfully implemented for example in [Lee et al., 2010; Mellinger and Kumar, 2011].

In [Martin and Salaun, 2010], the authors revisit the usual model resulting from the model introduced in equation (5.1) with the simple thrust and torque models from equation (5.3). Indeed, in this usual model, the acceleration measured by the embedded inertial measurement unit $\vec{a} = m\ddot{\vec{r}} - \vec{g}$ is collinear to $\vec{z}_{\mathcal{B}}$. Hence the lateral and longitudinal acceleration of the drone are zero. However, when considering additional aerodynamic terms in the expression of the rotor generated thrust and drag, Newton's equation reads:

$$m\dot{\vec{v}} + m\mathbf{S}(\vec{\Omega})\vec{v} = m\mathbf{R}_{\mathcal{I}}^{\mathcal{B}}\vec{g} - l\Big(\sum \omega_i^2\Big)\vec{z}_{\mathcal{B}} - \lambda\Big(\sum \omega_i\Big)\vec{v}^{\perp}, \tag{5.9}$$

where λ is a positive constant linked to the dissymetry of lift and $\vec{v}^{\perp}$ is the projection of $\vec{v}$ onto the $\{O_{\mathcal{B}}, \vec{x}_{\mathcal{B}}, \vec{y}_{\mathcal{B}}\}$ plane. In this revised model, the lateral and longitudinal accelerations measured by the IMU are now $-\lambda\big(\sum \omega_i\big)\vec{v}^{\perp}$. The authors show that such a model better suits the real experimental measurements. The authors propose a two-level PD controller and show that the accelerometer feedback then better accounts, in systems without speed feedback, for the proportionality between longitudinal (resp. lateral) and pitch (resp. roll) angle. But, as explained by the authors, the revised model is just an improvement and does not reject the simple model which was shown to work.

However, most of the works don't take wind into account. [Alexis et al., 2010] use Constrained Finite Time Optimal (**CFTO**) control to accommodate the effect of wind gusts on the attitude dynamics of a quadrotor. The authors linearize the system in various operating points. An affine term is added to account for the wind gusts. The authors then solve an optimal control problem on a given prediction horizon respecting various input and output constraints.

The given controller seems to correctly stabilize the system under moderate wind gusts. However, the suggested scheme does not stabilize the deviation due to the wind. Furthermore, the computational effort to solve the CFTO problem is particularly intense and the authors chose to limit the prediction horizon to one time step, thus reducing the smoothness of the controller.

A better understanding of the aerodynamic effect as in [Hoffmann et al., 2011] can also be used to model both the thrust efficiency with respect to the air flow and the blade flapping.

The authors used the obtained simplified model as a feedforward for their PIDD to reject these disturbances. This feedforward seems to be relatively efficient in simulation, notably when encountering sudden changes in angle of attack. However, in actual flight, the quadrotor experiments a transient, that the authors are unable to explain. Therefore, this feedforward is not used for trajectory tracking and the aerodynamic effects are treated as other disturbances. This feedforward compensation seems promising.

In a complementary way, [Hua et al., 2009] group all the external forces in a vector $\vec{F}_e$ on which the three following assumptions are made. First, it is supposed that this vector depends only on the vehicle's linear velocity $\dot{\vec{r}}$ and on time t. It is further assumed that there exist four positive real numbers such that $\|\vec{F}_e\| \leqslant c_1 + c_2\|\dot{\vec{r}}\|^2$ and $\dot{\vec{r}}^T\vec{F}_e \leqslant c_3\|\dot{\vec{r}}\| - c_4\|\dot{\vec{r}}\|^3$. These assumptions are used to build a nonlinear controller robust to aerodynamic perturbations which greatly impacts such systems at high-velocities.

However, a good understanding of these effects is not required. As suggested in [Wang et al., 2011], model-free control (see e.g. [Fliess and Join, 2013] for an extensive introduction) can be used on bad or partially known system. For example, one can write the altitude dynamics in the inertial reference frame as:

$$m\ddot{z} = (\cos\varphi\cos\theta)u_1 + F_z, \tag{5.10}$$

where u_1 is the total thrust and F_z stands for the various badly known effects and encompasses, notably, gravity. The unknown part can therefore be estimated as:

$$\hat{F}_z = m\hat{\ddot{z}}(t_k) - (\cos\varphi\cos\theta)u_1(t_{k-1}), \tag{5.11}$$

where hatted value are estimates. This leads to the following *i-PID*[15]:

$$\begin{cases} u_1(t_k) = & u_1(t_{k-1}) + \frac{m}{c\varphi c\theta}\left(\hat{\ddot{e}} + k_1\dot{e}(t_k) + k_2 e(t_k)\right) \\ \hat{\ddot{e}}(t_k) = & \ddot{z}_r(t_k) - \hat{\ddot{z}}(t_k) \\ \dot{e}(t_k) = & \dot{z}_r(t_k) - \dot{z}(t_k) \\ e(t_k) = & z_r(t_k) - z(t_k) \end{cases}, \tag{5.12}$$

where z_r is the reference trajectory. Therefore this control can be applied to the system with little knowledge of the real system. The simulation results compare well with usual PID. Indeed, the *i-PID* is rematched at every time-step while usual PID won't adapt to inaccuracies of the system.

To overcome the difficult modelisation of aerodynamic effects on the control of the altitude, [Waslander et al., 2005] propose to use either, a sliding mode controller or reinforcement learning. Integral sliding mode control applied to the altitude dynamics makes it possible to reject perturbations as long as they are below a certain limit. On the contrary, reinforcement learning control aims to approximate the system as a stochastic Markov process using an iterative weighted linear regression based on real flight data. The model is then searched for an optimal control policy adapted to the tasks to be accomplished. Both controllers give good results. However the perturbations in sliding mode control have to remain under the foreseen boundary and the reinforcement is particularly sensitive to disturbances it was not trained for. Therefore, it needs intense computations prior to the flight to construct the model.

[15] Where the "i" stands for intelligent.

5.2.2.2 Slung load transportation

Multirotor UAVs are often used in civil applications to carry removable payloads. While some of the possible payloads, for example cameras, can be embedded in the model, it might not be desirable in other cases such as in delivery drones. There are different methods to carry the payload. Some authors consider bounding the payload directly to the UAV. For example, [Pounds et al., 2012] study the performances of PIDs to reject the perturbations implied by an additional mass. First, the payload changes the overall mass of the UAV implying a greater thrust. Second, as the center of mass of this payload does not always align with the thrust axis – parallel to the thrust, through the center of mass of the empty UAV – the mass may also considerably change the rotational dynamics of the UAV. Furthermore, the payload can even slip. [Palunko and Fierro, 2011] study the adaptive control of an unbalanced quadrotor experiencing dynamic changes in its center of gravity. This can be for example the consequence of a displacement of the payload during the flight. On the contrary [Mellinger et al., 2013] study the cooperative grasping and transportation of a rigid lightweight payload. Having all the quadrotors grasp the same object leads to additional constraints for all the quadrotors and lessens their abilities. Most of the controllers presented in the previous section would be unable – without any specific change – to handle the added constraint of an additional payload. In the following, we will however focus on slung loads which allow a greater maneuverability .

Indeed, the addition of a swinging load adds two degrees of freedom to the system while the system still has only four independent controls. This topic of research recalls former results on the control of systems such as gantry cranes (see e.g. [d'Andréa-Novel et al., 1992; Petit and Rouchon, 2001; d'Andréa-Novel and Coron, 2000]) or on the transport of slung loads by usual helicopters (see e.g. [Bisgaard et al., 2009] for an example of oscillation damping using a delayed feedback) but is nonetheless very interesting.

In [Palunko et al., 2012], the authors consider the load as a point mass spherical pendulum with a massless cable of constant length. A simple modeling allows to find the absolute acceleration of the mass. This adds a force and a moment – the suspension point is away from the center of gravity of the quadrotor – to the usual quadrotor dynamic model. To handle this dynamic change, the authors suggest to use an adaptive controller. An open-loop strategy is suggested to generate trajectories for the swinging load that damp the residual oscillations out. An initial cubic polynomial trajectory is optimized using dynamic programming. The suppression of the residual oscillations is obtained by the addition of a penalty weight to the objective function in charge of minimizing the load-displacement angles and angular velocities.

On the contrary, [Sreenath et al., 2013] builds, with a similar model for the payload, a closed loop controller that lets the load undergo large swings while being able to experience free fall during finite durations. This is motivated by the fact that the system quadrotor-load may need to go through an opening shorter than the cable's length. For this purpose, the authors use flatness. Indeed, when the load experiences free fall, it is a flat system and a quadrotor is known to be flat with position and yaw as a flat output. When the cable is taught, the authors show that the position of the load and the yaw of the quadrotor form a set of flat outputs. The system thus forms a differentially-flat hybrid system. This work was enhanced to allow cooperative transportation in [Sreenath and Kumar, 2013]. However, the choice of the flat output, being constrained by the number of available controls, makes the position of the quadrotor a consequence of the position of the load, limiting the maneuverability of the system.

The cable can also be studied as a heavy chain. This has been already partially done by [Murray, 1996] to generate certain trajectories for the free end of the cable. The stabilization of such a heavy chain was also done for cart systems for example by [Knüppel and Woittennek, 2010].

As such, [Goodarzi et al., 2013] suggest to stabilize a chain of pendulum carrying at its free end a payload. This chain is mounted on a quadrotor and stabilized using a linear controller – an LQR – from a totally-actuated cart model. The control input required by the cart to stabilize the links is used as an input for the quadrotor's controller. [Dai et al., 2014] suggest adding a retrospective cost adaptive controller to make the controller more robust to uncertainties on the payload mass.

5.2.2.3 An application: flying an inverted pendulum

The problem of inverted pendulum is an interesting nonlinear problem. It has been studied for various systems. For example, [Lozano et al., 2000] study a simple pendulum – with one degree of rotational liberty – mounted on a cart while [Lenoir et al., 1998] introduce a spherical pendulum fixed to a "juggling" robot with three rotational degrees of actuation. The authors show that the system is flat and suggest trajectories to steer the pendulum from its lower equilibrium to the upper equilibrium. During such a trajectory, a pendulum necessarily experiences free fall which is a point of singularity for the control and is over-come by time-scaling.

[Hehn and D'Andrea, 2011] had the idea of using a quadrotor and its high maneuverability to carry and control an inverted pendulum. The model is however rather simple as the dynamic impact of the pendulum on the quadrotor is neglected. Furthermore, the pendulum is supposed to be attached directly to the mass center of the quadrotor. This allows for considerable simplifications in the system's equations. The authors identify two equilibria, the first one static – the pendulum standing vertically above the quadrotor – and the second one dynamic – the quadrotor performing a circular trajectory while the pendulum also performs a circular trajectories whose characteristics are defined by the drone trajectory. The dynamics of the system are linearized around these nominal trajectories and an LQR controller is used to stabilize the system. The suggested approach seems effective but can not handle other trajectories than the two identified. Neither is it designed to recover from important errors.

While designing a geometric controller to perform cooperative load transportation with quadrotors, [Lee et al., 2013] suggested to apply the developed model – which encompasses the effect of the load on the quadrotors – and a nonlinear controller to the case of a single quadrotor. Equipped with a rigid link mounted above the quadrotor, the authors show in simulations that the suggested controller is able to track any arbitrary trajectory for the pendulum with the mass starting upright.

On the other hand, [Figueroa et al., 2014] suggest to control the pendulum using Reinforcement Learning. They use a model similar to [Hehn and D'Andrea, 2011] but decompose the tasks in two subtasks to which a reinforcement learning is applied. The first subtask is to bring the pendulum from an arbitrary state to the upright state without condition on the quadrotor state. This is performed for initial angles up to 34°. The second subtask is to maintain the pendulum upright while keeping the quadrotor as close to hovering as possible. Both subtasks are then combined to get the quadrotor to stabilize the system from an arbitrary angle. This

task is performed in usually less than 5 s. However, due to the underactuation of the system, this stabilization implies a drift on the quadrotor that can be as big as a few meter.

5.3 Tilting rotor multirotor UAVs

A good way to overcome the inherent underactuation of the quadrotor design is to use a design involving tilting rotors. Using tilting rotors instead of usual helicopter mechanics makes it possible to keep using fixed-pitch blades. This way, tilting rotors can be used to reduce the number of rotors while preserving the simple blade mechanics.

[Ryll et al., 2012] suggest to fit a quadrotor with four independent tilt axis. The rotors are at the end of arms which can be tilted with independent input rotational speeds. Since the quadrotor keeps its four independent rotor angular velocities as inputs – the system has eight independent control inputs while having only six degrees of freedom – it is overactuated. The authors use dynamic feedback linearization to track trajectories. In this process, the inputs are obtained via a system inversion which relies on a Moore-Penrose pseudo-inverse. During this inversion, the angular velocities of the rotors are minimized – while kept above a minimal speed in order to maintain flight – allegedly to minimize energy consumption. However, controlling the rotor tilts through angular velocities rather than angles requires to compute the third-order dynamical model to perform the inversion. A simulation of a flip upside-down of the quadrotor is given where the rotor axis have to be tilted 180°. Nonetheless, no experimental realization is suggested and the mechanics seems difficult to carry out.

Another example is introduced by [Hua et al., 2012]. The authors study the case of a quadrotor where the rotors remain parallel but can be globally tilted with two degrees of freedom. That way, the quadrotor is not overactuated. The authors bear in mind maximum tilt angles and the suggested strategy introduces control saturation. The Lyapunov-like approach allows decoupling of attitude and position under certain conditions. The controller objectives are split in primary and secondary objectives to handle saturation. Indeed, simulations are given where the quadrotor follows an eight-shaped trajectory while having zero attitude over the trajectory. Nonetheless, when traveling faster on this trajectory at a point where saturation is encountered, the secondary objective of maintaining zero attitude can not be achieved due to saturation. The primary objective however remains fullfilled.

In [Thorel and d'Andréa-Novel, 2014] another modified quadrotor is presented. It is equipped with ball casters and a single tilting axis tilting one of the rotor pairs. The aim of the authors is to control the quadrotor on the ground to spare energy. A trajectory tracking solution is suggested based on flatness and dynamic feedback. But the design reveals a major difficulty: it presents a singularity when the tilt axis angle passes through zero. This makes braking difficult. Thus point-stabilization becomes a challenging problem. The resemblance to the unicycle is used to suggest a time-varying point stabilization control law which requires passing through the singularity. The authors combine both control laws to stabilize the drone at the end of the flatness-based trajectories. This interesting design lacks however sufficient control to which the tilting of the transverse axis might prove an interesting enhancement.

Inspired by some real-scale successful designs, [Kendoul et al., 2005] suggested a model with only two propellers. The two propellers are mounted on two servomotors and can therefore tilt

laterally and longitudinally. However, the design is limited to only five independent controls as the lateral tilts are imposed to be exactly opposed. Indeed, the lateral tilts are used for the gyroscopic moment such a rotation creates on the pitch axis. Furthermore, the center of mass of the UAV is located below the tilting axes of the propellers. This design choice allows control of the pitch and adds a residual term to the roll moment. The resulting model is however highly coupled and presents high nonlinearities. A backstepping procedure is then suggested to perform stabilization of the aircraft and trajectory tracking.

The model was later used by [Sanchez et al., 2008] and a successful decoupling strategy was demonstrated. [Amiri et al., 2011] suggest to lessen the assumption on the lateral tilting angles. These are no more required to have opposite values. This enables a sixth independent control ideally allowing total control of the aircraft. Using a similar aircraft (with an additional small tail rotor) equipped with fixed wings for forward flight, [Fan et al., 2010] demonstrated a controlled transition – based on a backstepping approach – between rotary-wing and fixed-wing modes.

It is also possible to build a multirotor equipped with three rotors instead of an even number of rotors. Due to the odd number of rotors, the drag torques can no longer be self-compensated by pairs of counter-rotating rotors and different control strategies for the yaw have to be introduced. [Salazar-Cruz et al., 2008] presented a three-rotor aircraft where the three rotors are at the end of a "T"-shaped body. In this design, only the tail rotor can be tilted around its axis. The control of the rolling dynamics is performed by the two main rotors, the control of the pitching dynamics by the tail rotor thrust and the control of the yawing dynamics by the tail rotor inclination. This architecture simplifies the yaw control since it is controlled by the thrust of the tail rotor rather than by the differential aerodynamic drag such as on quadrotors. However, this introduces a strong translation-rotation coupling and does not improve the capacities of quadrotors. However, the three-level – attitude, altitude, horizontal – control strategy respects the tilting constraints and seems to perform well.

A similar architecture is used in [Yoo et al., 2010] and stabilization is also achieved. An additional symmetric architecture is presented where all the rotors are replaced by coaxial rotors. However, this design, by eliminating the induced drag, does not need thrust tilting anymore for the stabilization purposes presented.

A symmetric design is introduced in [Escareño et al., 2008] and called "Delta" after the shape of their aircraft. In this configuration, the three rotors are designed to tilt about their respective axis. The authors however only present a simple strategy to stabilize the attitude of the aircraft. As a consequence, the authors assume all the tilting angles are equal.

[Mohamed and Lanzon, 2012] take full advantage of this design by using the tilt angles of the three rotors as three independent controls. As such, the aircraft has six independent controls and has therefore full authority over torque and force vectoring. The authors can thus partially, as in [Hua et al., 2012] – under constraints from the actuators saturation – independently control the attitude and position of the aircraft and this with only three rotors despite the system coupling along with the nonlinearity. The suggested strategy uses input-output feedback linearization and $\mathcal{H}_\infty$ loop shaping design and performs simultaneous stabilization of all the outputs.

CHAPTER 6

The tricopter: an agile UAV

Multirotor helicopters with tilting propellers form an increasing field of research among multirotor UAVs. This growing interest is due partially to the increased agility that might be inferred by the additional controls. The previous survey introduced and described various models developed by different institutions over the world.

Under the direction of Prof. Rudolph, the team at the Chair of Systems Theory and Control Engineering (short **CSTCE**) of Saarland University (Saarbrücken, Germany), performs research, among other topics, in the field of mobile robotics and control of dynamical systems. In 2012, the development of an experimental platform with a new propeller configuration started on the occasion of the stay of a visiting engineering student of the École Centrale de Lille ([Pillu, 2012]) and resulted in a first development version of a trirotor helicopter with tilting propellers [Kastelan et al., 2015]. CSTCE's tricopter was born. This project was conducted along with the advanced development by Dipl.-Ing. Matthias Konz of his quadrotor helicopter. The design of the tricopter evolved into a second version profiting from the improvements brought by Matthias Konz' quadrotor. This second version was first assembled by myself during a stay at CSTCE in June 2014.

This chapter is dedicated to the presentation of this platform, introducing its mechanic and electronic conception and its environment. A model of the tricopter is then presented and, in the last section, a short introduction is given to finite dimensional differential flatness. This special case of the flatness theory is the main tool used throughout this part for the control of our systems.

Contents

5.1 Flight dynamics **95**
5.1.1 Basics of flight 95
5.1.1.1 Aerostats 95
5.1.1.2 Aerodynes 96
5.1.2 The roll-pitch-yaw convention and aircrafts' centers 97
5.1.3 Fixed-wing aircrafts 98
5.1.4 Rotary-wing aircrafts 100
5.1.5 Convertible airplanes and other VTOL aircrafts 104
5.2 A usual model, the quadrotor **105**
5.2.1 Design and basic model of a quadrotor helicopter 106
5.2.2 Control and applications for a quadrotor UAV 108
5.2.2.1 Control and trajectory tracking 108
5.2.2.2 Slung load transportation 113
5.2.2.3 An application: flying an inverted pendulum 114

5.3 Tilting rotor multirotor UAVs 115

6.1 Design of the tricopter

The tricopter has been built on the knowledge acquired over the years by the team of Prof. Rudolph, notably during the development of quadrotors and other propeller driven mobile robots. For example, the chair developed a ballbot, a robot balancing on a ball, stabilized and actuated by four perpendicular propellers based on the components of Matthias Konz' quadrotor. As a consequence, the tricopter shares most of its actuators, sensors and electronic software libraries with the quadrotor concurrently developed. However, it differs from this design by the addition of the nacelle tilting capability. It is, in its shape and concept, related to other designs such as the "Delta" by [Escareño et al., 2008] or the trirotor helicopter presented in [Mohamed and Lanzon, 2012].

In this section, we first introduce the overall design of the tricopter. In a second subsection, we briefly sketch the embedded electronics components of the tricopter and present in a last subsection the flight test area of CSTCE.

6.1.1 Mechanics design

The tricopter is – as inferred by its name – a trirotor helicopter. As can be seen from the top-view of the tricopter drawing presented in figure 6.1, its arms – of equal length d – are disposed 120° apart, forming an equilateral triangle. They belong to a plane that defines the horizontal plane of the tricopter which is, at rest, parallel to the ground. The center of mass of the tricopter is supposed to belong to this plane and to be centered at the intersection of the three arm axes. As was earlier introduced, the main characteristic of the tricopter is the tilting ability of its arms which confers this platform its superb agility. Indeed, the three arms are mounted on rolling bearings and can be independently tilted by servomotors around their respective longitudinal axes. In the following, we call α_i the tilt angles of the respective arms. Due to limits of the servomotors in angular range, the tilt angle is bound and the tricopter cannot invert its thrust[1].

A BLDC motor is located at the end of each arm and drives a fixed-pitch propeller. We call *nacelle*[2] the element formed by the union of a motor, its fastening and the associated propeller. The nacelles are fastened to the arms and such rotates with them. We call rotor the part formed by the rotor of the motor and the propeller, it is the rotating part of the nacelle. Each rotor rotates with angular velocity ω_i around an axis $\vec{z}_i$ (see figure 6.2 for an illustration of these rotation axes) perpendicular to the nacelle rotation axis. The overall configuration of the different axes is represented in figure 6.2. Each rotor has a dedicated sense of rotation designated as $\varepsilon_i = \pm 1$. A positive sense of rotation designates a counterclockwise rotation around the axis[3] and conversely a negative sense of rotation designates a clockwise rotation.

[1]The rotation angle would be anyway limited by the power wirings of the motors housed in the arms which oppose the rotation.

[2]By analogy with airplanes nacelles which host the engines.

[3]When facing the nacelle as in figure 6.1, the nacelle vertical axis is pointing downwards, such that clockwise becomes counterclockwise and vice-versa.

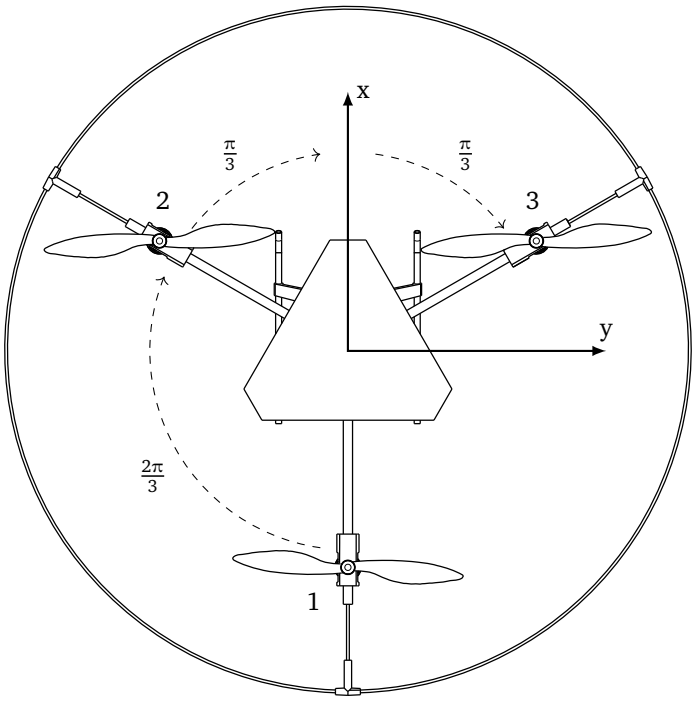


Figure 6.1 – Tricopter geometry.

Unlike multirotor configurations with an even number of rotors such as quadcopters, hexacopters or octocopters, the aerodynamic torque created by the rotating propellers is not naturally compensated by an equal, yet opposite, contribution of the counter-rotating rotors. This torque is compensated by tilting the propellers. Indeed the horizontal component of the global thrust adds a torque along the vertical axis of the body and is used to compensate the aerodynamic torque from the propellers.

In the following we will use the letter m to designate the mass of the tricopter and

$$\mathbf{J} = \begin{pmatrix} J_{xx} & J_{xy} & J_{xz} \\ J_{xy} & J_{yy} & J_{yz} \\ J_{xz} & J_{yz} & J_{zz} \end{pmatrix} \tag{6.1}$$

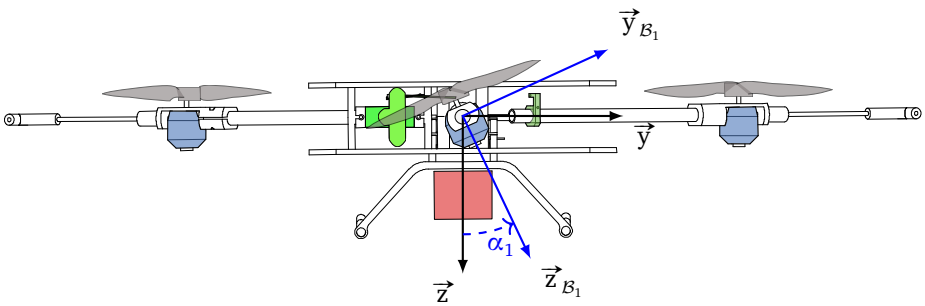


Figure 6.2 – Axial view of the first nacelle. Green elements are the servomotor and its linkage. The BLDC motors are depicted in blue. The part depicted in red is the battery.

to designate its inertia matrix. Some of the essential data of the tricopter is gathered in table 6.1. Due to imprecision in the realization of the model or, for example, when using another battery than the one the tricopter was designed with, these values may vary. The controllers have thus to be robust to these uncertainties. Notably the battery moves the center of mass downwards which is therefore significantly below the arms plane.

$m =$	$1.23\,\mathrm{kg}$
$J_{xx} =$	$1.3 \times 10^{-2}\,\mathrm{kg\,m^2}$
$J_{yy} =$	J_{xx}
$J_{zz} =$	$2.4 \times 10^{-2}\,\mathrm{kg\,m^2}$
$J_{xy} =$	$1.1 \times 10^{-6}\,\mathrm{kg\,m^2}$
$J_{xz} =$	$2.5 \times 10^{-4}\,\mathrm{kg\,m^2}$
$J_{yz} =$	$4.2 \times 10^{-8}\,\mathrm{kg\,m^2}$
$d =$	$0.245\,\mathrm{m}$
$\alpha_i \in$	$[-1.0\,;\,1.0]\,\mathrm{rad}$
$T_i \in$	$[2.0 \times 10^{-1}\,;\,8.0]\,\mathrm{N}$
$F_{xy} \in$	$[-3.0\,;\,3.0]\,\mathrm{N}$
$F_z \in$	$[0\,;\,18.0]\,\mathrm{N}$
$T_{xy} \in$	$[-4.0 \times 10^{-1}\,;\,4.0 \times 10^{-1}]\,\mathrm{N\,m}$
$T_z \in$	$[-2.0 \times 10^{-1}\,;\,2.0 \times 10^{-1}]\,\mathrm{N\,m}$
$[\varepsilon_1, \varepsilon_2, \varepsilon_3] =$	$[-1, +1, +1]$
$k_T =$	$1.42 \times 10^{-5}\,\mathrm{N\,s^2\,rad^{-2}}$
$\sigma =$	$3 \times 10^{-2}\,\mathrm{m}$

Table 6.1 – Tricopter physical parameters.

The tricopter body is mounted on a landing gear. It has proved its utility and efficiency as a kinetic energy absorber on the occasion of several less controlled and harsh landings and has kept marks of these rough events. A 750 mm large security ring is mounted in the arms plane. Its utility is twofold. On the one hand it plays the part of – poor – protection to operators and to the propellers during lateral translations such as during ground rolling phases. On the other hand it is particularly useful to materialize the spatial extension of the propellers, which are almost invisible to the naked eye when rotating. The final assembled tricopter can be seen in figure 6.3.

6.1.2 Electronics design

We can summarize the various tasks of the electronics as follows:

Computing the controls: This is done by the brain of the tricopter, a 32 bit microcontroller by Atmel. This microcontroller, namely the AT32UC3C2512C, has floating point operation capability unlike several other less powerfull microcontroller. It can work at a frequency of up to 66 MHz. This microcontroller also manages all the other functions. It is mounted on a printed circuit board (in short **PCB**) developed at the CTSCE which interfaces the central electronics to all the other components of the tricopter.

Figure 6.3 – The tricopter (courtesy of D. Kastelan).

Communication with the ground: Either to receive data from the base computer (e.g. position measures, reference trajectories) or to send data (e.g. to register or plot online parameters), the tricopter uses a ZigBee module. ZigBee modules are wireless modules designed for low energy consumption communications. In the used configuration, it works on the 2.4 GHz radio band which allows a communication range of about ten meters with a theoretical data rate of about 250 Kibit s^{-1}. In our application, we were decided to send to the ground 193 bit of data every 50 ms while the ground was sending 58 bit of data every 30 ms. The risk of packet mixing prevented us to increase the data rate.

Communication with the remote control: Either to fly the tricopter per hand or to dispose of an emergency switch when flying autonomously, the tricopter embeds a receiver connected over serial to the microcontroller and attached to a seven channels Futaba remote control.

Motor control: The three BLDC motors are each driven by an open-loop controller, on-board a control PCB developed and sold by the German UAV company Mikrokopter. The microcontroller periodically sends through an I^2C link reference speeds to the three controllers. This control is however open-loop and a closed-loop controller board is therefore currently being developed at CSTCE to replace it in the future.

Orientation and acceleration sensing: The tricopter is embedded with an inertial measurement unit (in short **IMU**) which gives access to estimations of the attitude of the tricopter, of angular rates and acceleration. However some of the data presents flaws. Notably the estimation of the heading, which is given with respect to the magnetic north, drifts over the time and can't be reliably used. However the IMU gives updated estimation values at each controller loop (every 5 ms) and is therefore faster than the motion capture platform. A filter is thus embedded to make the most of the high data-rate of the IMU and the accuracy of the motion capture platform.

Tilting the arms: The three servomotors are controlled by embedded controllers. The main microcontroller simply sends the reference value over a digital output as a pulse-width modulated signal (in short **PWM**). This reference value is encoded into 10 bit and allows a precision on the order of the tenth of a degree. The dynamics of the servomotors is neglected in the following but is currently under investigation at CSTCE.

Providing efficient power supply: As the tension of the battery varies with time and is not by default at the level needed by the various components, the tricopter is embedded with a voltage regulator PCB developed at CSTCE.

The dynamics of the tricopter is particularly fast and needs fast responding electronics and controls. Therefore, the microcontroller runs no operating system to ensure its real-time character. The microcontroller runs only the binaries of the developed code the execution time of which can be accurately evaluated. The code is written in C but the team at CSTCE ported some C++ libraries, notably Eigen – a library for linear algebra – to be used on the microcontroller.

6.1.3 The motion capture platform

A room at CSTCE is dedicated to flight experiments. It is roughly three times five meters and has four motion tracking cameras by Vicon fixed at each corner and spanning over almost the entire flight space.

Motion tracking systems has been developed first for use in the film industry for special effect. They were first used to capture the motion of actors to overcome the difficulty of giving a natural walk to the computer animation of a human. They work as follows. The body which has to be tracked is fitted with several markers: small spheres reflective to the infrared. These spheres define a solid body with position and orientation. The cameras illuminate the markers at a constant rate with infrared LED. Using several cameras, it is possible to convert these snapshots to the current orientation and position of the solid body.

As illustrated in figure 6.4, the precision of this tracking is of the order of the tenth of a millimeter and of the thousandth of a radian (about 0.06°). It has the advantage of experiencing no drift over time and to give an absolute measure with respect to the room. In our configuration, we were able to perform this evaluation and send it to the tricopter over ZigBee at about 30 Hz. However, this whole process induces a delay between the acquisition of the real position and the transmission to the tricopter of as much as 65 ms. This can be seen by comparing the angular data acquired by the IMU to the angular data acquired by the motion tracking system as has been done in figure 6.5. This delay is significant and creates difficulties when flying aggressive trajectories. To overcome this problem, a filter is included in the tricopter that performs fusion of the angular velocity measured by the IMU and of the position and orientation measured by the motion tracking system. This fusion algorithm allows to compensate the delay in attitude and position in the motion tracking system data. Furthermore in the frequent case of a motion tracking system packet loss, the algorithm interpolates the data.

6.2 Mechanical model of the tricopter

6.2.1 Formalism and assumptions

To develop a dynamic model of the tricopter, we have to identify the various forces and moments the tricopter is subject to. The first two obvious forces are gravity and the thrusts of the propellers. All the other forces and notably gyroscopic effect induced by the rotation of the rotors and servomotors and aerodynamic forces such as air drag, constant wind, wind gusts and turbulences

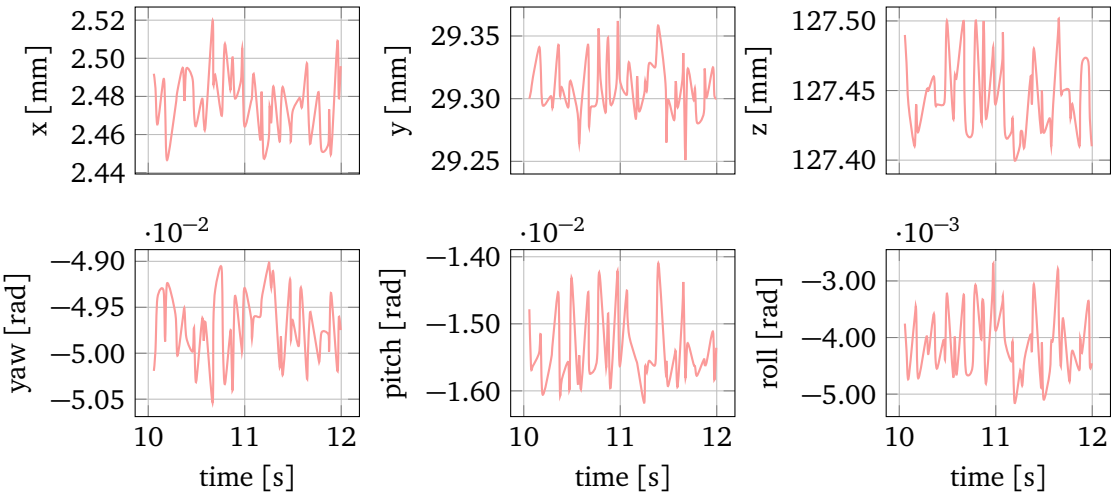


Figure 6.4 – Example of position and attitude estimate of a still laying object as seen by the motion tracking system.

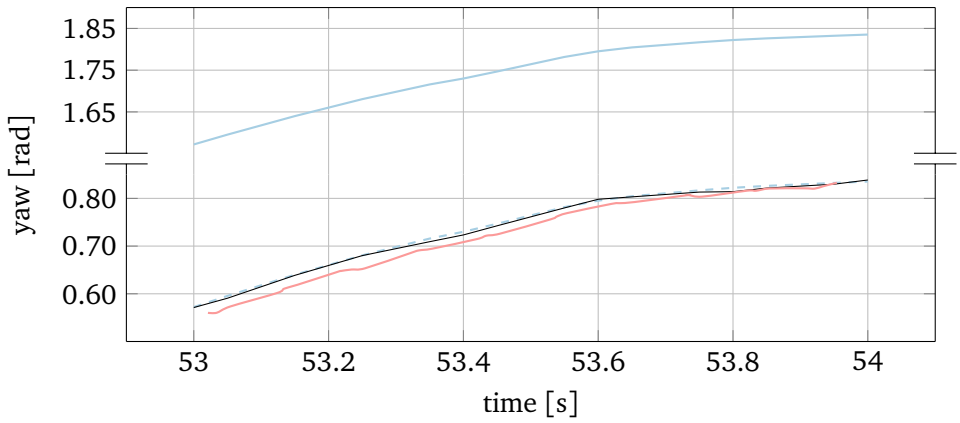


Figure 6.5 – Delay of the motion tracking system. The blue solid line represents the yaw measured by the IMU, the red line represents the yaw measured by the motion tracking system while the black line represents the filtered yaw. The blue dashed line represents the data measured by the IMU corrected by the yaw offset.

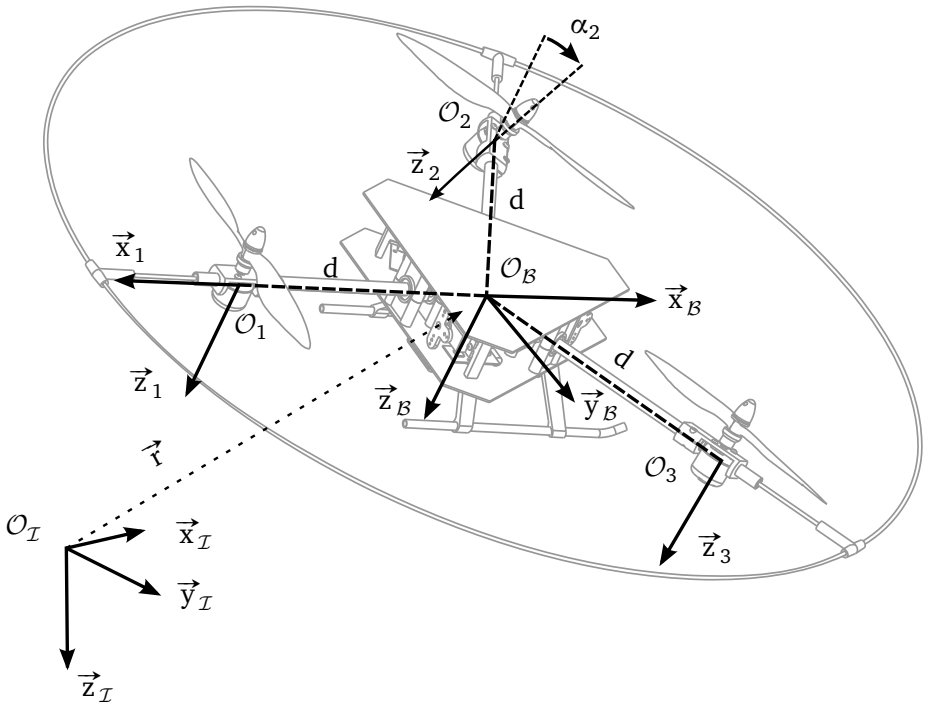


Figure 6.6 – Tricopter geometry and reference frames [Kastelan et al., 2015].

will be neglected. This is a common hypothesis (see e.g. section 5.2 and the references therein) and will help greatly in establishing a model simple enough to be controlled.

In the following, the thrust created by propeller i is written $\vec{T}_i$ while the aerodynamic torque is called $\vec{\Gamma}_i$. A common model for these two values is:

$$\vec{T}_i = -k_T \omega_i^2 \vec{z}_i, \qquad \vec{\Gamma}_i = \varepsilon_i \sigma \vec{T}_i \tag{6.2}$$

The rather unusual negative thrust is due to the choice of a downward-pointing vertical axis. This model holds rather well in ideal conditions, that is without wind or displacement of the propellers and far enough from the ground. The k_T and σ parameters are aerodynamic constants depending mainly on the propeller geometry. This model has been verified experimentally at CSTCE and the parameters for the propellers used on the tricopter have been experimentally identified. The values used in the following may be found in table 6.1 on page 120.

To establish the model of the tricopter, we adopt the following formalism. We use six-dimensional pseudo-vectors called tensors to describe generalized forces. This formalism is sometimes described as the "screw theory", the pseudo-vector is then called a *screw*. The first three components of a screw represent the force, and thus a translation, while the three later represent the moment, that is a rotation. Screws depend on the point they are evaluated at and the reference frame they are expressed in. In this formalism real forces are such that there exists a point where the three later components of the pseudo-vector are null. Pure moments on the contrary always have null first components. Euler-Newton equations describe the motion of solid bodies and are best expressed using such screws. This is the choice made to establish the equations of motion of the tricopter.

The various points and reference frames used thereafter are illustrated in figure 6.6 and the

notations are gathered in table 6.2. The attitude of the tricopter is described by the rotation matrix $\mathbf{R}_{\mathcal{I}}^{\mathcal{B}}$ which sends a vector u expressed in the reference frame $\mathcal{I}$ into the reference frame $\mathcal{B}$. Since $\mathcal{I} = (\mathcal{O}_{\mathcal{I}}, \vec{x}_{\mathcal{I}}, \vec{y}_{\mathcal{I}}, \vec{z}_{\mathcal{I}})$ and $\mathcal{B} = (\mathcal{O}_{\mathcal{B}}, \vec{x}_{\mathcal{B}}, \vec{y}_{\mathcal{B}}, \vec{z}_{\mathcal{B}})$ are both chosen right handed, we have the following properties:

$$\mathbf{R}_{\mathcal{B}}^{\mathcal{I}} = \mathbf{R}_{\mathcal{I}}^{\mathcal{B}^{\mathrm{T}}}, \qquad \mathbf{R}_{\mathcal{I}}^{\mathcal{B}}\mathbf{R}_{\mathcal{B}}^{\mathcal{I}} = \mathbf{I}, \qquad \det \mathbf{R}_{\mathcal{I}}^{\mathcal{B}} = 1 \tag{6.3}$$

The angular velocity $\vec{\Omega}$ of the tricopter is defined by:

$$\vec{\Omega} = \left(\mathbf{R}_{\mathcal{I}}^{\mathcal{B}}\dot{\mathbf{R}}_{\mathcal{B}}^{\mathcal{I}}\right)^{\vee} \tag{6.4}$$

where the *vee* operator $(\cdot)^{\vee}$ is the inverse of the skew operator $\mathbf{S}(\cdot) : \vec{u} \mapsto \mathbf{S}(\vec{u})$. The skew-matrix of a vector $\vec{u} = (u_1, u_2, u_3)^{\mathrm{T}}$ is the matrix:

$$\mathbf{S}(\vec{u}) = \begin{pmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{pmatrix} \tag{6.5}$$

It is an equivalent to the cross-product $\times$ such that for two vectors $\vec{u}$ and $\vec{v}$, we have the relation:

$$\mathbf{S}(\vec{u})\vec{v} = \vec{u} \times \vec{v} \tag{6.6}$$

The *vee* operator is hence defined as:

$$(\mathbf{S}(\vec{u}))^{\vee} = \vec{u} \tag{6.7}$$

We have then the following equalities:

$$\mathbf{R}_{\mathcal{I}}^{\mathcal{B}}\dot{\mathbf{R}}_{\mathcal{B}}^{\mathcal{I}} = \mathbf{S}(\vec{\Omega}), \qquad \dot{\mathbf{R}}_{\mathcal{B}}^{\mathcal{I}} = \mathbf{R}_{\mathcal{B}}^{\mathcal{I}}\mathbf{S}(\vec{\Omega}) \tag{6.8}$$

We define the velocity $\vec{v}$ of the tricopter as:

$$\vec{v} = \mathbf{R}_{\mathcal{I}}^{\mathcal{B}}\dot{\vec{r}} \tag{6.9}$$

Newton-Euler equations make use of the acceleration. Using the previous definitions, differentiating the velocity according to time leads to:

$$\mathbf{R}_{\mathcal{I}}^{\mathcal{B}}\ddot{\vec{r}} = \dot{\vec{v}} + \mathbf{S}(\vec{\Omega})\vec{v} \tag{6.10}$$

Finally, each nacelle has a body-fixed frame $\mathcal{B}_i = (\mathcal{O}_i, \vec{x}_i, \vec{y}_i, \vec{z}_i)$ that is centered in the center of mass of the nacelle. We assume that this center of mass is located in the plane $\mathcal{O}_i\vec{x}_{\mathcal{B}}\vec{y}_{\mathcal{B}}$ and is therefore located on the arm rotation axis. The angle between $\vec{z}_i$ and $\vec{z}_{\mathcal{B}}$ is called α_i. As stated in the hypotheses, we neglect gyroscopic effects. Hence, the model depends only on the angles α_i and not on their derivatives. Furthermore, we assume that the inertia matrix of the tricopter $\mathbf{J}$ does not depend on the angles α_i.

$\mathcal{I} = (\mathcal{O}_\mathcal{I}, \vec{x}_\mathcal{I}, \vec{y}_\mathcal{I}, \vec{z}_\mathcal{I})$	Inertial reference frame
$\mathcal{B} = (\mathcal{O}_\mathcal{B}, \vec{x}_\mathcal{B}, \vec{y}_\mathcal{B}, \vec{z}_\mathcal{B})$	Tricopter body-fixed frame
$\mathbf{R}_\mathcal{I}^\mathcal{B}$	Rotation matrix from frame $\mathcal{I}$ to frame $\mathcal{B}$
$\vec{r} = (x, y, z)^T$	Position of the tricopter expressed in the inertial reference frame
$\vec{\Omega}$	Angular velocity of the tricopter with respect to the inertial reference frame
$\vec{v}$	Velocity of the tricopter expressed in its body-fixed frame
$\mathcal{B}_i = (\mathcal{O}_i, \vec{x}_i, \vec{y}_i, \vec{z}_i)$	Frame fixed to the i-th nacelle
$\vec{T}_i$	Thrust produced by the i-th propeller
$\vec{\Gamma}_i$	Aerodynamic moment produced by the i-th propeller
ω_i	Angular velocity of propeller i with respect to the i-th nacelle

Table 6.2 – Dynamic and kinematic notations used in the model of the tricopter. Bold upright values are matrix, slanted values with upper arrows are vectors.

6.2.2 Screws acting on the tricopter

Following the chosen formalism, we want to find the global screw applied to the tricopter in its center of mass. Following our simplifying assumptions, this screw is the sum of the gravity screw and of the screws from the three propellers. We will apply Euler-Newton equations in the tricopter body-fixed frame and in its center of mass and thus need an expression in this frame and point. We may first express the screw of the gravity at the center of mass $\mathcal{O}_\mathcal{B}$ in the reference frame $\mathcal{I}$ as:

$$_{\mathcal{O}_\mathcal{B}}\{\mathcal{G}\}_\mathcal{I} = {}_{\mathcal{O}_\mathcal{B}}\left\{ \begin{array}{c} m\vec{g} \\ 0 \end{array} \right\}_\mathcal{I} = {}_{\mathcal{O}_\mathcal{B}}\left\{ \begin{array}{c} mg\vec{z}_\mathcal{I} \\ 0 \end{array} \right\}_\mathcal{I} \tag{6.11}$$

This screw can be expressed in the body-fixed frame $\mathcal{B}$ and reads then:

$$_{\mathcal{O}_\mathcal{B}}\{\mathcal{G}\}_\mathcal{B} = {}_{\mathcal{O}_\mathcal{B}}\left\{ \begin{array}{c} mg\mathbf{R}_\mathcal{I}^\mathcal{B}\vec{z}_\mathcal{I} \\ 0 \end{array} \right\}_\mathcal{B} \tag{6.12}$$

The screw of a propeller at point $\mathcal{O}_i$ in the reference frame $\mathcal{B}_i$ of the i-th nacelle is:

$$_{\mathcal{O}_i}\{\Theta_i\}_{\mathcal{B}_i} = {}_{\mathcal{O}_i}\left\{ \begin{array}{c} \vec{T}_i \\ \vec{\Gamma}_i \end{array} \right\}_{\mathcal{B}_i} = {}_{\mathcal{O}_i}\left\{ \begin{array}{c} -T_i\vec{z}_i \\ -\varepsilon_i\sigma T_i\vec{z}_i \end{array} \right\}_{\mathcal{B}_i} \tag{6.13}$$

In the reference frame $\mathcal{B}_i$, the center of mass of the tricopter is located at $-d\vec{x}_i$. We neglect the possible offset in $\vec{z}_i$ as it creates no additional torque. By the definition of the torque, known in French under the name of "Varignon's rule", this screw expressed in the center of mass $\mathcal{O}_\mathcal{B}$ reads:

$$_{\mathcal{O}_\mathcal{B}}\{\Theta_i\}_{\mathcal{B}_i} = {}_{\mathcal{O}_\mathcal{B}}\left\{ \begin{array}{c} -T_i\vec{z}_i \\ -\varepsilon_i\sigma T_i\vec{z}_i - dT_i\mathbf{S}(\vec{x}_i)\vec{z}_i \end{array} \right\}_{\mathcal{B}_i} \tag{6.14}$$

We have $\mathbf{S}(\vec{x}_i)\vec{z}_i = -\vec{y}_i$, the previous screw then reads:

$$_{\mathcal{O}_\mathcal{B}}\{\Theta_i\}_{\mathcal{B}_i} = {}_{\mathcal{O}_\mathcal{B}}\left\{ \begin{array}{c} -T_i\vec{z}_i \\ -\varepsilon_i\sigma T_i\vec{z}_i + dT_i\vec{y}_i \end{array} \right\}_{\mathcal{B}_i} \tag{6.15}$$

This screw can then be expressed in the body-fixed frame $\mathcal{B}$ as:

$$ {}_{\mathcal{O}_i}\{\Theta_i\}_{\mathcal{B}} = \left\{ \begin{array}{c} -T_i \mathbf{R}_{\mathcal{B}_i}^{\mathcal{B}} \vec{z}_i \\ -\varepsilon_i \sigma T_i \mathbf{R}_{\mathcal{B}_i}^{\mathcal{B}} \vec{z}_i + d T_i \mathbf{R}_{\mathcal{B}_i}^{\mathcal{B}} \vec{y}_i \end{array} \right\}_{\mathcal{B}}^{\mathcal{O}_i} \tag{6.16} $$

The rotation $\mathbf{R}_{\mathcal{B}}^{\mathcal{B}_i}$ is obtained as a rotation of angle α_i around axis $\vec{x}_i$. The vector $\vec{x}_i$ might be expressed as the rotation by an angle $\beta_i \in \{-\pi, -\frac{\pi}{3}, \frac{\pi}{3}\}$ around $\vec{z}_{\mathcal{B}}$ of the basis vector $\vec{x}_{\mathcal{B}}$. It is thus the product of two rotations and reads (see [Diebel, 2006] for the adopted rotation formalism):

$$ \mathbf{R}_{\mathcal{B}_i}^{\mathcal{B}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha_i & \sin\alpha_i \\ 0 & -\sin\alpha_i & \cos\alpha_i \end{pmatrix} \begin{pmatrix} \cos\beta_i & \sin\beta_i & 0 \\ -\sin\beta_i & \cos\beta_i & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{6.17} $$

Hence, the three rotation matrices read respectively:

$$ \mathbf{R}_{\mathcal{B}_1}^{\mathcal{B}} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -\cos\alpha_1 & \sin\alpha_1 \\ 0 & \sin\alpha_1 & \cos\alpha_1 \end{pmatrix} \tag{6.18} $$

together with:

$$ \mathbf{R}_{\mathcal{B}_2}^{\mathcal{B}} = \begin{pmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2}\cos\alpha_2 & -\frac{\sqrt{3}}{2}\sin\alpha_2 \\ -\frac{\sqrt{3}}{2} & \frac{1}{2}\cos\alpha_2 & -\frac{1}{2}\sin\alpha_2 \\ 0 & \sin\alpha_2 & \cos\alpha_2 \end{pmatrix} \tag{6.19} $$

and:

$$ \mathbf{R}_{\mathcal{B}_3}^{\mathcal{B}} = \begin{pmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2}\cos\alpha_2 & \frac{\sqrt{3}}{2}\sin\alpha_2 \\ \frac{\sqrt{3}}{2} & \frac{1}{2}\cos\alpha_2 & -\frac{1}{2}\sin\alpha_2 \\ 0 & \sin\alpha_2 & \cos\alpha_2 \end{pmatrix} \tag{6.20} $$

To get a simple expression of the global screw, we introduce the vector:

$$ \vec{f}_a = (T_1\cos\alpha_1, T_2\cos\alpha_2, T_3\cos\alpha_3, T_1\sin\alpha_1, T_2\sin\alpha_2, T_3\sin\alpha_3)^T \tag{6.21} $$

The six controls can be obtained from this vector as:

$$ \begin{cases} \alpha_i = & \operatorname{atan2}(f_a^{i+3}, f_a^i) \\ T_i = & \sqrt{(f_a^i)^2 + (f_a^{i+3})^2} \end{cases} \tag{6.22} $$

where the function atan2 is prolonged in $(0,0)$ and is therefore defined as:

$$ \operatorname{atan2}(x,y) = \begin{cases} \arctan\frac{y}{x} & x > 0 \\ \arctan\frac{y}{x} + \pi & y \geqslant 0, x < 0 \\ \arctan\frac{y}{x} - \pi & y < 0, x < 0 \\ +\frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ 0 & y = 0, x = 0 \end{cases} \tag{6.23} $$

Using the control vector $\vec{f}_a$, the three screws can be written ${}_{\mathcal{O}_1}\{\Theta_1\}_{\mathcal{B}} = \mathbf{W}_i \vec{f}_a$ where the matrices $\mathbf{W}_i$ read:

$$\mathbf{W}_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -d & 0 & 0 & -\varepsilon_1\sigma & 0 & 0 \\ -\varepsilon_1\sigma & 0 & 0 & d & 0 & 0 \end{pmatrix} \tag{6.24}$$

together with:

$$\mathbf{W}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & \frac{\sqrt{3}}{2} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{3}}{2}d & 0 & 0 & \frac{\sqrt{3}}{2}\varepsilon_2\sigma & 0 \\ 0 & \frac{1}{2}d & 0 & 0 & \frac{1}{2}\varepsilon_2\sigma & 0 \\ 0 & -\varepsilon_2\sigma & 0 & 0 & d & 0 \end{pmatrix} \tag{6.25}$$

and:

$$\mathbf{W}_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & -\frac{\sqrt{3}}{2} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -\frac{\sqrt{3}}{2}d & 0 & 0 & -\frac{\sqrt{3}}{2}\varepsilon_3\sigma \\ 0 & 0 & \frac{1}{2}d & 0 & 0 & \frac{1}{2}\varepsilon_3\sigma \\ 0 & 0 & -\varepsilon_3\sigma & 0 & 0 & d \end{pmatrix} \tag{6.26}$$

The total screw applied by the propellers to the tricopter is then:

$${}_{\mathcal{O}_B}\{\Theta\}_{\mathcal{B}} = \sum_{i=1}^{3} {}_{\mathcal{O}_B}\{\Theta_i\}_{\mathcal{B}} = \mathbf{W}\vec{f}_a \tag{6.27}$$

where $\mathbf{W} = \mathbf{W}_1 + \mathbf{W}_2 + \mathbf{W}_3$ is the matrix:

$$\mathbf{W} = \begin{pmatrix} 0 & 0 & 0 & 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ 0 & 0 & 0 & -1 & \frac{1}{2} & \frac{1}{2} \\ -1 & -1 & -1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{3}}{2}d & -\frac{\sqrt{3}}{2}d & 0 & \frac{\sqrt{3}}{2}\varepsilon_2\sigma & -\frac{\sqrt{3}}{2}\varepsilon_3\sigma \\ -d & \frac{1}{2}d & \frac{1}{2}d & -\varepsilon_1\sigma & \frac{1}{2}\varepsilon_2\sigma & \frac{1}{2}\varepsilon_3\sigma \\ -\varepsilon_1\sigma & -\varepsilon_2\sigma & -\varepsilon_3\sigma & d & d & d \end{pmatrix} \tag{6.28}$$

Different versions of this matrix were introduced in [Kastelan et al., 2015; Servais et al., 2015a,b]. They differ in signs due to the various conventions adopted in these works. A formal evaluation of the determinant of this matrix reads:

$$\det \mathbf{W} = -\frac{3}{4}d\left(2(\varepsilon_1^2 + \varepsilon_3^2 + \varepsilon_2^2 - \varepsilon_1\varepsilon_2 - \varepsilon_1\varepsilon_3 - \varepsilon_2\varepsilon_3)\sigma^2 + 9d^2\right) \tag{6.29}$$

which can be written:

$$\det \mathbf{W} = -\frac{3}{4}d(9d^2 + 2\mu\sigma^2) \tag{6.30}$$

together with the following coefficient μ:

$$\mu = \sum_i \varepsilon_i^2 - \sum_{i>j} \varepsilon_i \varepsilon_j \tag{6.31}$$

Which is found by identification to take the two only possible values:

$$\mu = \begin{cases} 0 & \text{if } \varepsilon_1 = \varepsilon_2 = \varepsilon_3 \\ 4 & \text{otherwise} \end{cases} \tag{6.32}$$

In both cases, the matrix $\mathbf{W}$ is invertible. It is therefore possible to find a control vector $\vec{f}_a$ for any desired screw. In the case $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = 1$, the matrix reads:

$$\mathbf{W}(1,1,1) = \begin{pmatrix} 0 & 0 & 0 & 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ 0 & 0 & 0 & -1 & \frac{1}{2} & \frac{1}{2} \\ -1 & -1 & -1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{3}d}{2} & -\frac{\sqrt{3}d}{2} & 0 & \frac{\sqrt{3}\sigma}{2} & -\frac{\sqrt{3}\sigma}{2} \\ -d & \frac{d}{2} & \frac{d}{2} & -\sigma & \frac{\sigma}{2} & \frac{\sigma}{2} \\ -\sigma & -\sigma & -\sigma & d & d & d \end{pmatrix} \tag{6.33}$$

Due to the simpler expression of the determinant of the matrix in this case, the invert of this matrix has a simpler expression and reads:

$$\frac{3}{2} d \mathbf{W}^{-1}(1,1,1) = \begin{pmatrix} 0 & \sigma & -\frac{d}{2} & 0 & -1 & 0 \\ -\frac{\sqrt{3}\sigma}{2} & -\frac{\sigma}{2} & -\frac{d}{2} & \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 \\ \frac{\sqrt{3}\sigma}{2} & -\frac{\sigma}{2} & -\frac{d}{2} & -\frac{\sqrt{3}}{2} & \frac{1}{2} & 0 \\ 0 & -d & -\frac{\sigma}{2} & 0 & 0 & \frac{1}{2} \\ \frac{\sqrt{3}d}{2} & \frac{d}{2} & -\frac{\sigma}{2} & 0 & 0 & \frac{1}{2} \\ -\frac{\sqrt{3}d}{2} & \frac{d}{2} & -\frac{\sigma}{2} & 0 & 0 & \frac{1}{2} \end{pmatrix} \tag{6.34}$$

The configuration of the tricopter can be obtained as the product of the two matrics:

$$\mathbf{W}(-1,1,1) = \mathbf{Q}(-1,1,1)\mathbf{W}(1,1,1) \tag{6.35}$$

With:

$$\mathbf{Q}(-1,1,1) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -\frac{4\sigma}{3} & -\frac{2\sigma^2}{3d} & 0 & 1 & \frac{2\sigma}{3d} \\ 0 & \frac{4\sigma^2}{3d} & -\frac{2\sigma}{3} & 0 & -\frac{4\sigma}{3d} & 1 \end{pmatrix} \tag{6.36}$$

Which invert reads:

$$\mathbf{Q}^{-1}(-1,1,1) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{4\sigma\left(3d^2+2\sigma^2\right)}{9d^2+8\sigma^2} & \frac{2d\sigma^2}{9d^2+8\sigma^2} & 0 & \frac{9d^2}{9d^2+8\sigma^2} & -\frac{6d\sigma}{9d^2+8\sigma^2} \\ 0 & \frac{4d\sigma^2}{9d^2+8\sigma^2} & \frac{2\sigma\left(3d^2+4\sigma^2\right)}{9d^2+8\sigma^2} & 0 & \frac{12d\sigma}{9d^2+8\sigma^2} & \frac{9d^2}{9d^2+8\sigma^2} \end{pmatrix} \tag{6.37}$$

The invert of the matrix $\mathbf{W}(-1,1,1)$ in the chosen configuration can thus be obtained as the product of two simple inverts as:

$$\mathbf{W}^{-1}(-1,1,1) = \mathbf{W}^{-1}(1,1,1)\mathbf{Q}^{-1}(-1,1,1) \tag{6.38}$$

And reads:

$$\mathbf{W}^{-1}(-1,1,1) = \begin{pmatrix} 0 & -\frac{3d^2\sigma}{9d^2+8\sigma^2} & -\frac{3d\left(3d^2+4\sigma^2\right)}{18d^2+16\sigma^2} & 0 & -\frac{9d^2}{9d^2+8\sigma^2} & \frac{6d\sigma}{9d^2+8\sigma^2} \\ -\frac{\sqrt{3}\sigma}{2} & \frac{3d^2\sigma}{18d^2+16\sigma^2} & -\frac{3d\left(3d^2+2\sigma^2\right)}{18d^2+16\sigma^2} & \frac{\sqrt{3}}{2} & \frac{9d^2}{18d^2+16\sigma^2} & -\frac{3d\sigma}{9d^2+8\sigma^2} \\ \frac{\sqrt{3}\sigma}{2} & \frac{3d^2\sigma}{18d^2+16\sigma^2} & -\frac{3d\left(3d^2+2\sigma^2\right)}{18d^2+16\sigma^2} & -\frac{\sqrt{3}}{2} & \frac{9d^2}{18d^2+16\sigma^2} & -\frac{3d\sigma}{9d^2+8\sigma^2} \\ 0 & -\frac{3d\left(3d^2+2\sigma^2\right)}{9d^2+8\sigma^2} & -\frac{3d^2\sigma}{18d^2+16\sigma^2} & 0 & \frac{6d\sigma}{9d^2+8\sigma^2} & \frac{9d^2}{18d^2+16\sigma^2} \\ \frac{\sqrt{3}d}{2} & \frac{3d\left(3d^2+4\sigma^2\right)}{18d^2+16\sigma^2} & -\frac{3d^2\sigma}{18d^2+16\sigma^2} & 0 & \frac{6d\sigma}{9d^2+8\sigma^2} & \frac{9d^2}{18d^2+16\sigma^2} \\ -\frac{\sqrt{3}d}{2} & \frac{3d\left(3d^2+4\sigma^2\right)}{18d^2+16\sigma^2} & -\frac{3d^2\sigma}{18d^2+16\sigma^2} & 0 & \frac{6d\sigma}{9d^2+8\sigma^2} & \frac{9d^2}{18d^2+16\sigma^2} \end{pmatrix} \tag{6.39}$$

Using the physical values given in table 6.1 on page 120, an approximation of the matrix $\mathbf{W}^{-1}(-1,1,1)$ can be given as (for clarity, units are not given but are all SI units):

$$\mathbf{W}^{-1} = \begin{pmatrix} 0 & -0.0269 & -0.336 & 0 & -2.69 & 0.219 \\ -0.0707 & 0.0134 & -0.332 & 2.36 & 1.34 & -0.11 \\ 0.0707 & 0.0134 & -0.332 & -2.36 & 1.34 & -0.11 \\ 0 & -0.664 & -0.0134 & 0 & 0.219 & 1.34 \\ 0.577 & 0.336 & -0.0134 & 0 & 0.219 & 1.34 \\ -0.577 & 0.336 & -0.0134 & 0 & 0.219 & 1.34 \end{pmatrix} \tag{6.40}$$

6.2.3 Equations of motion

Applying Newton-Euler equations to the flying drone in the body-fixed frame reads:

$${}_{\mathcal{O}_\mathcal{B}}\{\Theta\}_\mathcal{B} + {}_{\mathcal{O}_\mathcal{B}}\{\mathcal{G}\}_\mathcal{B} = \begin{pmatrix} m\mathbf{I} & 0 \\ 0 & \mathbf{J} \end{pmatrix} \begin{pmatrix} \mathbf{R}_\mathcal{I}^\mathcal{B}\ddot{\vec{r}} \\ \dot{\vec{\Omega}} \end{pmatrix} + \begin{pmatrix} 0 \\ \mathbf{S}(\vec{\Omega})\mathbf{J}\vec{\Omega} \end{pmatrix} \tag{6.41}$$

where $\mathbf{I}$ is the identity matrix. Using the notations introduced previously, we may write this as

$$\begin{bmatrix} m\mathbf{R}_\mathcal{I}^\mathcal{B}\left(\ddot{\vec{r}} - \vec{g}\right) \\ \mathbf{J}\dot{\vec{\Omega}} + \mathbf{S}(\vec{\Omega})\mathbf{J}\vec{\Omega} \end{bmatrix} = \vec{f}_b = \mathbf{W}\vec{f}_a, \tag{6.42}$$

These six independent equations totally define the motion of the tricopter. In the following section, we introduce the concept of differential flatness. This concept will apply, throughout the different applications of the tricopter, to the various models developed thereafter.

6.3 Introduction to flatness based control

Flatness of dynamic systems was introduced in the early nineties by Fliess et al. and introductory examples were given in [Rouchon et al., 1993; Fliess et al., 1995; Murray et al., 1995]. The formal definition of a differentially flat system was given as:

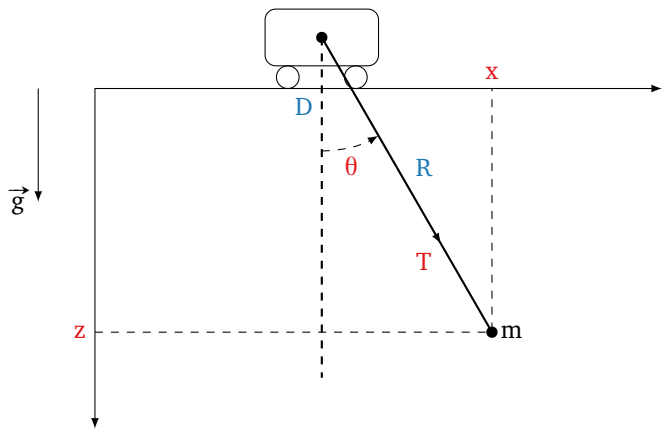


Figure 6.7 – The introductory example of flat systems: the crane [Fliess et al., 1995]. Inputs are written in blue, unknowns in red.

Definition 6.3.1 (Differentially flat systems [Fliess et al., 1995]). The system defined by $\dot{w} = f(w, u), w \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, is said to be flat if there exist a function $h : \mathbb{R}^n \times (\mathbb{R}^m)^{n+1} \to \mathbb{R}^m$, and functions $f : (\mathbb{R}^m)^r \to \mathbb{R}^n$ and $g : (\mathbb{R}^m)^{r+1} \to \mathbb{R}^m$ such that:

$$\begin{aligned} v &= h(w, u, \dot{u}, \ldots, u^{(r)}) \\ w &= f(v, \dot{v}, \ldots, v^{(r-1)}) \\ u &= g(v, \dot{v}, \ldots, v^{(r)}). \end{aligned}$$

then v is called a *flat output* of the system.

A simple form of this definition is given by Murray et al. in [Murray et al., 1995]:

> Roughly speaking, a system is differentially flat if we can find a set of outputs (equal in number to the number of inputs) such that all states and inputs can be determined from these outputs without integration.

This is best explained by the following example. A simple crane, depicted in figure 6.7 consists of a trolley controlled in position D along the x axis and of a massless rigid rope of length R. These two functions of time are the inputs of our system. The system has four unknowns, the position (x, z) of the load of mass m, the angle θ between the rope and the vertical axis and the tension T in the rope. The whole system is supposed to move in the xz plane only. The equations of motion of the system read:

$$\begin{cases} m\ddot{x} &= -T\sin\theta \\ m\ddot{z} &= -T\cos\theta + mg \\ x &= R\sin\theta + D \\ z &= R\cos\theta \end{cases} \tag{6.43}$$

We may rewrite these equations to express the two inputs with two of the unknows, x and z, which are the flat output of the system. The two inputs read:

$$\begin{cases} D &= x - \frac{\ddot{x}z}{\ddot{z}-g} \\ R^2 &= z^2 + \left(\frac{\ddot{x}z}{\ddot{z}-g}\right)^2 \end{cases} \tag{6.44}$$

The two remaining unknowns in equation (6.43) may be obtained from the two first equations:

$$\begin{cases} T^2 &= m^2(\ddot{x}^2 + (\ddot{z}-g)^2) \\ \theta &= \operatorname{atan2}(\ddot{x}, \ddot{z}-g) \end{cases} \tag{6.45}$$

Equation (6.44) gives immediately an open-loop controller to follow any arbitrary choice of the flat output. Therefore, showing a system is flat is a convenient way to start building a controller. This approach has been extensively used for finite differential systems (see for example chapter 5 and the numerous references therein). It was also extended to infinite differential systems as was shown in the first part of this thesis. Differential flatness will be used in the following of this work thanks to the following result:

Proposition 6.3.2. *The tricopter modeled in equation (6.42) is flat with flat output* $(\vec{r}, \mathbf{R}_{\mathcal{I}}^{\mathcal{B}})$.

Proof. Based on equation (6.30), for each of the eight choices of configuration $(\varepsilon_i)_{1,2,3} \in \{-1;+1\}^3$, the matrix $\mathbf{W}$ is invertible. Equation (6.22) defines inputs for all possible values of the control vector $\vec{f}_a$ and thus for all possible value of $\vec{f}_b$ which is defined by the only flat outputs and their derivatives. □

CHAPTER 7

Various applications of the tricopter

In this chapter we present various applications of the tricopter. In a first section, we present the simulator and the control approach that have been developed for the tricopter. To test the real model, we study then the tricopter on the ground as a rolling platform. This first experiment allows to test the complete platform and the controller. Next, autonomous flight control of the tricopter is studied and trajectory tracking experiments are presented. Finally, the application of the tricopter to pendulum load transportation is studied.

Contents

6.1 Design of the tricopter **118**
6.1.1 Mechanics design 118
6.1.2 Electronics design 120
6.1.3 The motion capture platform 122
6.2 Mechanical model of the tricopter **122**
6.2.1 Formalism and assumptions 122
6.2.2 Screws acting on the tricopter 126
6.2.3 Equations of motion 130
6.3 Introduction to flatness based control **130**

7.1 Simulation platform, test trajectory and control approach

7.1.1 The simulation platform

We developed during the thesis a simulator of the tricopter. This simulator was used for both application cases, flight and drive. The platform consists of various modules written in C code managed by an interface written in Python. This interface allows various online data plotting and is also interfaced with a flight-simulator, flightgear. A computer model was graphically animated and imported into flightgear and can be used as an output to the simulator platform.

The various components of the simulation platform can be summarized as:

The controller: This module consists of the controller running on the tricopter. It is written in C and performs at time t the evaluation of the controls as given by the closed-loop controller presented in equation (7.30).

Figure 7.1 – The graphical model of the tricopter developed for use in FlightGear Flight Simulator.

The integrator: This module performs the integration between two time steps of the model given in equation (6.42). It is written in C and uses the numerical integration procedures of the GNU Scientific Library. This module currently uses the "Explicit embedded Runge-Kutta (2, 3) method" stepping function which accuracy proved to be sufficient while being computationally less intensive in comparison to more precise stepping functions such as the available "Explicit embedded Runge-Kutta Prince-Dormand (8, 9) method" which is the default in several other ODE solvers[1].

The trajectory generator: This module returns online the reference positions, speeds and accelerations at time t. It is written in C and can use either a trajectory chosen among a set of predefined trajectories or be replaced by inline data.

The bindings: All the above mentioned C modules have python bindings. These bindings define for every accessible functions a python procedure calling the C function and converting the arguments.

The interface: This is the core of the simulation platform. It interfaces the three modules together. Noise and delay are added by this interface which also allows to save the data, plot it using the python matplotlib library or send it to an external viewer.

The simulation viewer: An optional viewer has been added to the simulation platform. It uses the free and open-source FlightGear Flight Simulator as an external viewer. The module performs online conversion and sending over a network connexion of the model data to FlightGear's server. A graphical model of the tricopter, illustrated in figure 7.1, has been created, animated, and imported into FlightGear. Using FlightGear's capacities, we have been able to simulate and represent two tricopters[2] flying in formation.

[1]Prince-Dormand methods are the default in Matlab, GNU Octave and Simulink.
[2]Unfortunately, the architecture of the multi-player system does not allow to fly more than two aircrafts at a time.

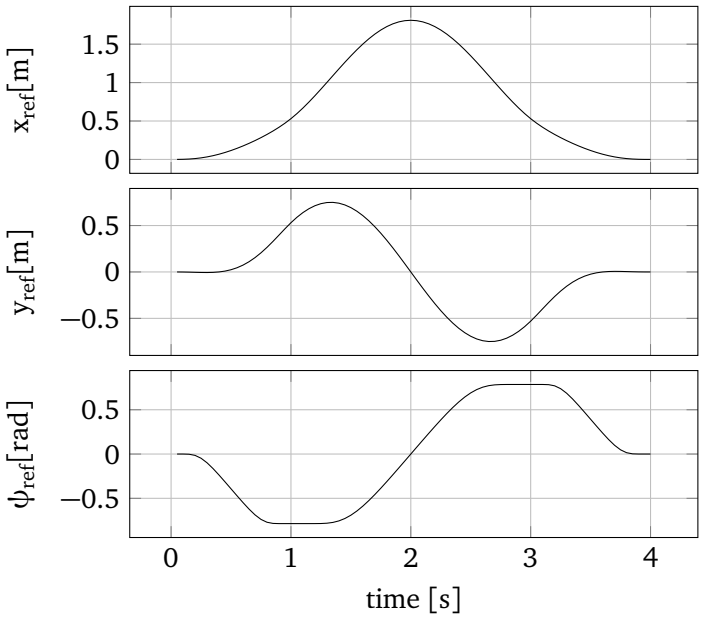


Figure 7.2 – Reference trajectories with x in the upper part, y in the middle part and ψ in the lower part with $\gamma = 1.1, \nu = 1.0$.

7.1.2 The test trajectory

To test the tracker, we constructed a trajectory inspired by the trajectories created in part I. This trajectory is from rest to rest and consists of an almost straight corner from which the tricopter starts and finishes and a portion of a circle driven at constant speed. The global trajectory, depicted in figure 7.2, resembles a drop. This offers variety in the trajectory profile with a simple parametrization. The suggested trajectory, defined with respect to the normalized time $\tau = t/T$, is:

$$x_{ref} = \begin{cases} \tau^3 P(\tau) & 0 \leqslant \tau < 1 \\ d\sqrt{2} - d\sin\left(\frac{3\pi}{4}\tau\right) & 1 \leqslant \tau < 3 \\ (4-\tau)^3 P(4-\tau) & 3 \leqslant \tau < 4 \end{cases} \tag{7.1}$$

$$y_{ref} = \begin{cases} \tau^3 Q(\tau) & 0 \leqslant \tau < 1 \\ d\sqrt{2} - d\cos\left(\frac{3\pi}{4}\tau\right) & 1 \leqslant \tau < 3 \\ -(4-\tau)^3 Q(4-\tau) & 3 \leqslant \tau < 4 \end{cases} \tag{7.2}$$

where P and Q are polynomials of second order chosen to ensure a $\mathcal{C}^2$ transition between the starting point at $(0,0)$ and the arc. Thanks to the τ^3 factor (resp. $(4-\tau)^3$), initial and final speeds and accelerations are null. These trajectories are plotted in figure 7.2 while the resulting path can be seen in figure 7.3.

The reference yaw is defined as:

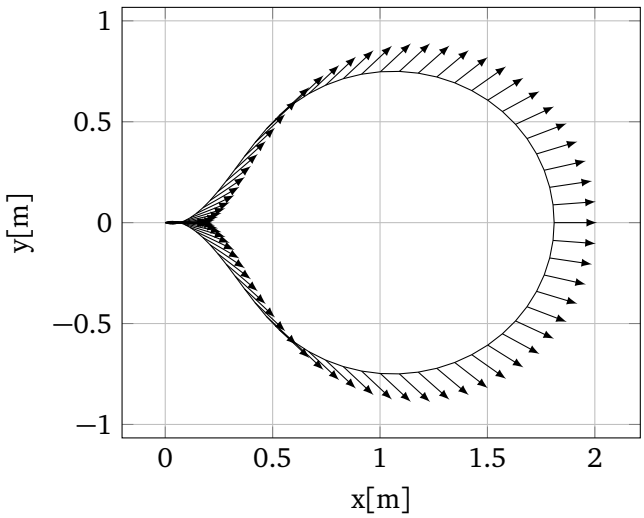


Figure 7.3 – The reference path used as a test to the tricopter. The arrows represent the reference heading of the tricopter.

$$\psi_{\text{ref}} = \begin{cases} -\frac{\pi}{4}\Phi_\gamma\left(\frac{\tau}{\nu}\right) & 0 \leqslant \tau < 1 \\ -\frac{\pi}{4}\left(2\Phi_\gamma\left(\frac{\tau-1}{2}\right)-1\right) & 1 \leqslant \tau < 3 \\ \frac{\pi}{4}\Phi_\gamma\left(\frac{4-\tau}{\nu}\right) & 3 \leqslant \tau < 4 \end{cases} \tag{7.3}$$

together with the tuning parameter $\nu \in (0,1]$ used to set the transition time and $\Phi_\gamma(t)$ the transition function

$$\Phi_\gamma(t) = \begin{cases} 0 & \text{if } t \leqslant 0 \\ 1 & \text{if } t \geqslant 1 \\ \frac{1}{2}\left(1+\tanh\left(\frac{2(2t-1)}{(4t(1-t))^\gamma}\right)\right) & \text{elsewhere} \end{cases} \tag{7.4}$$

This trajectory is plotted in the lower part of figure 7.2. The use of the $\Phi_\gamma(t)$ Gevrey function gives an example of infinitely differentiable trajectory with constant states.

7.1.3 The control approach

The model of the flying tricopter was previously established in section 6.2, equation (6.42). We showed in proposition 6.3.2 that the tricopter is flat with a possible flat output $(\vec{r}, \mathbf{R}_{\mathcal{I}}^{\mathcal{B}})$. This allows to develop the following controller.

For this purpose, we define a reference position $\vec{r}_r$ and a reference attitude $\mathbf{R}_{\mathcal{I}}^{\mathcal{R}}$. We call $(\vec{r}_r, \mathbf{R}_{\mathcal{I}}^{\mathcal{R}})$ the reference trajectory of our system. Based on this reference trajectory, we define the position error $\vec{r}_e$ and the attitude error $\mathbf{R}_e$ as:

$$\vec{r}_e = \vec{r} - \vec{r}_r \tag{7.5}$$

and:

$$\mathbf{R}_e = \mathbf{R}_{\mathcal{B}}^{\mathcal{R}} = \mathbf{R}_{\mathcal{I}}^{\mathcal{R}}\mathbf{R}_{\mathcal{B}}^{\mathcal{I}} \tag{7.6}$$

The derivation of the attitude error reads:

$$\dot{\mathbf{R}}_e = \mathbf{R}_{\mathcal{I}}^{\mathcal{R}}\dot{\mathbf{R}}_{\mathcal{B}}^{\mathcal{I}} + \dot{\mathbf{R}}_{\mathcal{I}}^{\mathcal{R}}\mathbf{R}_{\mathcal{B}}^{\mathcal{I}} \tag{7.7}$$

using the definition of the angular velocity, this reads:

$$\dot{\mathbf{R}}_e = \mathbf{R}_{\mathcal{B}}^{\mathcal{R}}\mathbf{S}(\vec{\Omega}) - \mathbf{S}(\vec{\Omega}_r)\mathbf{R}_{\mathcal{B}}^{\mathcal{R}} \tag{7.8}$$

We may then evaluate:

$$\mathbf{R}_e^T\dot{\mathbf{R}}_e = \mathbf{S}(\vec{\Omega}) - \mathbf{R}_e^T\mathbf{S}(\vec{\Omega}_r)\mathbf{R}_{\mathcal{B}}^{\mathcal{R}} \tag{7.9}$$

This finally reads:

$$\mathbf{R}_e^T\dot{\mathbf{R}}_e = \mathbf{S}(\vec{\Omega} - \mathbf{R}_e^T\vec{\Omega}_r) \tag{7.10}$$

We define the angular velocity error as:

$$\vec{\Omega}_e = \vec{\Omega} - \mathbf{R}_e^T\vec{\Omega}_r \tag{7.11}$$

or equivalently:

$$\vec{\Omega}_e = \left(\mathbf{R}_e^T\dot{\mathbf{R}}_e\right)^\vee \tag{7.12}$$

where the so-called *vee* operator is such that $(\mathbf{S}(\vec{\Omega}))^\vee = \vec{\Omega}$. Furthermore, we define the angular velocity $\vec{\Omega}_d$ as:

$$\vec{\Omega}_d = \vec{\Omega} - \vec{\Omega}_e = \mathbf{R}_e^T\vec{\Omega}_r \tag{7.13}$$

Deriving equation (7.13) with respect to time reads:

$$\dot{\vec{\Omega}}_d = \mathbf{R}_e^T\dot{\vec{\Omega}}_r - \mathbf{S}(\vec{\Omega}_e)\mathbf{R}_e^T\vec{\Omega}_r \tag{7.14}$$

or, using equation (7.11):

$$\dot{\vec{\Omega}}_d = \mathbf{R}_e^T\dot{\vec{\Omega}}_r - \mathbf{S}(\vec{\Omega})\mathbf{R}_e^T\vec{\Omega}_r \tag{7.15}$$

Based on the nonlinear attitude controller presented in [Konz and Rudolph, 2013] and applied first in [Kastelan et al., 2015] to the case of the trirotor, we suggest the following error dynamics:

$$\begin{pmatrix} \ddot{\vec{r}}_e \\ \mathbf{J}\dot{\vec{\Omega}}_e + \mathbf{S}(\vec{\Omega}_e)\mathbf{J}\vec{\Omega}_e \end{pmatrix} = \begin{pmatrix} -\mathbf{K}_d^t\dot{\vec{r}}_e - \mathbf{K}_p^t\vec{r}_e \\ -\mathbf{K}_d^a\vec{\Omega}_e - \left(\mathbf{S}\left(\mathbf{K}_p^a\mathbf{R}_e\right)\right)^\vee \end{pmatrix} \tag{7.16}$$

Where the matrices $\mathbf{K}_p^t$, $\mathbf{K}_d^t$, $\mathbf{K}_p^a$ and $\mathbf{K}_d^a$ are gain matrices and the skew operator $\mathbf{S}$ is defined for matrices as $\mathbf{S}(\mathbf{R}) = \frac{1}{2}\left(\mathbf{R} - \mathbf{R}^T\right)$. For appropriate choices of the gain matrices, these error dynamics can be shown to converge exponentially to zero for a large set of initial conditions. For the translational gain matrices $\mathbf{K}_p^t$ and $\mathbf{K}_d^t$, we simply choose a time constants λ_1 and λ_2 and set $\mathbf{K}_p^t = \mathrm{Diag}\{\lambda_1^2, \lambda_1^2, \lambda_2^2\}$ and $\mathbf{K}_d^t = \mathrm{Diag}\{2\lambda_1, 2\lambda_1, 2\lambda_2\}$. For the rotational gain matrices, [Bullo and Murray, 1999] show in lemma 9 that with $\mathbf{K}_d^a$ a positive definite matrix and $\mathbf{K}_p^a$ a symmetric matrix with eigenvalues $\{k_1, k_2, k_3\}$ such that $k_i + k_j > 0$ for $i \neq j$, there exist a Lyapunov function exponentially converging to zero for a certain set of initial conditions

increasing with $\min_{i\neq j}(k_i + k_j)$. Using, the expressions of equation (7.13), the error dynamics of equation (7.16) reads:

$$\begin{pmatrix} \ddot{\vec{r}} \\ \mathbf{J}\dot{\vec{\Omega}} + \mathbf{S}(\vec{\Omega})\mathbf{J}\vec{\Omega} \end{pmatrix} = \begin{pmatrix} \ddot{\vec{r}}_r - \mathbf{K}_d^t \dot{\vec{r}}_e - \mathbf{K}_p^t \vec{r}_e \\ \mathbf{J}\dot{\vec{\Omega}}_d + \mathbf{S}(\vec{\Omega})\mathbf{J}\vec{\Omega} - \mathbf{S}(\vec{\Omega}_e)\mathbf{J}\vec{\Omega}_e - \mathbf{K}_d^a \vec{\Omega}_e - \left(\mathbf{S}\left(\mathbf{K}_p^a \mathbf{R}_e\right)\right)^\vee \end{pmatrix} \tag{7.17}$$

And, based on equation (6.42), we suggest the following controller:

$$\vec{f}_b = \begin{pmatrix} m\mathbf{R}_{\mathcal{I}}^{\mathcal{B}}\left(\ddot{\vec{r}}_r - \mathbf{K}_d^t \dot{\vec{r}}_e - \mathbf{K}_p^t \vec{r}_e - \vec{g}\right) \\ \mathbf{J}\dot{\vec{\Omega}}_d + \mathbf{S}(\vec{\Omega})\mathbf{J}\vec{\Omega} - \mathbf{S}(\vec{\Omega}_e)\mathbf{J}\vec{\Omega}_e - \mathbf{K}_d^a \vec{\Omega}_e - \left(\mathbf{S}\left(\mathbf{K}_p^a \mathbf{R}_e\right)\right)^\vee \end{pmatrix} \tag{7.18}$$

In the following, we present three applications of this controller. The first application allows to test the controller rolling on the ground, the second application is the main dedication of this controller: the flying tricopter. The third and last application is the transport of a pendulum with the flying tricopter.

7.2 The rolling tricopter

7.2.1 Motivations and mechanical modifications

Sylvain Thorel in his thesis [Thorel, 2014] studied the ground control of a quadrotor mounted on ball casters having a tilting axis. This UAV was thought for indoor exploration. The ground control was justified by the fact that rolling requires less thrust upwards (ideally, no thrust at all) and that the autonomy might then been improved. However, the control of this rolling UAV proved itself to be particularly difficult as it wasn't able to brake and was thus difficult to stabilize at a chosen point. Furthermore the trajectories had to be built with a particular care since the resulting rolling robot was not totally actuated and resembled a hovercraft.

The tricopter, on the contrary, is totally actuated. We therefore suggested to build on the idea introduced in Sylvain Thorel's work. On the first hand, we benefit from the advantages enumerated for the quadrotor, namely the greater autonomy for indoor exploration. On the other hand, it appears to be an efficient and safe way to test the tricopter together with the trajectory generation and tracking before flying. Indeed, we will focus on level flights and will track a trajectory that had been tested on the ground.

To allow the tricopter to roll, we designed and manufactured a rolling gear that is presented in figure 7.4. It was designed to be easily removable to allow for easy reconfiguration between the rolling mode and the flying mode. Indeed, whereas the addition in mass and inertia around the vertical axis as given in table 7.1 is acceptable when rolling, even without changing the model, the change in inertia matrix seemed to be too important to fly safely and efficiently.

This second mode for the tricopter induces however a limitation. With the landing gear and the rolling gear, the arms of the tricopter are at about 110 mm height while the radius of the propellers is 130 mm. In order to avoid the propellers to hit the ground, the tilt angles are limited to $\alpha_i \in [-0.5\ ;\ 0.5]$ rad.

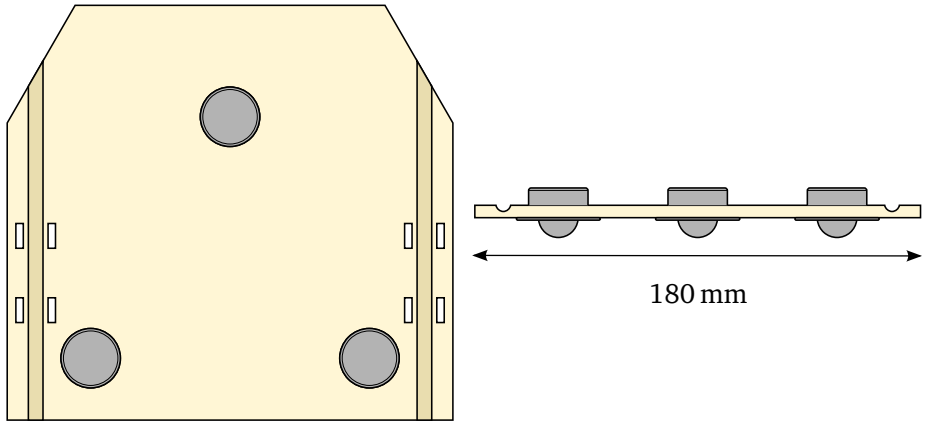


Figure 7.4 – The rolling base used by the tricopter. The yellow plate was carved out from PVC to host the tricopter landing gear. The balls (grey) are omni-directional ball casters.

$m^b =$	$0.22\,\mathrm{kg}$
$J^b_{zz} =$	$1 \times 10^{-3}\,\mathrm{kg\,m^2}$

Table 7.1 – Parameters of the rolling gear.

Finally, the complete actuation of our model allows a much simpler control strategy. The quadrotor with tilting propellers of Sylvain Thorel was proved to be flat but presented a singularity, time varying control was thus used in [Thorel and d'Andréa-Novel, 2014]. In the following, we show a simple way relying only on flatness-based control to perform trajectory tracking and point stabilization for a rolling UAV.

7.2.2 Controlling the rolling tricopter, a flatness-based control approach

The tricopter is rolling on the ground. As a consequence the pitch and roll angles as well as altitude of the tricopter remain constant to zero. We may then adapt the model presented in equation (6.42) by taking these constraints into account and updating the tricopter physical parameters. We define the yaw angle ψ as the angle between the heading of the tricopter $\vec{x}_{\mathcal{B}}$ and the reference "north" $\vec{x}_{\mathcal{I}}$. We distinguish between positive and negative angles using the atan2 function and the definition:

$$\psi = \mathrm{atan2}(\|\mathbf{S}(\vec{x}_{\mathcal{B}})\vec{x}_{\mathcal{I}}\|, \vec{x}_{\mathcal{B}} \cdot \vec{x}_{\mathcal{I}}) \tag{7.19}$$

where $\|\cdot\|$ is the euclidean norm. The attitude of the tricopter may be represented with the use of the rotation matrix:

$$\mathbf{R}^{\mathcal{B}}_{\mathcal{I}} = \begin{pmatrix} c_\psi & s_\psi & 0 \\ -s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{7.20}$$

The velocity of the tricopter then reads:

$$\vec{v} = \begin{pmatrix} c_\psi \dot{x} + s_\psi \dot{y} \\ -s_\psi \dot{x} + c_\psi \dot{y} \\ 0 \end{pmatrix} \tag{7.21}$$

and its angular velocity is obtained as:

$$\vec{\Omega} = (\mathbf{R}_{\mathcal{I}}^{\mathcal{B}} \dot{\mathbf{R}}_{\mathcal{B}}^{\mathcal{I}})^\vee = \begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix} \tag{7.22}$$

Finally, the model of the rolling tricopter can be simplified, using the appropriate matrix to select only the wished dynamics, to:

$$\mathbf{R}_{\mathcal{I}}^{\mathcal{B}} \begin{pmatrix} m'\ddot{x} \\ m'\ddot{y} \\ J'_{zz}\ddot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \vec{f}_b \tag{7.23}$$

where $\vec{f}_b = (f_b^i, 1 \leqslant i \leqslant 6) = \mathbf{W}\vec{f}_a$, together with, $m' = m + m^b = 1.45\,\mathrm{kg}$, the total mass of the

$m' =$	$1.23\,\mathrm{kg}$
$J'_{zz} =$	$2.4 \times 10^{-2}\,\mathrm{kg\,m^2}$
$\alpha_i \in$	$[-0.5\ ;\ 0.5]\,\mathrm{rad}$
$T_z =$	$4.3\,\mathrm{N}$

Table 7.2 – Updated physical parameters of the rolling Tricopter.

tricopter and $J'_{zz} = J_{zz} + J^b_{zz} = 2.5 \times 10^{-2}\,\mathrm{kg\,m^2}$, the total inertia of the tricopter about its vertical axis. The coefficient $f_b^i, 3 \leqslant i \leqslant 5$ may be chosen freely. These three degrees of freedom could be optimized in order to reduce the energy consumption as was done by the authors of [Ryll et al., 2012]. On the contrary, we simply choose to set the three remaining degrees of freedom to constant values:

$$\begin{cases} f_b^3 = -T_z \\ f_b^4 = 0 \\ f_b^5 = 0 \end{cases} \tag{7.24}$$

The coefficient T_z corresponds to the vertical component of the total thrust of the tricopter. T_z is chosen to be positive, the negative sign is due to the choice of a downward pointing vertical axis. This force is composed by summing the vertical thrusts of the three propellers. Due to various limitations, this coefficient cannot be set to zero. Furthermore, it should not make the tricopter take-off. An appropriate value will be determined by simple open-loop simulations in the following section. As a consequence of these choices, the generalized-body-forces vector $\vec{f}_b$ can be put into the form:

$$\vec{f}_b = \begin{pmatrix} m'(c_\psi \ddot{x} + s_\psi \ddot{y}) \\ m'(-s_\psi \ddot{x} + c_\psi \ddot{y}) \\ -T_z \\ 0 \\ 0 \\ J'_{zz}\ddot{\psi} \end{pmatrix} \tag{7.25}$$

or:

$$\vec{f}_b = \begin{pmatrix} \mathbf{R}_{\mathcal{I}}^{\mathcal{B}} & 0 \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ -T_z \\ 0 \\ 0 \\ J'_{zz}\ddot{\psi} \end{pmatrix} \tag{7.26}$$

The corresponding control vector $\vec{f}_a$ is:

$$\vec{f}_a = \mathbf{W}^{-1}\vec{f}_b \tag{7.27}$$

This six controls on the tilt angles and the propeller thrusts can then be found using equation (6.22). We therefore dispose over a simple way to construct open-loop controls to track an assigned trajectory. Indeed, for a chosen trajectory $\xi_{ref}(t)$ parametrized as:

$$\xi_{ref}(t) = \begin{pmatrix} x_{ref}(t) \\ y_{ref}(t) \\ \psi_{ref}(t) \end{pmatrix} \tag{7.28}$$

We can compute reference velocity and acceleration $\dot{\xi}_{ref}(t)$ and $\ddot{\xi}_{ref}(t)$. From $\psi_{ref}(t)$, a reference orientation matrix $\mathbf{R}_{\mathcal{I}\,ref}^{\mathcal{B}}$ can be computed. An appropriate open-loop controller can be constructed from the trajectory as:

$$\vec{f}_b = \begin{pmatrix} \mathbf{R}_{\mathcal{I}\,ref}^{\mathcal{B}} & 0 \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} m'\ddot{x}_{ref} \\ m'\ddot{y}_{ref} \\ -T_z \\ 0 \\ 0 \\ J'_{zz}\ddot{\psi}_{ref} \end{pmatrix} \tag{7.29}$$

This reference open-loop controller is however not sufficient to track the assigned trajectories. Due to various inaccuracy in the model it is necessary to use a closed-loop controller. We make the assumption that the orientation controller is fast enough so that the hypothesis $\mathbf{R}_{\mathcal{I}\,ref}^{\mathcal{B}} = \mathbf{R}_{\mathcal{I}}^{\mathcal{B}}$ is reasonable. Considering the constrained attitude of the tricopter, the controller introduced in equation (7.18) can be simplified to:

$$\vec{f}_b = \begin{pmatrix} \mathbf{R}_{\mathcal{I}}^{\mathcal{B}} & 0 \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} m'\left(\ddot{x}_{ref} + k_{1,d}(\dot{x}_{ref} - \dot{x}) + k_{1,p}(x_{ref} - x)\right) \\ m'\left(\ddot{y}_{ref} + k_{1,d}(\dot{y}_{ref} - \dot{y}) + k_{1,p}(y_{ref} - y)\right) \\ -T_z \\ 0 \\ 0 \\ J'_{zz}\left(\ddot{\psi}_{ref} + k_{2,d}(\dot{\psi}_{ref} - \dot{\psi}) + k_{2,p}(\psi_{ref} - \psi)\right) \end{pmatrix} \tag{7.30}$$

where $k_{1,p}$, $k_{1,d}$, $k_{2,p}$ and $k_{1,d}$ are properly chosen positive terms. Due to the symmetry of the system, the gains for the longitudinal and lateral dynamics are set equal. To ensure the safe operation of the tricopter, either in open-loop or in closed-loop, the force vector $\vec{f}_b$ is updated after computation. The force vector undergoes the simple saturation procedure presented in

algorithm 7.1 using the bounds given in table 6.1. This is the very same procedure as used for the flying case. As presented in algorithm 7.2, the body-forces vector $\vec{f}_b$ is then inverted to find the control vector $\vec{f}_a$. The angles and thrusts are computed using equation (6.22). The controls are afterwards checked for saturations using the values for the specific case of the rolling drone given in tables 6.1 and 7.2.

Data:
$\vec{f}_b$ ▷ body-forces vector
$\vec{f}_{max} = (F_{xy,max}, F_{xy,max}, F_{z,max}, \Gamma_{xy,max}, \Gamma_{xy,max}, \Gamma_{z,max})$ ▷ Upper bounds for forces and torques
$\vec{f}_{min} = (-F_{xy,max}, -F_{xy,max}, F_{z,min}, -\Gamma_{xy,max}, -\Gamma_{xy,max}, -\Gamma_{z,max})$ ▷ Lower bounds for forces and torques
function SATURATE BODY FORCES
  for $i \leftarrow 1, 6$ **do**
    if $f_b^i > f_{max}^i$ **then** ▷ $f_b^i, f_{max}^i, f_{min}^i$ are the i-th components of $\vec{f}_b$, $\vec{f}_{max}$, $\vec{f}_{min}$
      $f_b^i \leftarrow f_{max}^i$
    if $f_b^i < f_{min}^i$ **then**
      $f_b^i \leftarrow f_{min}^i$
  return $\vec{f}_b$

Algorithm 7.1 – Saturation of the body forces.

7.2.3 Admissible trajectories and open-loop simulations

7.2.3.1 Saturations and admissible trajectories

A first step into performing accurate simulations of our system and into performing real experiments is to design *admissible* trajectories. We define an admissible trajectory as a function $\xi_{ref} = (x_{ref}(t), y_{ref}(t), \psi_{ref}(t))^T$ such that the ideal tricopter modeled by equation (7.23) using the open-loop controller introduced in equation (7.29) follows the reference trajectory. The only obstacles appearing when simulating an ideal model hindering the tracking of the designed trajectories are the two levels of saturations. Two different parameters play a role in these saturations. The first is the aggressiveness of the trajectory, *i.e.* the longitudinal, lateral and rotational acceleration needed to follow the path in the allotted time. The aggressiveness can be reduced by increasing the allotted time. The second parameter is the chosen resulting vertical thrust T_z.

Indeed, in the controller introduced previously, it is important to choose a viable vertical thrust T_z. This component has to verify two important conditions. First, due to the possible ranges for the tilt angles and the thrusts, the tricopter always experiences a non zero vertical thrust. Second, as we do not want the tricopter to take-off, this component has to be bounded from above. These bounds read:

$$3 \cos \alpha_{max} T_{min} < T_z < m' g, \tag{7.31}$$

The value of the vertical thrust, that has to be determined with the help of the first simulations and experiments, is therefore in the range:

```
Data:
  f⃗_b, W^{-1}                                   ▷ body-forces vector
  T_min, T_max, α_max                            ▷ Upper bounds for thrusts and tilt angles
  function SATURATE ACTUATORS
      f⃗_a ← W^{-1} f⃗_b
    for i ← 1,3 do
        T_i ← sqrt((f_a^i)^2 + (f_a^{i+3})^2)     ▷ Computing the thrust
        if T_i > T_max then                       ▷ Saturating the thrusts
            T_i ← T_max
        if T_i < T_min then
            T_i ← T_min
        α_i ← atan2(f_a^{i+3}, f_a^i)             ▷ Computing the angle
        if α_i > α_max then                       ▷ Saturating the angles
            α_i ← α_max
        if α_i < −α_max then
            α_i ← −α_max
    return f⃗_a
```

Algorithm 7.2 – Saturation of the actuators.

$$5.3 \times 10^{-1}\,\mathrm{N} < T_z < 1.4 \times 10^{1}\,\mathrm{N}, \tag{7.32}$$

7.2.3.2 Point stabilization

The first condition for the selected value of the vertical thrust is to allow for point stabilization in the case of rotating propellers. This corresponds obviously, following equation (7.29), to the generalized body-forces vector:

$$\vec{f}_b = \begin{pmatrix} 0 \\ 0 \\ -T_z \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{7.33}$$

This vector is independent of the sense of rotation of the propellers. On the contrary, the control vector $\vec{f}_a$ depends on this choice because of the control matrix $\mathbf{W}$. In the chosen layout, using the matrix given in equation (6.40) on page 130, we find the control vector:

$$\vec{f}_a = \frac{T_z}{(9d^2 + 8\sigma^2)} \begin{pmatrix} 3d^2 + 4\sigma^2 \\ 3d^2 + 2\sigma^2 \\ 3d^2 + 2\sigma^2 \\ d\sigma \\ d\sigma \\ d\sigma \end{pmatrix} \tag{7.34}$$

Using equation (6.22), we can compute the formal values of the angles and thrusts:

$$\begin{cases} \alpha_1 &= \operatorname{atan2}(d\sigma, 3d^2 + 4\sigma^2) \\ \alpha_2 &= \operatorname{atan2}(d\sigma, 3d^2 + 2\sigma^2) \\ \alpha_3 &= \operatorname{atan2}(d\sigma, 3d^2 + 2\sigma^2) \\ T_1 &= \frac{\sqrt{d^2\sigma^2 + (3d^2+4\sigma^2)^2}}{9d^2+8\sigma^2} T_z \\ T_2 &= \frac{\sqrt{d^2\sigma^2 + (3d^2+2\sigma^2)^2}}{9d^2+8\sigma^2} T_z \\ T_3 &= \frac{\sqrt{d^2\sigma^2 + (3d^2+2\sigma^2)^2}}{9d^2+8\sigma^2} T_z \end{cases} \tag{7.35}$$

It should be noticed that the rest angles do not depend on the chosen vertical thrust but only on the length of the arm and on the ratio between thrust and moment generated by a propeller. The rest angles and thrusts can be numerically evaluated to be:

$$\begin{cases} \alpha_1 &= 4.00 \times 10^{-2}\,\mathrm{rad} \\ \alpha_2 &= 4.04 \times 10^{-2}\,\mathrm{rad} \\ \alpha_3 &= 4.04 \times 10^{-2}\,\mathrm{rad} \\ T_1 &= 0.336 \times T_z \\ T_2 &= 0.333 \times T_z \\ T_3 &= 0.333 \times T_z \end{cases} \tag{7.36}$$

To respect the minimal thrust condition given in table 6.1 on page 120, we should choose T_z so that:

$$T_z > \frac{9d^2 + 8\sigma^2}{\sqrt{d^2\sigma^2 + (3d^2 + 2\sigma^2)^2}} T_{min} = 6.0 \times 10^{-1}\,\mathrm{N} \tag{7.37}$$

which appears to be a greater lower bound than the one found in equation (7.32).

7.2.3.3 Translations

We mentioned previously that the aggressiveness of the chosen trajectory plays a role in the saturation of the actuators. We suggest to study simple translations, *i.e.* with constant yaw. Without loss of generality, we suppose that the yaw is null. The generalized-body-forces vector reads:

$$\vec{f}_b = \begin{pmatrix} m'\ddot{x} \\ m'\ddot{y} \\ -T_z \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{7.38}$$

Therefore, the control vector formally reads:

$$\vec{f}_a = \frac{T_z}{(9d^2+8\sigma^2)}\begin{pmatrix}3d^2+4\sigma^2\\3d^2+2\sigma^2\\3d^2+2\sigma^2\\d\sigma\\d\sigma\\d\sigma\end{pmatrix} - \frac{\sqrt{3}}{3}m'\ddot{x}\begin{pmatrix}0\\\frac{\sigma}{d}\\-\frac{\sigma}{d}\\0\\-1\\1\end{pmatrix} - \frac{m'\ddot{y}}{(9d^2+8\sigma^2)}\begin{pmatrix}2d\sigma\\-d\sigma\\-d\sigma\\2\left(3d^2+2\sigma^2\right)\\-\left(3d^2+4\sigma^2\right)\\-\left(3d^2+4\sigma^2\right)\end{pmatrix} \tag{7.39}$$

It appears there clearly that the first rotor does not bring any contribution to forward translations. The translation contributes (*i.e.* in the absence of the T_z and $\ddot{y}$ terms) to the two other rotors same thrusts and opposite angles. To simplify the analysis, we perform numerical evaluations of the different terms based on the values given in table 6.1 on page 120 and table 7.2 on page 140. These are:

$$\vec{f}_a = \ddot{x}\begin{pmatrix}0\\-0.0707\\0.0707\\0\\0.577\\-0.577\end{pmatrix} + \ddot{y}\begin{pmatrix}-0.0269\\0.0134\\0.0134\\-0.664\\0.336\\0.336\end{pmatrix} + T_z\begin{pmatrix}0.336\\0.332\\0.332\\0.0134\\0.0134\\0.0134\end{pmatrix} \tag{7.40}$$

Numerical evaluations, as given for the second propeller during a forward acceleration in figure 7.5, show that a low value of T_z leads to saturations of the tilt-angle even for low accelerations. This calls for a rather high value of T_z in contradiction with the objective of energy saving. The choice was then made to set $T_z = 4.3\,\mathrm{N}$. This allows to perform accelerations as high as $1\,\mathrm{m\,s^{-2}}$ with the third of the energy consumption of the flying tricopter.

7.2.4 Experiments

7.2.4.1 Differentiation of noisy data

The data gathered during the experiments come from different sources with a non constant time step which is a multiple of 0.05 s due to packet losses and come with noise. Position data, for example, are particularly noisy, and it appears to be difficult to evaluate the real speed and acceleration of the tricopter by simple computational means. Figure 7.6 shows in blue in the left-hand column the values obtained through finite differences.

It is common knowledge that this process does not apply well to noisy data. Indeed, while giving appropriate results for the speed, with the presence of various peaks, the peaks due to the noise make the acceleration computation useless.

The controller suggested in equation (7.18) on page 138 makes use only of the velocity data. On-line, the tricopter uses a simple finite differences scheme to access to an evaluation of its velocity.

To obtain, after the experiments, a more accurate evaluation of the velocity, the scheme suggested by [Chartrand, 2011], was applied. It relies on a total-variation regularization which uses gradient descent to minimize a functional equation penalizing irregularities in the output

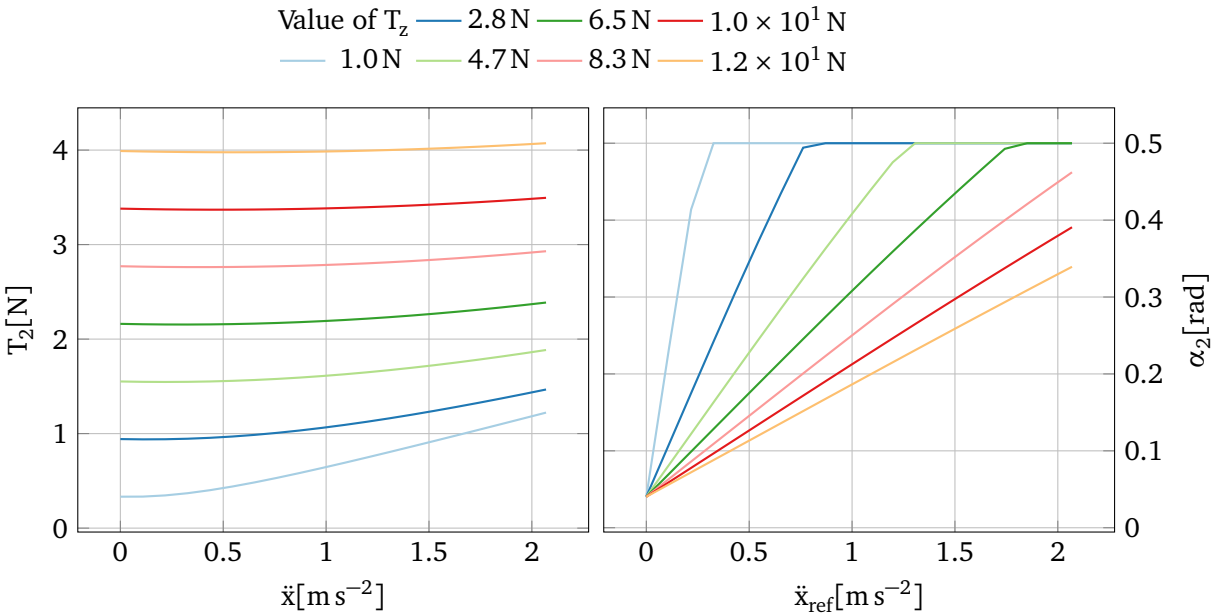


Figure 7.5 – Influence of the choice of T_z on saturations. Open-loop controls are represented for the second nacelle for given longitudinal accelerations in the case of null lateral and yaw dynamics for various choices of T_z.

function and discrepancy between the data and the computed solution. That is, the derivative u of the function f is computed as the minimum of the functional equation:

$$F(u) = \alpha \int_0^L |u'| + \frac{1}{2} \int_0^L |Au - f|^2 \tag{7.41}$$

where α balances the importance of the two terms and $Au(x) = \int_0^x u$ is the operator of integration. This process needs the whole data to be available and is computationally intense and may thus be applied only offline. The methods assumes constant steps in the data. Therefore, missing data points are recreated using linear interpolation. An example of applying this method is given in figure 7.6, along with the derivatives obtained through finite differences. The derivative obtained by this process is plotted in green in the right-hand column.

Chartrand's derivation presents none of the peaks appearing with finite differences and matches elsewhere the finite differences derivation. As was explained in [Chartrand, 2011], this process might induce a discrepancy of the derivative. This contrast loss appears in our application at narrow peaks. We may observe this effect using the angular data given by the IMU. Indeed, the IMU outputs both the angular velocity of the tricopter and its attitude. We compare in figure 7.7 the angular velocity obtained by the IMU (in blue) and the angular velocity obtained by differentiating the angular data output of the IMU using Chartrand's derivation (in green). There is no discrepancy on the "broad" extremum of the lower-right figure but there is some occurring at the narrow extrema of the lower-left figure.

In the remaining of this chapter, every time noisy data has to be differentiated, the differentiation is performed using Chartrand's regularization scheme and is plotted in green: ——. In this

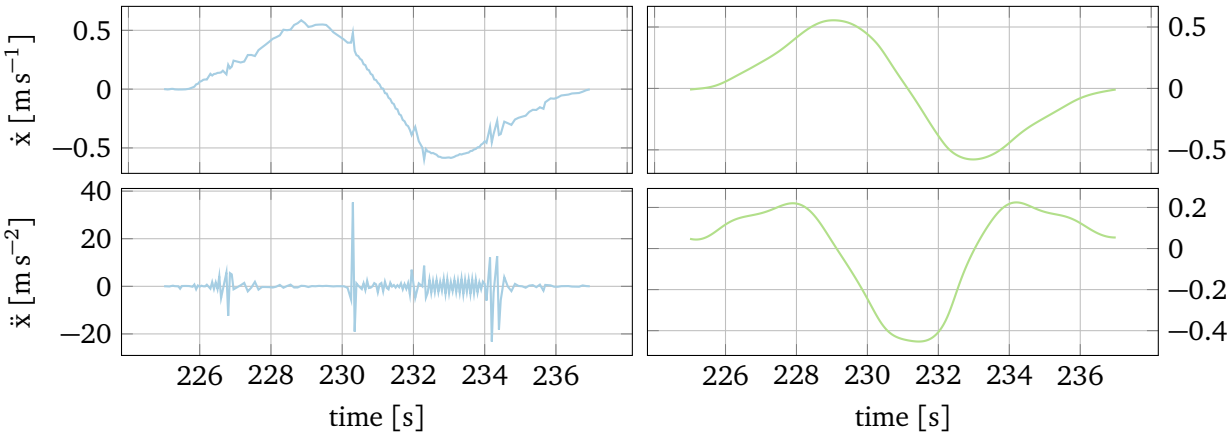


Figure 7.6 – Differentiation of noisy data, using finite differences in the left-hand column, and Chartrand's derivation [Chartrand, 2011] in the right-hand column.

different cases, we should keep in mind the aforementioned discrepancy on narrow maxima and be careful when analyzing tracking of sharp changes in velocities or accelerations. Data obtained directly from the various data acquisition tools are plotted in blue: ——. Reference data are plotted in red: ——.

7.2.4.2 Experimental results

During the experiments, we made the tricopter follow repeatedly the trajectory presented in section 7.1.2. The travel time of $T = 3\,s$ was chosen. This leads the tricopter to complete the trajectory in 12 s and reaching in the curve a speed of $5.9 \times 10^{-1}\,m\,s^{-1}$ (about $2\,km\,s^{-1}$). It is the maximum possible speed for the rolling tricopter. It appeared that for greater speeds the tricopter leans and nearly takes-off. This unexpected behavior can be at least partially explained by the ground effect. Indeed, the thrust of a propeller increases at constant power with ground proximity. A simple model in hover is [Johnson, 1994, 3.6 p. 124]:

$$\frac{T}{T_\infty} = \frac{1}{1-(R/4z)^2} \tag{7.42}$$

Where T_∞ is the thrust of the propeller for the given power in an infinite volume,R is the radius of the propeller and z the altitude of the rotor. In our case the ratio between propeller radius and altitude is around 1 and the ground proximity is supposed to increase the efficiency of each propeller by 7 %. [Powers et al., 2012] showed that this model is relevant for for quadrotors flying near the ground and can be used, with an accurate measurement of the altitude, to map the ground. In our case, a greater efficiency of one or more of the rotors is a pertinent explanation of the inclination of the tricopter. Anyway, as will be shown in the following, the quality of our experimental results show that the suggested model for the rolling tricopter is accurate. Ground effect can thus be neglected at the chosen power level.

The x and y trajectories followed by the tricopter are presented in figure 7.8 and are compared with their respective reference trajectories. The errors, plotted in the lower row, are mostly due to the delay. Static error is low and the tracking error is anyway lower than 1×10^{-1} m. The resulting path is depicted in figure 7.9. It can be seen on this figure that the path traveled by

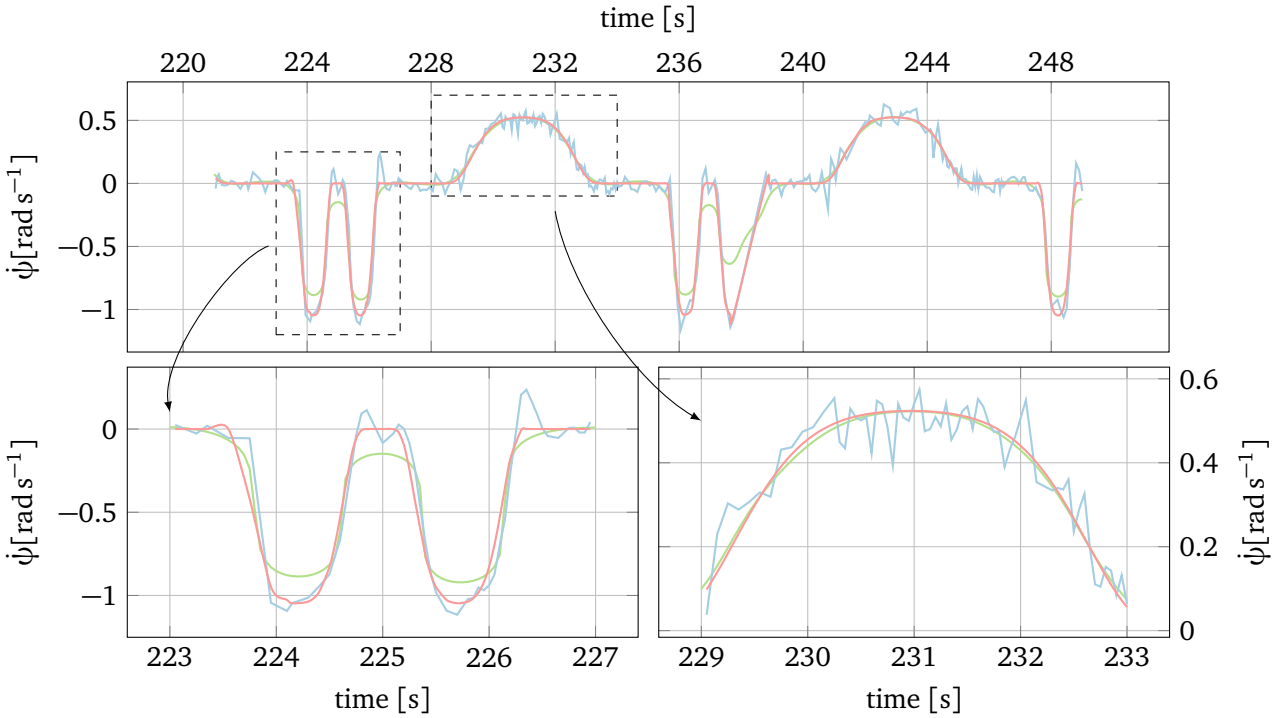


Figure 7.7 – Discrepancy of the derivation induced by Chartrand's regularization. The reference trajectory is red, the output of the IMU is blue and the velocity obtained by applying Chartrand's derivation to the yaw angle given by the IMU is green. The discrepancy induced by Chartrand's derivation can be seen at the various extrema of the reference on the lower -left plot.

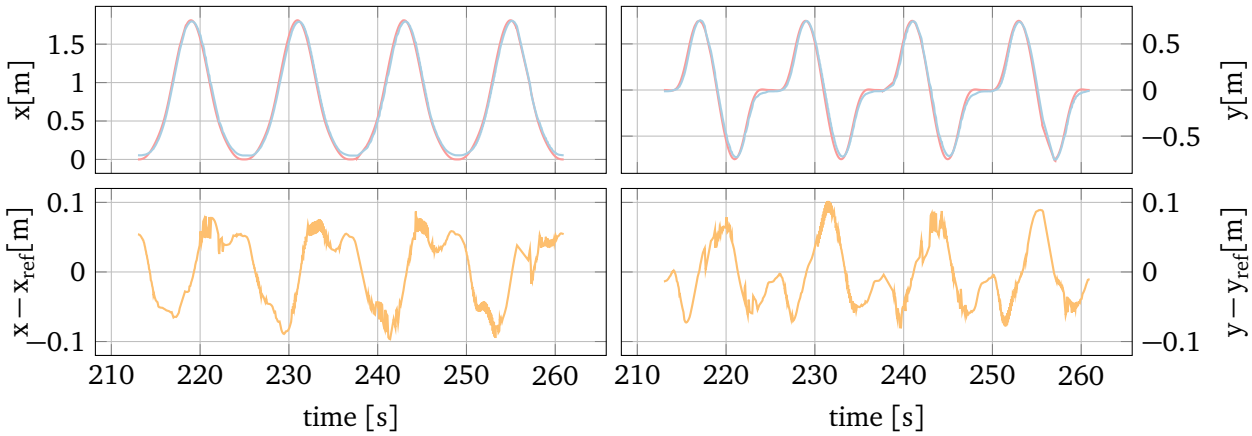


Figure 7.8 – Evolution of the reference trajectories (red), their real values (blue) and the error (orange) with x in the left-hand column and y in the right-hand column.

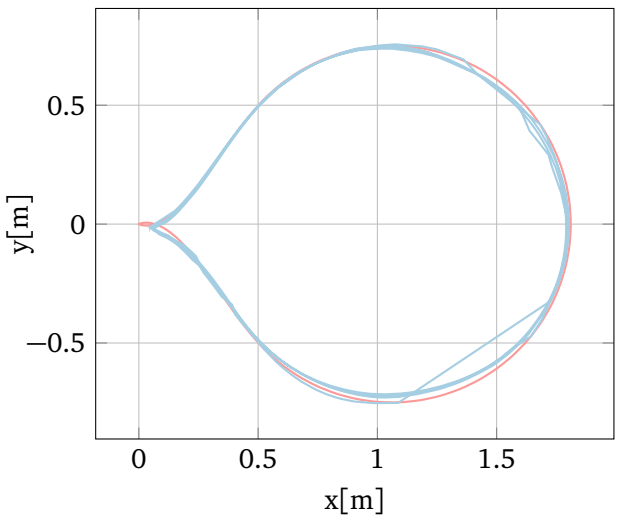


Figure 7.9 – Reference (red) and real (blue) paths. The straight line in the lower-right part of the path is due to a data loss and was not really traveled.

the rolling tricopter is really close to the wanted path. The performance of the controller on the translations can thus be considered as satisfying.

The velocity of the tricopter is represented in the left-hand column of figure 7.10. The controller tracks well the reference velocity on the x axis and y axes. However, it does not reach the peak velocities and the little "bumps" seen on the reference y velocity are damped out. The computation of the norm of the velocity, that is depicted in the lower left graph, makes an oscillation appear that is not obvious on the separate axis-velocities. This oscillation starts after overshooting the reference speed on the circular section of the curve where the norm of the velocity should be constant.

The acceleration of the tricopter represented in the right-hand column of figure 7.10 gives an explanation of this undesired behavior. The peaks and "bumps" of the acceleration on the x and y axes are damped out. While this damping could be partially explained by the discrepancy

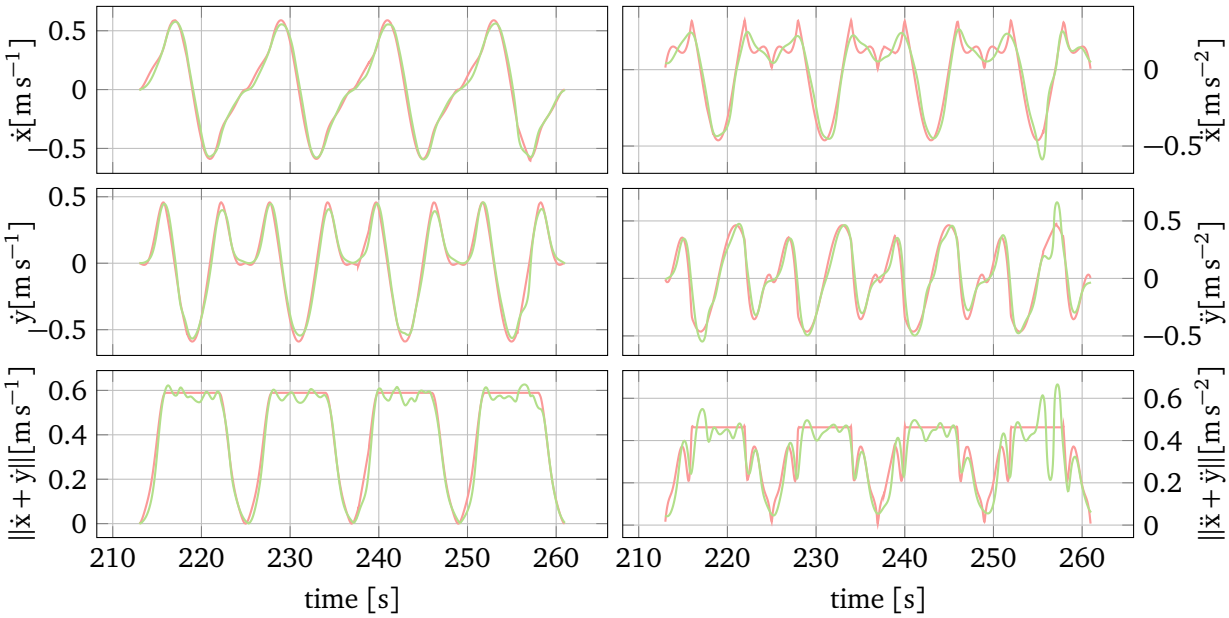


Figure 7.10 – Evolution of the velocity (left-hand column) and acceleration (right-hand column). References are plotted in red, values obtained by Chartrand's derivation of the Motion tracking data are plotted in green. The first row represents the values on the x axis, the second row represents the values on the y axis, the last row represents the evolution of the norm.

induced by Chartrand's derivation, oscillations are clearly to be seen on the constant sections of the reference total acceleration. This unwanted behavior can be explained by the choice of only twice differentiable reference trajectories. Indeed, the dynamics of the actuators (servomotors and BLDC motors) were not considered while establishing the model of the tricopter. Taking these dynamics into account would lead to a higher order model. Smoother reference trajectories could be thus enhance the quality of the trajectory tracking.

During the presented set of experiments, we choose to set the coefficient ν to 0.5 in equation (7.3). The tricopter turns then twice as fast and in the first (resp. last) half of the allowed time from (resp. to) zero heading at the start (resp. end) of the trajectory. It has thus to remain longer with constant attitude $-\frac{\pi}{4}$ (resp. $\frac{\pi}{4}$) than in the default reference yaw trajectory as plotted figure 7.2. As can be seen from in figure 7.11, the angular velocity obtained by Chartrand's derivation of the IMU angular data (in green) almost perfectly matches the reference angular velocity in the "sombrero"-like section of the angular velocity reference. In the narrow peaks, Chartrand's derivative of the angle, experiences the aforementioned discrepancy in the narrow extrema in comparison with the angular velocity measured by the IMU. Simular observations might be done for the angular acceleration. The controller appears thus to be particularly efficient on the angular dynamics as well.

The thrust and tilt angles controls during the experiment are plotted in figure 7.12. The controls based on the sole feedforward term are plotted in black above the actual controls. We may first notice that the feedforward terms and the actual controls have similar shapes. However, the thrusts and the tilt angles present several not predicted peaks. These are mostly located during the circular section with constant acceleration of the trajectory as is outlined by the vertical dashed lines. These peaks seem to correspond to the oscillations found in the norm

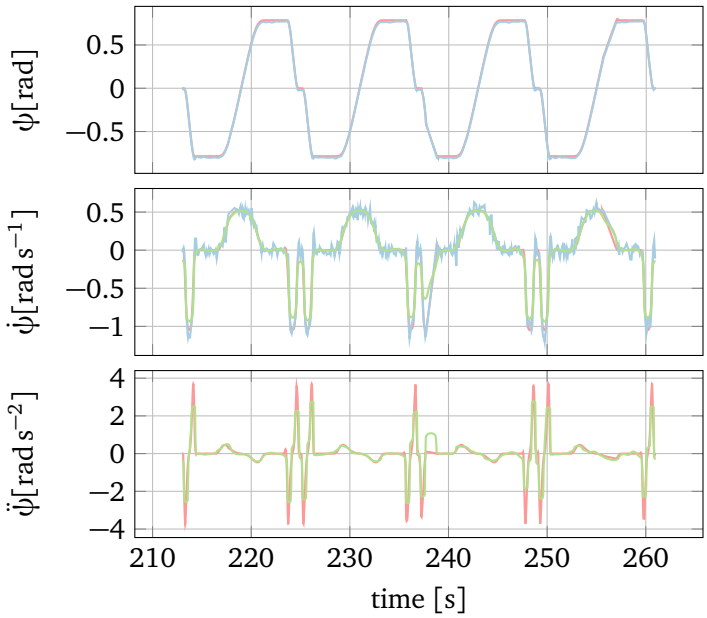


Figure 7.11 – Reference and experimental value of the yaw (upper graph) with reference (red) and real value (blue), velocity (middle graph) with reference (red) and real value as given by the IMU (blue) and acceleration (lower graph) with reference (red) and real value (green) obtained by differentiation of the angular velocity. The "bump" in angular acceleration at 239 s is a computational artifact due to data loss.

of the acceleration seen in figure 7.10. They are systematic and are thus the proof of unmodeled effects. These unmodeled effects might be explained first by ground effect or static and dynamic friction of the ball casters. Static friction, for example, is clearly impacting the tricopter. With the sole feedforward term, the applied thrust is not sufficient to over come the static friction of the ball casters and the tricopter is not able to start.

As a conclusion, it appears that using the tricopter as a ground robot allows the platform to be reliably used as an indoor exploration platform. A first flatness-based controller was implemented and tested. The results of this trajectory tracking experiment are particularly conclusive and lead us to the second milestone of our applications: trajectory tracking for the flying tricopter.

7.3 The flying tricopter

In the previous section, we tested the tricopter with a first application. The tricopter proved itself to be reliable enough to track autonomously a trajectory when rolling on the ground. In this section, we apply the controller suggested in equation (7.18) to flights. We tested various aspects of the controller: Position and attitude stabilization, altitude control and the tracking of the trajectory presented in the previous sections.

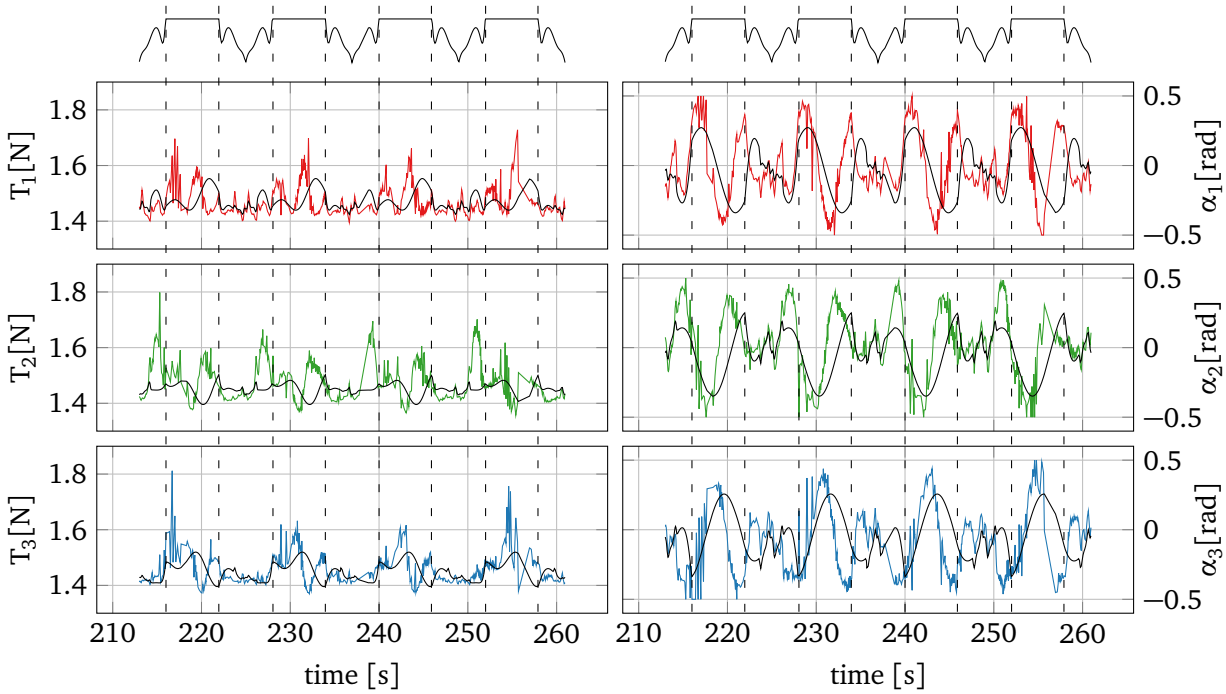


Figure 7.12 – Evolution of the controls along the trajectory. The thrusts are represented in the left-hand column while the tilt angles are represented in the right hand side. The first row (in red) corresponds to the controls of the first arm, the second row (in green) to the second arm and the third row (in blue) to the third arm. The nominal value obtained by the sole feedforward term is represented in black. The graph sketched atop the whole figure is the norm of the reference acceleration. Dashed vertical lines outline the limits of the circular section of the trajectory.

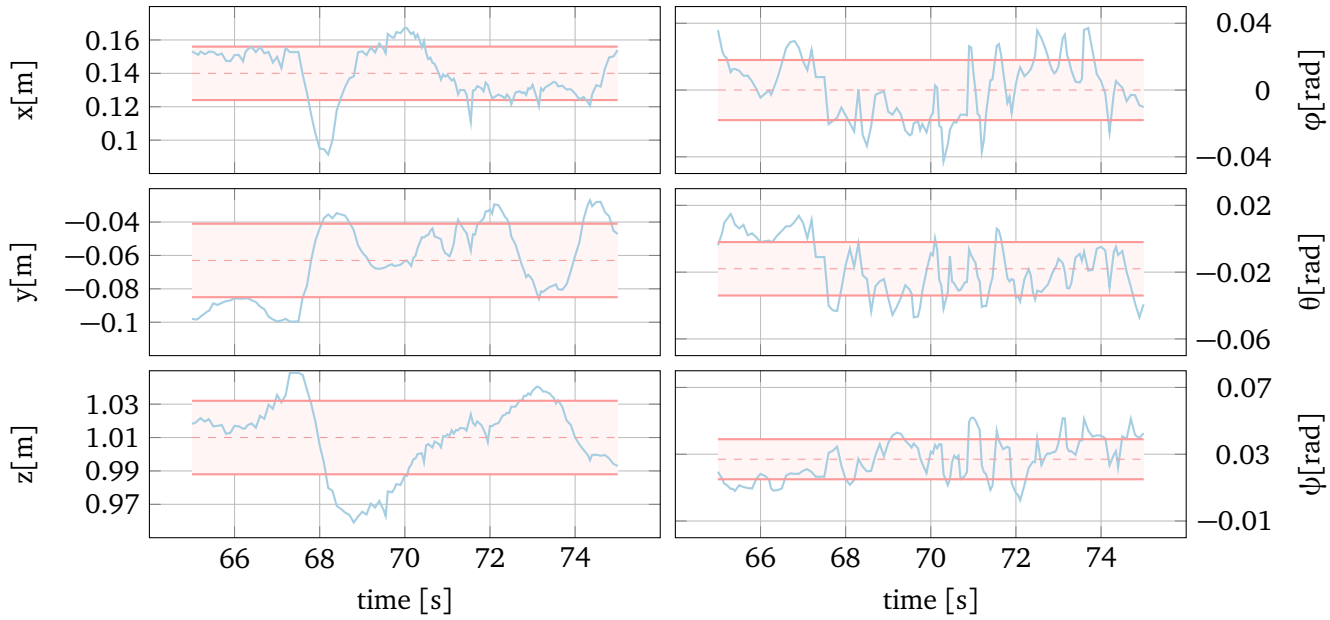


Figure 7.13 – Position and attitude stabilization. The dashed red line represents the mean of the value during the time window, the area filled in red represents the standard deviation. The reference position is (0 m, 0 m, 0.9 m, 0 rad, 0 rad, 0 rad,).

7.3.1 Position and attitude stabilization

Taking-off autonomously is a hazardous task for a helicopter. Indeed, the model adopted for the thrust in equation (6.2) does not take into account the proximity of the ground which induces the so-called "ground effect". As a consequence, the first experiments were conducted after manual take-off, *i.e.* using the remote control. The first experiment we perform, after a manual take-off, is to maintain the position and attitude of the tricopter. In the following we describe the attitude of the tricopter by the roll-pitch-yaw angles, respectively represented by the variables φ, θ and ψ. We adopt the $z-y-x$ convention (also called Cardan or nautical angles). The angles are then defined such that the following equality holds:

$$\mathbf{R}_{\mathcal{I}}^{\mathcal{B}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{7.43}$$

The position and attitude of the tricopter are plotted in figure 7.13. On the one side, the static error in position is of the order of 1×10^{-1} m and the standard deviation around this mean value is of the order of 2×10^{-2} m. On the other side, the static error in attitude is of the order of 2×10^{-2} rad (about 1°) and the standard deviation around this mean value is of the same order. We can thus assert that the controller is reliable to maintain attitude and position. The static error could however been corrected by the addition of an integral term. This could be a future axis of research.

The analysis of the controls show an unexpected behavior. The thrust controls for the second and third rotors, *i.e.* the clockwise rotating propellers, are slightly below the predicted value. This behavior has been observed by the team at CSTCE during parameter identification but this

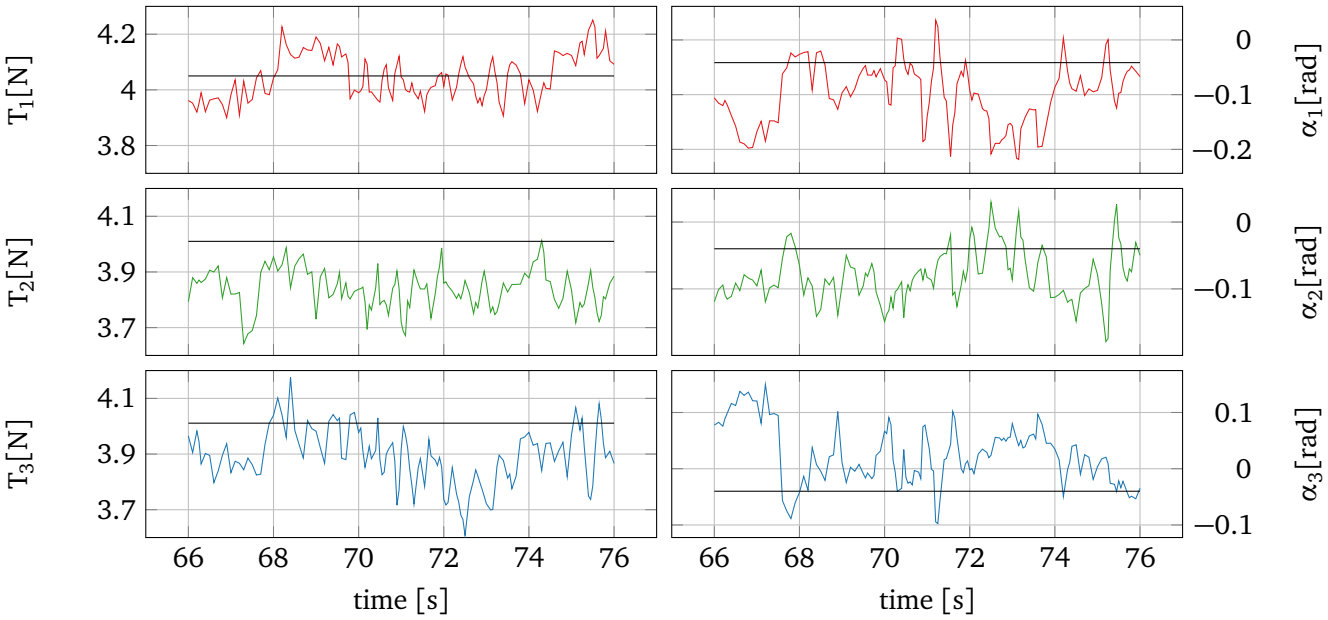


Figure 7.14 – Evolution of the controls during position and attitude flight stabilization. The thrusts are represented in the left-hand column while the tilt angles are represented in the right hand side. The first row (in red) corresponds to the controls of the first arm, the second row (in green) to the second arm and the third row (in blue) to the third arm. The nominal value obtained by the only feedforward term is represented in black.

effect was not taken into account in our model and the same thrust coefficient k_T has been used for both sort of propellers.

7.3.2 Altitude tracking

We describe by altitude tracking the tracking of a vertical trajectory. This task is conceptually similar to horizontal trajectory tracking but differs in an important aspect: the ground effect introduced in the previous section. Assuming the horizontal dimensions of the flight area are much greater than the vertical dimension, ground effect plays a more significant role in tracking vertical trajectories than in tracking a trajectory parallel to the ground.

We present in figure 7.15 two sets of the position, speed and acceleration of the tricopter on an altitude increase step of 0.75 m. In both cases, the altitude presents a static error in the beginning. The shape of the reference is accurately followed by the tricopter. However, in both cases, the real increase in altitude is only about 0.5 m. It would be necessary to confirm this experience in a greater flight area with a higher ceiling. This would allow to perform the transition far from the ground and from the ceiling. Indeed, the latter also has an influence on the performance of the propellers, though lower than the ground.

Anyway, the controller proved itself to be efficient enough to maintain altitude. Therefore, we perform in the next section trajectory tracking in level flight.

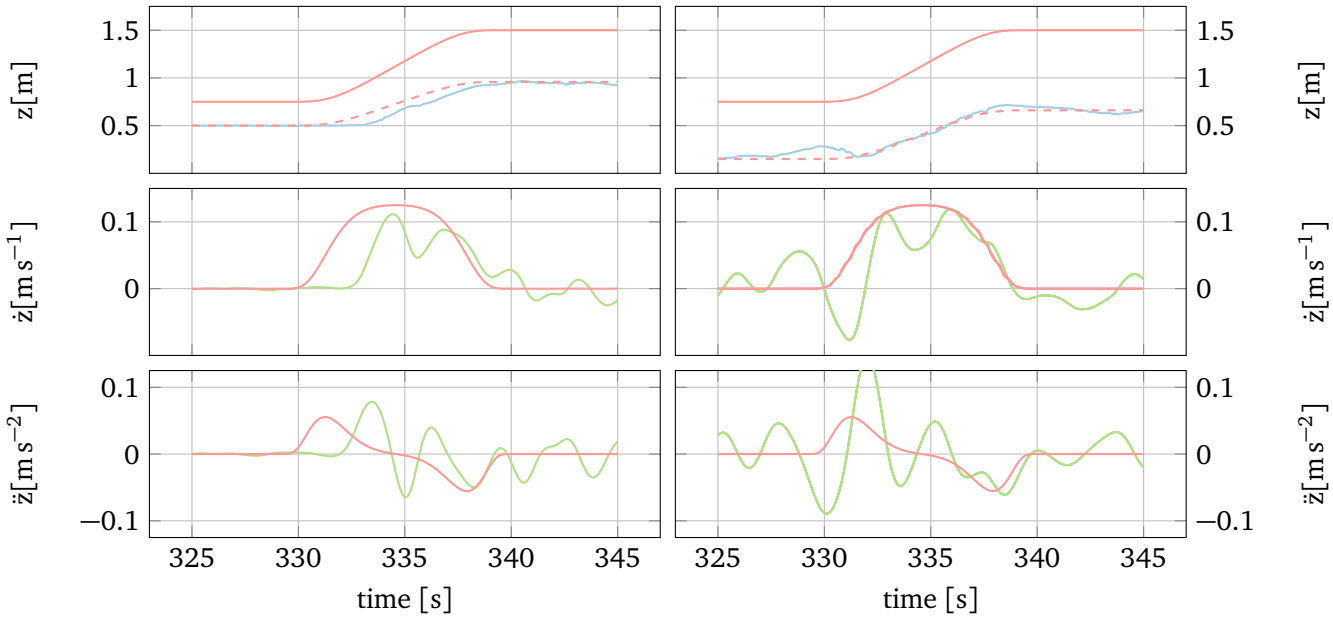


Figure 7.15 – Tracking an increase in altitude reference. The position red dashed line is scaled and translated from the reference trajectory. Each column represents a different experiment.

7.3.3 Trajectory tracking

After successively showing the ability of the tricopter to stabilize in flight and to follow a trajectory on the ground, we perform the last experiment of this section: trajectory tracking in level flight. The reference trajectory used in the experiment is the one that has been tested on the ground.

As shown by figure 7.16, the tricopter follows accurately the reference velocities and accelerations. The shape of the trajectory is appropriately followed. Again, the static error in position is in the order of 10^{-1} m. The path traveled by the tricopter is illustrated figure 7.17. After correcting the important static offset in x, the path (in dashed blue line) follows properly the reference path. It has to be noticed that the flown circle is smaller than the reference circle. Indeed, the real trajectory of the tricopter in y does not reach the two positive "peaks" of the reference trajectory in figure 7.16 top right. As no other precise data on this discrepancy are at hand, this could be, for example, explained by an asymmetry of the test room resulting in a higher repulsive thrust in positive y than in negative y due to different proximities of the walls.

The tracking of the attitude is illustrated for the yaw in figure 7.18 while the stabilization of the roll and pitch angles is illustrated in figure 7.19. The performance of the controller is satisfying in both cases. The tricopter accurately follows the reference yaw angle while decently stabilizing the roll and pitch angles.

The controls along the trajectory are illustrated in figure 7.20. The feedforward terms for the angles are adequately followed by the controller but, again, the feedforward overestimate the needed thrust. The thrusts follow the feedforward term but are lower by an average of 0.1 N.

This last experiment finishes to prove the performance of the proposed controller for trajectory tracking purposes in flight condition. The static error is higher in such conditions than when rolling on the ground but the tricopter tracks accurately the assigned trajectories. It is to

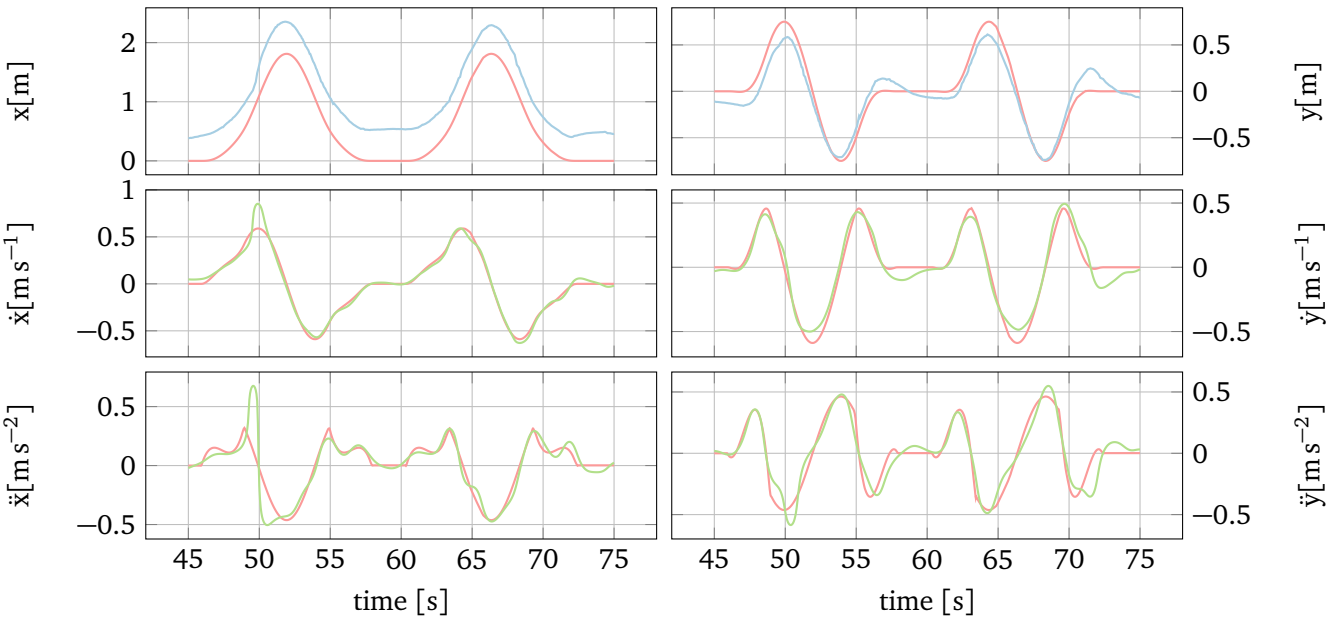


Figure 7.16 – Tracking a trajectory in level flight with position (upper graph) with reference (red) and real value (blue), velocity (middle graph) with reference (red) and real value as given by the IMU (blue) and by differentiation (green) and acceleration (lower graph) with reference (red) and real value (green) obtained by differentiation of the angular velocity.

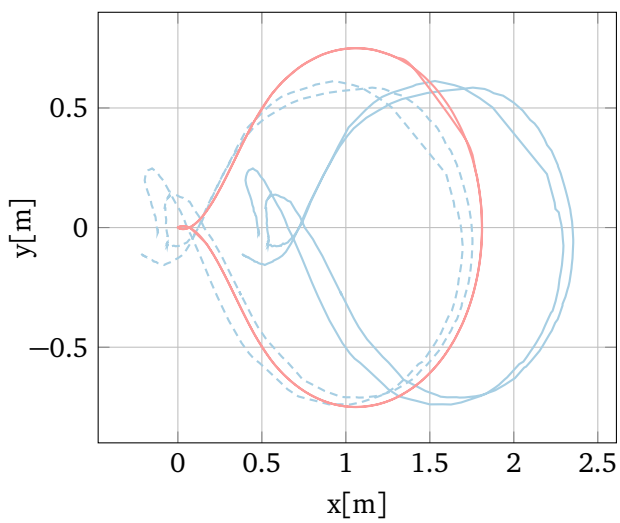


Figure 7.17 – Trajectory tracking in level flight: xy path. Reference is red while the real path obtained while flying is blue. The dotted blue path is obtained by subtracting the x offset as a comparison.

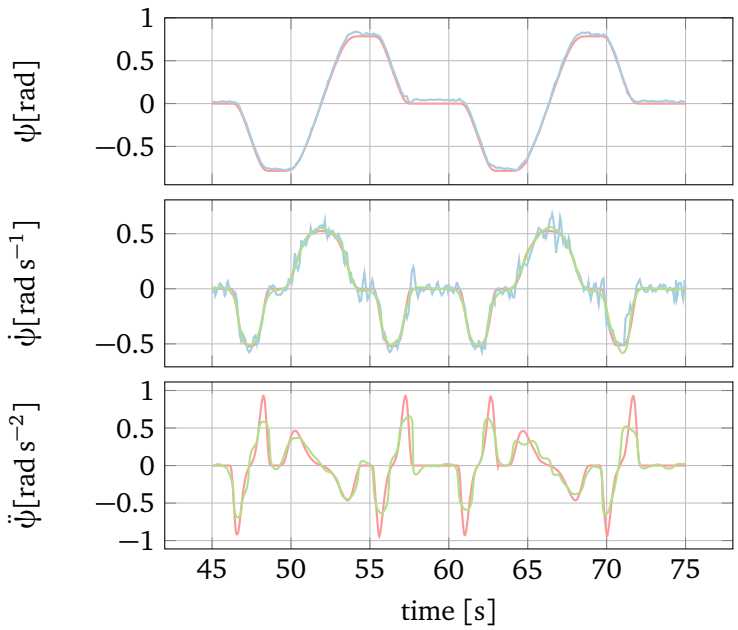


Figure 7.18 – Reference and experimental value of the yaw during level flight (upper graph) with reference (red) and real value (blue), velocity (middle graph) with reference (red) and real value as given by the IMU (blue) and acceleration (lower graph) with reference (red) and real value (green) obtained by differentiation of the angular velocity.

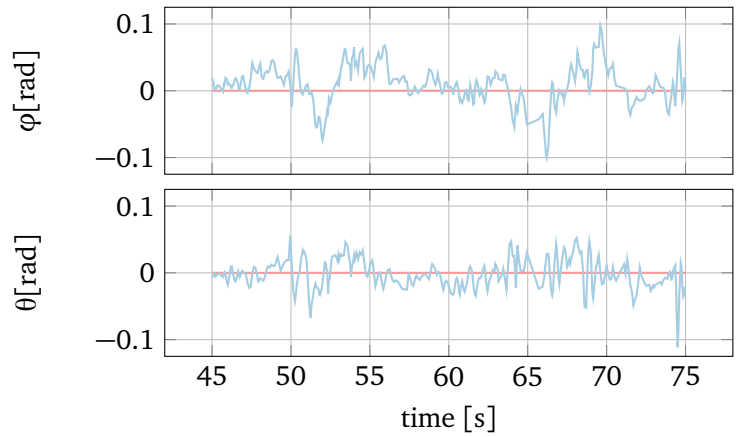


Figure 7.19 – Reference and experimental value of the roll (upper graph) and pitch (lower graph) angles with reference (red) and real value (blue).

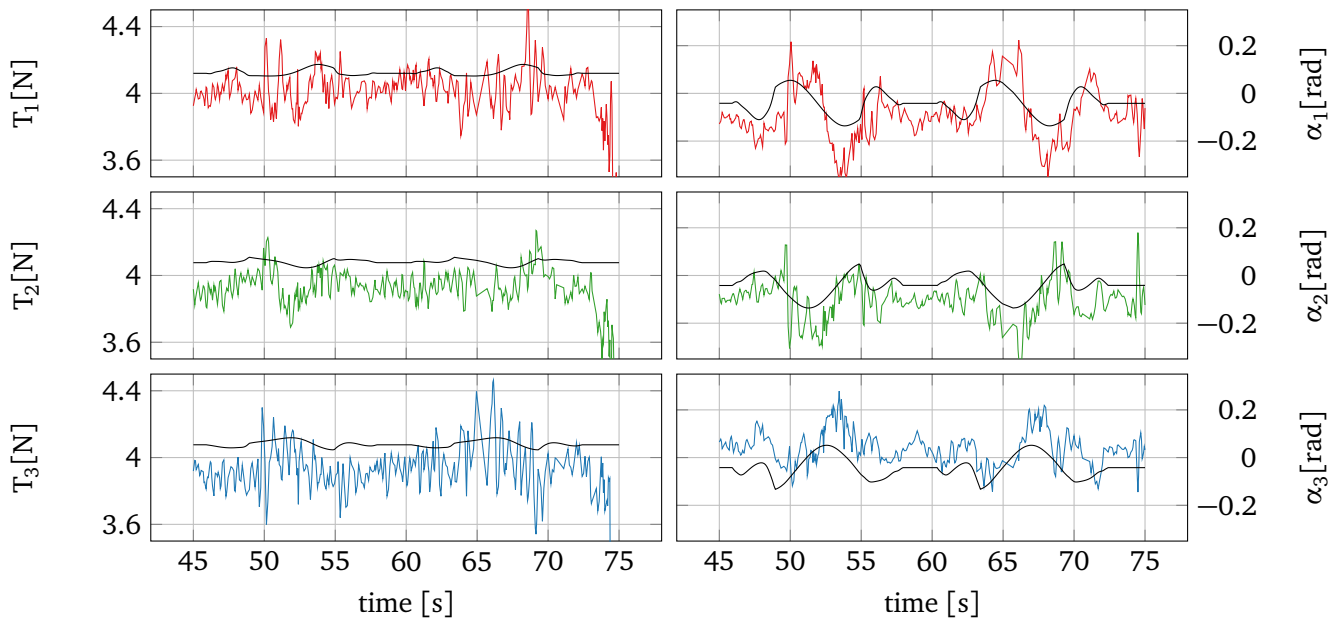


Figure 7.20 – Evolution of the controls during trajectory tracking in level flight. The thrusts are represented in the left-hand column while the tilt angles are represented in the right hand side. The first row (in red) corresponds to the controls of the first arm, the second row (in green) to the second arm and the third row (in blue) to the third arm. The nominal value obtained by the only feedforward term is represented in black. The drop in thrust at the end of the trajectory is due to a landing maneuver.

Figure 7.21 – The pendulum. The load orientation PCB is on the left side of the basis. One of the potentiometers is on the right side of the basis. The other is hidden, on the side opposite to the PCB.

noticed that the controller was tested in particularly harsh conditions. Indeed, the flight area is particularly small. Therefore, the tricopter was flying in presence of constant turbulences. Next, we present the premises of a future application: pendulum load transportation.

7.4 Carrying a load

In comparison with usual quadrotor UAVs, the tricopter has six independent controls. As was presented in the review, various works are conducted to perform pendulum load transportation with such quadrotor UAVs. In the following, we show how using the tricopter allows for more control on this class of underactuated compound systems. We present first the orientation measurement unit that we designed to achieve this task. We present then a flatness-based control approach achieving pendulum load transportation.

7.4.1 The pendulum load

7.4.1.1 The load orientation measurement unit

The goal of this chapter is to use the tricopter to carry a pendulum load. The load we want to carry is a steel cylinder weighting $m_L = 100\,g$. We want to link the load to the tricopter using a carbon rod of length $l = 20\,cm$ weighting $2\,g$. To attach it to the tricopter, we made the choice of using an industrial joystick, namely the M31L-0-M1P by CH-Products. The whole pendulum load set is to see in figure 7.21. This choice relies on various aspects of this joystick:

- It is ready to use and affordable ;
- It is easily integrated to either the top or bottom plate of the tricopter by its fastening screws ;
- Its "core" weights only 55 g and is thus easily integrable to the tricopter with a total additional weight to the tricopter of 157 g ;
- It has two rotation axes that can be set parallel to $\vec{x}_{\mathcal{B}}$ and $\vec{y}_{\mathcal{B}}$. Thus the pendulum can move in a cone of 60° opening ;
- It works on the same current levels as the tricopter ;
- It provides a direct measure of the angles by two potentiomenters with reasonable noise level and linearity.

The last point is the most important. Indeed, to the best of our knowledge, most previous works (with the exception of [Maza et al., 2010]) used motion tracking systems to measure the position of the load. As was seen in the previous section, such systems are pertinent for position control. For its orientation, the tricopter needs accurate data at a higher rate. This is the role of the IMU. The potentiometers of the joystick play this role and we can get a precise measure of the position of the load with respect to the tricopter.

To perform this measure, we developed a dedicated PCB. This PCB, depicted in figure 7.22 is built around the same microcontroller used on the main tricopter board. Therefore, our custom designed board works on the same power level as the main board and can communicate over its various communication buses. As illustrated on figure 7.22, we made the choice to communicate with the main board over the available I²C bus. This bus is fast enough for the few data we need to send and needs only two wires. The maximal volume of data can be evaluated at 32 bit – eight floating point numbers – every 5 ms – a tick on the main board. This represents a maximal data throughput of 6.25 Kibit s^{-1} easily accommodated by I²C. In our implementation, only 11 bit are sent. They correspond to the two angles, the number of samples and a parity bit.

The load PCB uses two Analog-to-Digital-Converters (short **ADC**) to convert the power levels of the potentiometers –,the measures of the angular positions – to digital values. The analog values are first passively filtered by hardware low-pass filters and then digitally averaged by the microprocessor. In the 5 ms between two transmissions, 17 samples are taken. This step could be improved using another filtering scheme to get a more precise evaluation of the angles at the exact time of the transmission.

7.4.1.2 Parameter identification

The power output of the potentiometers is proportional both to the power input and to the angular rotation of the potentiometer. This linearity is said, quoting the information of the manufacturer, to be in the limits of ±1 %. To test this assertion, we compare the outputs of the potentiometers to absolute orientation data evaluated by our motion tracking system.

On the one hand, the motion tracking system outputs the orientation as unit quaternions. On the other hand, the potentiometers measure two voltages. We convert them to angle assuming a

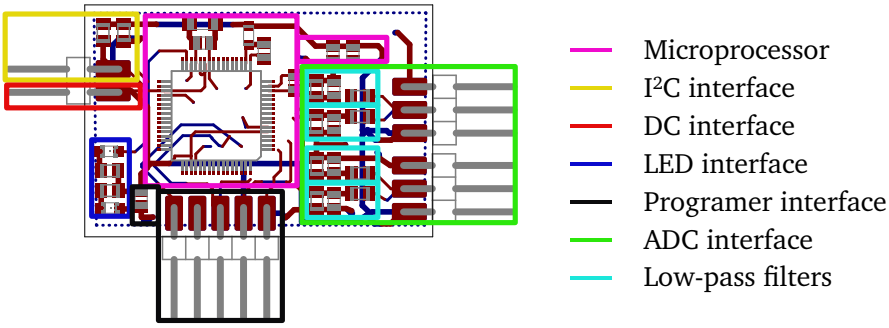


Figure 7.22 – The load orientation measurement board. The board is 1” × 1.5” (about 25.4 mm × 31.8 mm).

linear dependency. These angles are given with respect to fixed axes as depicted in figure 7.23. To investigate the validity of the linear hypotheses, we converted the quaternions from Vicon to these two absolute angles. The result of this comparison is shown in figure 7.24 and validate the linearity of the potentiometers. The apparent errors may have as origin both an inaccurate choice of the parameters for the potentiometers or an imprecise evaluation of the offset e used by the motion tracking system. This imprecisions could be solved in the future by a more careful study of the system.

7.4.2 Dynamics of the system

7.4.2.1 Modified dynamics of the tricopter

The addition of the load modifies the dynamics of the tricopter presented in 6.42. Indeed the load adds both a force term and a torque term that we will not neglect. This departs from other works (see *e.g.* [Sreenath et al., 2013]), where this influence was neglected in order to simplify the model. This unusual assumption is justified by the total actuation of our tricopter: the orientation and the position of the load are decoupled. Hence it seems possible to stabilize the pendulum load acting on the attitude of the tricopter while keeping a constant position.

The additional load is described by a point mass load of mass m_L, at position $\vec{p}$, linked to the tricopter by a massless rigid rod. This rod is linked to the tricopter by a frictionless joint. This joint has only two degrees of freedom, rotations around $\vec{x}_{\mathcal{B}}$ and $\vec{y}_{\mathcal{B}}$ axes. The orientation of the load with respect to the tricopter body $\mathbf{R}_{\mathcal{B}}^{\mathcal{P}}$ can thus be parametrized by only two degree of freedoms. We assume that the force exerted by the link on the tricopter is of the form $\vec{s} = s\vec{z}_{\mathcal{P}}$. This force is applied in $-e\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}}$ and adds a torque $e\mathbf{S}(\vec{z}_{\mathcal{B}})s\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}}$ to the angular dynamics of the tricopter. The general layout of this system is depicted in figure 7.25. Following these assumptions, the modified dynamics of the tricopter 6.42 reads:

$$m\mathbf{R}_{\mathcal{I}}^{\mathcal{B}}\left(\ddot{\vec{r}} - g\vec{z}_{\mathcal{I}}\right) = \vec{T} + s\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}} \tag{7.44}$$

$$\mathbf{J}\dot{\vec{\Omega}} + \mathbf{S}(\vec{\Omega})\mathbf{J}\vec{\Omega} = \vec{\Gamma} + es\mathbf{S}(\vec{z}_{\mathcal{B}})\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}} \tag{7.45}$$

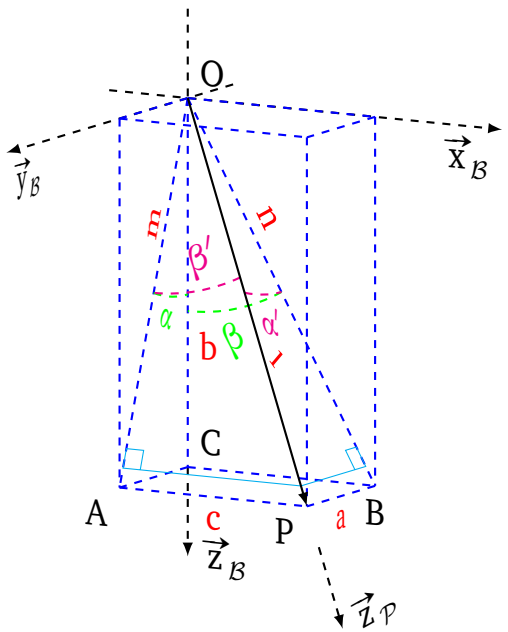


Figure 7.23 – Schematics of the Pendulum. The angles α and β (in green) are the angles measured by the potentiometers. The sequences (α, β') and (β, α') are possible Euler rotation sequences.

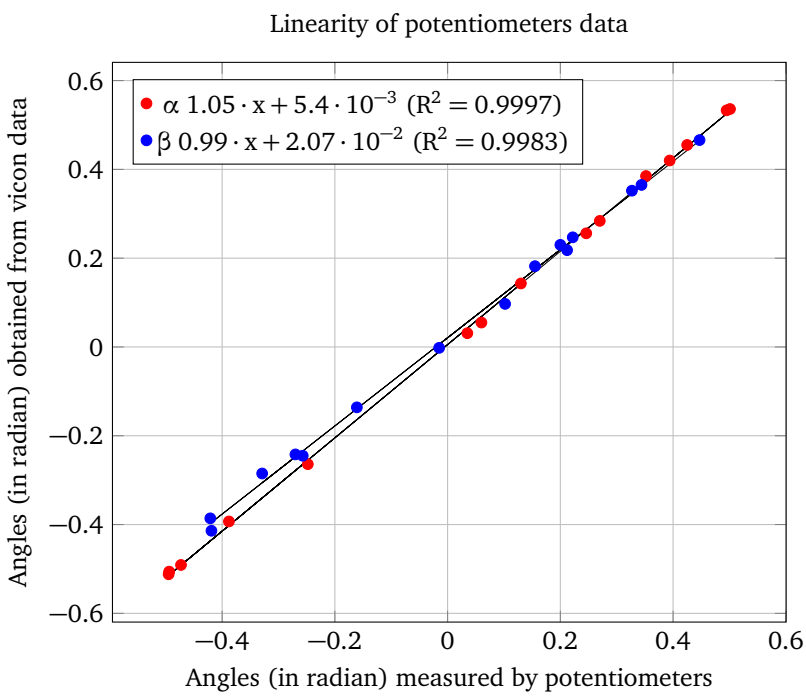


Figure 7.24 – Validation of the parameters of the potentiometers.

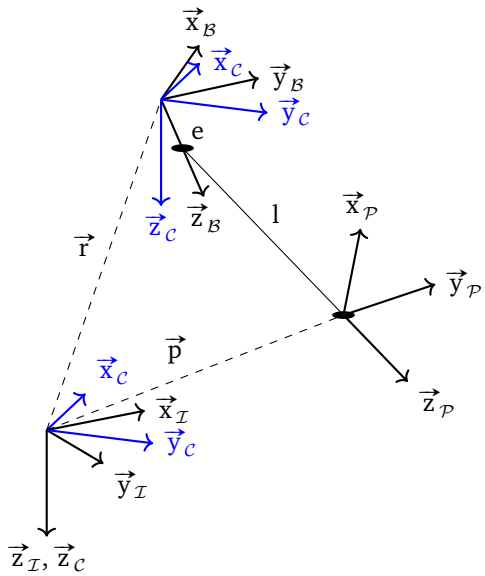


Figure 7.25 – Schematics of the tricopter carrying a pendulum load. The frame depicted in blue corresponds to the intermediate yaw frame.

7.4.2.2 Kinematics of the pendulum

The position of the load is first constrained by the length of the rod. This condition reads:

$$\vec{p} = \vec{r} + e\mathbf{R}_{\mathcal{B}}^{\mathcal{I}}\vec{z}_{\mathcal{B}} + l\mathbf{R}_{\mathcal{P}}^{\mathcal{I}}\vec{z}_{\mathcal{P}} \tag{7.46}$$

We find an expression of $\ddot{\vec{r}}$ with respect to the chosen states by successive derivations of equation (7.46). For this we define the angular velocity of the load with respect to the tricopter $\vec{\omega}$ as $\dot{\mathbf{R}}_{\mathcal{P}}^{\mathcal{B}} = \mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\mathbf{S}(\vec{\omega})$. We rewrite equation (7.46) as:

$$\vec{p} = \vec{r} + \mathbf{R}_{\mathcal{B}}^{\mathcal{I}}(e\vec{z}_{\mathcal{B}} + l\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}}) \tag{7.47}$$

The first time derivative of equation (7.47) reads:

$$\dot{\vec{p}} = \dot{\vec{r}} + \mathbf{R}_{\mathcal{B}}^{\mathcal{I}}\mathbf{S}(\vec{\Omega})(e\vec{z}_{\mathcal{B}} + l\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}}) + l\mathbf{R}_{\mathcal{P}}^{\mathcal{I}}\mathbf{S}(\vec{\omega})\vec{z}_{\mathcal{P}} \tag{7.48}$$

The second derivative of equation (7.47) reads:

$$\begin{aligned}\ddot{\vec{p}} = \ddot{\vec{r}} + \mathbf{R}_{\mathcal{B}}^{\mathcal{I}}(\mathbf{S}^2(\vec{\Omega}) + \mathbf{S}(\dot{\vec{\Omega}}))(e\vec{z}_{\mathcal{B}} + l\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}}) \\ + l\mathbf{R}_{\mathcal{P}}^{\mathcal{I}}(\mathbf{S}^2(\vec{\omega}) + \mathbf{S}(\dot{\vec{\omega}}))\vec{z}_{\mathcal{P}} + 2l\mathbf{R}_{\mathcal{P}}^{\mathcal{I}}\mathbf{S}(\mathbf{R}_{\mathcal{B}}^{\mathcal{P}}\vec{\Omega})\mathbf{S}(\vec{\omega})\vec{z}_{\mathcal{P}}\end{aligned} \tag{7.49}$$

Second, the rotation of the load with respect to the tricopter $\mathbf{R}_{\mathcal{B}}^{\mathcal{P}}$ has only two degrees of liberty. We express this constraints based on a quaternion representation of the rotation. As depicted in figure 7.26, the position of the pendulum load can be expressed as a rotation of angle δ around the axis $\hat{y} = \frac{\vec{z}_{\mathcal{P}} \wedge \overrightarrow{OP}}{\|\vec{z}_{\mathcal{P}} \wedge \overrightarrow{OP}\|}$ (as long as $\delta \neq 0$) of the vector $\vec{z}_{\mathcal{B}}$. Following the formalism defined in [Diebel, 2006], this convert to the quaternion:

$$q = \begin{bmatrix} \cos\frac{1}{2}\delta \\ \hat{y}\sin\frac{1}{2}\delta \end{bmatrix} \tag{7.50}$$

or, writing $\hat{y}$ as $\hat{y} = (-\sin\gamma, \cos\gamma, 0)$, this quaternion reads:

$$q = \begin{bmatrix} \cos\frac{1}{2}\delta \\ -\sin\gamma\sin\frac{1}{2}\delta \\ \cos\gamma\sin\frac{1}{2}\delta \\ 0 \end{bmatrix} \tag{7.51}$$

This is equivalent to the rotation matrix [Diebel, 2006, eq. 125]:

$$\mathbf{R}_{\mathcal{B}}^{\mathcal{P}} = \begin{pmatrix} \sin^2\gamma + \cos^2\gamma\cos\delta & -\sin\gamma\cos\gamma(1-\cos\delta) & -\sin\delta\cos\gamma \\ -\sin\gamma\cos\gamma(1-\cos\delta) & \cos^2\gamma + \sin^2\gamma\cos\delta & -\sin\delta\sin\gamma \\ \sin\delta\cos\gamma & \sin\delta\sin\gamma & \cos\delta \end{pmatrix} \tag{7.52}$$

Using this formalism allows to find an appropriate expression of the angular velocity of the load with respect to the tricopter $\vec{\omega}$. Diebel gives a formula for the angular velocity of the pendulum:

$$\vec{\omega} = 2\begin{pmatrix} -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \end{pmatrix}\dot{q} \tag{7.53}$$

Introducing the matrix:

$$\mathbf{Q} = \begin{pmatrix} \sin\gamma\sin\frac{1}{2}\delta & \cos\frac{1}{2}\delta & 0 & -\cos\gamma\sin\frac{1}{2}\delta \\ -\cos\gamma\sin\frac{1}{2}\delta & 0 & \cos\frac{1}{2}\delta & -\sin\gamma\sin\frac{1}{2}\delta \\ 0 & \cos\gamma\sin\frac{1}{2}\delta & \sin\gamma\sin\frac{1}{2}\delta & \cos\frac{1}{2}\delta \end{pmatrix} \tag{7.54}$$

the angular velocity reads:

$$\frac{\vec{\omega}}{2} = \mathbf{Q}\begin{bmatrix} -\frac{1}{2}\dot{\delta}\sin\frac{1}{2}\delta \\ -\dot{\gamma}\cos\gamma\sin\frac{1}{2}\delta - \frac{1}{2}\dot{\delta}\sin\gamma\cos\frac{1}{2}\delta \\ -\dot{\gamma}\sin\gamma\sin\frac{1}{2}\delta + \frac{1}{2}\dot{\delta}\cos\gamma\cos\frac{1}{2}\delta \\ 0 \end{bmatrix}. \tag{7.55}$$

This may finally read:

$$\vec{\omega} = \begin{pmatrix} -\dot{\delta}\sin\gamma - \dot{\gamma}\cos\gamma\sin\delta \\ \dot{\delta}\cos\gamma - \dot{\gamma}\sin\gamma\sin\delta \\ \dot{\gamma}(\cos\delta - 1) \end{pmatrix} \tag{7.56}$$

Writing $\vec{\omega}$ as $\vec{\omega} = (\omega_x, \omega_y, \omega_z)$, we may express ω_z as:

$$\omega_z = \tan\frac{1}{2}\delta(\cos\gamma\omega_x + \sin\gamma\omega_y) \tag{7.57}$$

Remark. In the case where $\dot{\gamma} = 0$, the angular velocity reads:

$$\vec{\omega} = \begin{pmatrix} -\dot{\delta}\sin\gamma \\ \dot{\delta}\cos\gamma \\ 0 \end{pmatrix} = \hat{y}\dot{\delta}$$

which is the expected result.

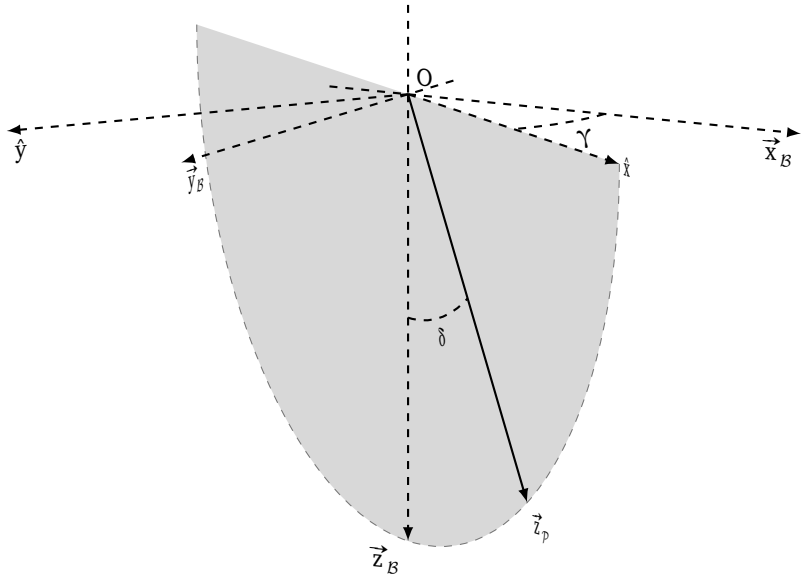


Figure 7.26 – Axis-angle representation of the pendulum load. The pendulum load rotates around axis $\hat{y}$.

Remark. Considering figure 7.23, γ and δ might be obtained as:

$$\begin{cases} \tan\gamma &= \frac{a}{c} \\ \tan\delta &= \frac{\sqrt{a^2+c^2}}{b} \end{cases} \tag{7.58}$$

or, expressed with the angles α and β measured by the potentiometers:

$$\begin{cases} \tan\gamma &= \dfrac{\tan\alpha}{\tan\beta} \\ \tan\delta &= \sqrt{\tan^2\alpha + \tan^2\beta} \end{cases} \tag{7.59}$$

7.4.2.3 Dynamics of the pendulum

Newton equation applied to the load in the inertial reference frame reads:

$$m_L(\ddot{\vec{p}} - \vec{g}) = -s\mathbf{R}_{\mathcal{P}}^{\mathcal{I}}\vec{z}_{\mathcal{P}} \tag{7.60}$$

The angular equivalent of this equation might be found using vectorial product:

$$\mathbf{S}(\mathbf{R}_{\mathcal{P}}^{\mathcal{I}}\vec{z}_{\mathcal{P}})(\ddot{\vec{p}} - \vec{g}) = 0 \tag{7.61}$$

7.4.2.4 Global model of the system

The dynamics of the system is constituted of equations (7.44), (7.45), (7.60) and (7.61) and reads:

$$\begin{cases} m\mathbf{R}_{\mathcal{I}}^{\mathcal{B}}\left(\ddot{\vec{r}}-\vec{g}\right) & = \vec{T}+s\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}} \\ m_L\mathbf{R}_{\mathcal{I}}^{\mathcal{B}}\left(\ddot{\vec{p}}-\vec{g}\right) & = -s\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}} \\ \mathbf{J}\dot{\vec{\Omega}}+\mathbf{S}(\vec{\Omega})\mathbf{J}\vec{\Omega} & = \vec{\Gamma}+es\mathbf{S}(\vec{z}_{\mathcal{B}})\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}} \\ \mathbf{S}(\mathbf{R}_{\mathcal{P}}^{\mathcal{I}}\vec{z}_{\mathcal{P}})(\ddot{\vec{p}}-\vec{g}) & = 0 \end{cases} \tag{7.62}$$

We first try to eliminate the tension s in the expression of the system. Using the second line, we may rewrite the model as:

$$\begin{cases} \mathbf{R}_{\mathcal{I}}^{\mathcal{B}}\left(m\ddot{\vec{r}}+m_L\ddot{\vec{p}}\right) & = \vec{T}+\mathbf{R}_{\mathcal{I}}^{\mathcal{B}}(m+m_L)\vec{g} \\ m_L\mathbf{R}_{\mathcal{I}}^{\mathcal{B}}\left(\ddot{\vec{p}}-\vec{g}\right) & = -s\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}} \\ \mathbf{J}\dot{\vec{\Omega}}+\mathbf{S}(\vec{\Omega})\mathbf{J}\vec{\Omega} & = \vec{\Gamma}-em_L\mathbf{S}(\vec{z}_{\mathcal{B}})\mathbf{R}_{\mathcal{I}}^{\mathcal{B}}(\ddot{\vec{p}}-\vec{g}) \\ \mathbf{S}(\mathbf{R}_{\mathcal{P}}^{\mathcal{I}}\vec{z}_{\mathcal{P}})(\ddot{\vec{p}}-\vec{g}) & = 0 \end{cases} \tag{7.63}$$

In the preceding system, only the first, third and fourth equations are needed to describe the model. The second equation gives only an expression of the tension. The model is totally described by the system:

$$\begin{cases} \mathbf{R}_{\mathcal{I}}^{\mathcal{B}}\left(m\ddot{\vec{r}}+m_L\ddot{\vec{p}}\right) & = \vec{T}+\mathbf{R}_{\mathcal{I}}^{\mathcal{B}}(m+m_L)\vec{g} \\ \mathbf{J}\dot{\vec{\Omega}}+\mathbf{S}(\vec{\Omega})\mathbf{J}\vec{\Omega} & = \vec{\Gamma}-em_L\mathbf{S}(\vec{z}_{\mathcal{B}})\mathbf{R}_{\mathcal{I}}^{\mathcal{B}}(\ddot{\vec{p}}-\vec{g}) \\ \mathbf{S}(\mathbf{R}_{\mathcal{P}}^{\mathcal{I}}\vec{z}_{\mathcal{P}})(\ddot{\vec{p}}-\vec{g}) & = 0 \end{cases} \tag{7.64}$$

Nonetheless, this description is implicit. Therefore, we try to make the equations explicit. Using equation (7.49), the first line of equation (7.63) reads:

$$\begin{aligned} M\mathbf{R}_{\mathcal{I}}^{\mathcal{B}}\left(\ddot{\vec{p}}-\vec{g}\right) = \vec{T}+m(\mathbf{S}^2(\vec{\Omega})+\mathbf{S}(\dot{\vec{\Omega}}))(e\vec{z}_{\mathcal{B}}+l\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}}) \\ +ml\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}(\mathbf{S}^2(\vec{\omega})+\mathbf{S}(\dot{\vec{\omega}}))\vec{z}_{\mathcal{P}}+2ml\mathbf{S}(\vec{\Omega})\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\mathbf{S}(\vec{\omega})\vec{z}_{\mathcal{P}} \end{aligned} \tag{7.65}$$

where $M = m + m_L$ is the total mass of the tricopter together with its pendulum load. We used also the relation:

$$\mathbf{S}(\mathbf{R}\vec{u}) = \mathbf{R}\mathbf{S}(\vec{u})\mathbf{R}^T \tag{7.66}$$

for any rotation matrix $\mathbf{R}$ and vector $\vec{u}$. The two angular equations – the second and the third – can be written in term of angular variables only. Multiplying equation (7.65) to the left by $\mathbf{S}(\vec{z}_{\mathcal{B}})$ leads to :

$$\begin{aligned} \mathbf{S}(\vec{z}_{\mathcal{B}})\mathbf{R}_{\mathcal{I}}^{\mathcal{B}}\left(\ddot{\vec{p}}-\vec{g}\right) = \frac{1}{M}\mathbf{S}(\vec{z}_{\mathcal{B}})\vec{T}+\frac{m}{M}\mathbf{S}(\vec{z}_{\mathcal{B}})(\mathbf{S}^2(\vec{\Omega})+\mathbf{S}(\dot{\vec{\Omega}}))(e\vec{z}_{\mathcal{B}}+l\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}}) \\ +\frac{ml}{M}\mathbf{S}(\vec{z}_{\mathcal{B}})\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}(\mathbf{S}^2(\vec{\omega})+\mathbf{S}(\dot{\vec{\omega}}))\vec{z}_{\mathcal{P}}+2\frac{ml}{M}\mathbf{S}(\vec{z}_{\mathcal{B}})\mathbf{S}(\vec{\Omega})\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\mathbf{S}(\vec{\omega})\vec{z}_{\mathcal{P}} \end{aligned} \tag{7.67}$$

and the second line of equation (7.64) reads:

$$\begin{aligned} \mathbf{J}\dot{\vec{\Omega}}+\mathbf{S}(\vec{\Omega})\mathbf{J}\vec{\Omega}+e\frac{mm_L}{M}\mathbf{S}(\vec{z}_{\mathcal{B}})(\mathbf{S}^2(\vec{\Omega})+\mathbf{S}(\dot{\vec{\Omega}}))(e\vec{z}_{\mathcal{B}}+l\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}}) = \vec{\Gamma}-e\frac{m_L}{M}\mathbf{S}(\vec{z}_{\mathcal{B}})\vec{T} \\ -el\frac{mm_L}{M}\mathbf{S}(\vec{z}_{\mathcal{B}})\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}(\mathbf{S}^2(\vec{\omega})+\mathbf{S}(\dot{\vec{\omega}}))\vec{z}_{\mathcal{P}}-2el\frac{mm_L}{M}\mathbf{S}(\vec{z}_{\mathcal{B}})\mathbf{S}(\vec{\Omega})\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\mathbf{S}(\vec{\omega})\vec{z}_{\mathcal{P}} \end{aligned} \tag{7.68}$$

Going back to equation (7.65), the third line of equation (7.64) reads:

$$\begin{aligned} &\mathrm{ml}\omega_z \mathbf{S}(\vec{z}_{\mathcal{P}})\vec{\omega} + \mathrm{ml}\mathbf{S}^2(\vec{z}_{\mathcal{P}})\dot{\vec{\omega}} = \mathbf{S}(\vec{z}_{\mathcal{P}})\mathbf{R}_{\mathcal{B}}^{\mathcal{P}}\vec{T} \\ &\qquad + \mathrm{m}\mathbf{S}(\vec{z}_{\mathcal{P}})\mathbf{R}_{\mathcal{B}}^{\mathcal{P}}(\mathbf{S}^2(\vec{\Omega}) + \mathbf{S}(\dot{\vec{\Omega}}))(\mathrm{e}\vec{z}_{\mathcal{B}} + \mathrm{l}\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}}) + 2\mathrm{ml}\mathbf{S}(\vec{z}_{\mathcal{P}})\mathbf{S}(\mathbf{R}_{\mathcal{B}}^{\mathcal{P}}\vec{\Omega})\mathbf{S}(\vec{\omega})\vec{z}_{\mathcal{P}} \end{aligned} \tag{7.69}$$

We may notice that $\mathbf{S}^3(\vec{z}_{\mathcal{P}})\vec{u} = -\mathbf{S}(\vec{z}_{\mathcal{P}})\vec{u}$, so that:

$$\begin{aligned} &\mathrm{ml}\mathbf{S}(\dot{\vec{\omega}})\vec{z}_{\mathcal{P}} = \mathbf{S}^2(\vec{z}_{\mathcal{P}})\mathbf{R}_{\mathcal{B}}^{\mathcal{P}}\vec{T} - \mathrm{ml}\omega_z\mathbf{S}^2(\vec{z}_{\mathcal{P}})\vec{\omega} \\ &\qquad + \mathrm{m}\mathbf{S}^2(\vec{z}_{\mathcal{P}})\mathbf{R}_{\mathcal{B}}^{\mathcal{P}}(\mathbf{S}^2(\vec{\Omega}) + \mathbf{S}(\dot{\vec{\Omega}}))(\mathrm{e}\vec{z}_{\mathcal{B}} + \mathrm{l}\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}}) + 2\mathrm{ml}\mathbf{S}^2(\vec{z}_{\mathcal{P}})\mathbf{S}(\mathbf{R}_{\mathcal{B}}^{\mathcal{P}}\vec{\Omega})\mathbf{S}(\vec{\omega})\vec{z}_{\mathcal{P}} \end{aligned} \tag{7.70}$$

We introduce the matrix $\mathbf{B} = \mathbf{S}(\vec{z}_{\mathcal{B}}) + \mathbf{S}(\vec{z}_{\mathcal{B}})\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\mathbf{S}^2(\vec{z}_{\mathcal{P}})\mathbf{R}_{\mathcal{B}}^{\mathcal{P}}$. Injecting the previous equation into equation (7.68) leads to:

$$\begin{aligned} &\mathbf{J}\dot{\vec{\Omega}} + \mathbf{S}(\vec{\Omega})\mathbf{J}\vec{\Omega} + \mathrm{e}\frac{\mathrm{mm_L}}{\mathrm{M}}\mathbf{B}(\mathbf{S}^2(\vec{\Omega}) + \mathbf{S}(\dot{\vec{\Omega}}))(\mathrm{e}\vec{z}_{\mathcal{B}} + \mathrm{l}\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}}) = \vec{\Gamma} - \mathrm{e}\frac{\mathrm{m_L}}{\mathrm{M}}\mathbf{B}\vec{T} \\ &\qquad + \mathrm{el}\frac{\mathrm{mm_L}}{\mathrm{M}}\mathbf{S}(\vec{z}_{\mathcal{B}})\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\left(\omega_z\mathbf{S}^2(\vec{z}_{\mathcal{P}})\vec{\omega} - \mathbf{S}^2(\vec{\omega})\vec{z}_{\mathcal{P}}\right) - 2\mathrm{el}\frac{\mathrm{mm_L}}{\mathrm{M}}\mathbf{B}\mathbf{S}(\vec{\Omega})\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\mathbf{S}(\vec{\omega})\vec{z}_{\mathcal{P}} \end{aligned} \tag{7.71}$$

We notice that:

$$\left(\mathbf{S}(\vec{z}_{\mathcal{B}}) + \mathbf{S}(\vec{z}_{\mathcal{B}})\mathbf{S}^2(\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}})\right)\mathbf{S}(\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}}) = \mathbf{B}\mathbf{S}(\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}}) = 0 \tag{7.72}$$

So that the previous equation can be written as:

$$\begin{aligned} &\left(\mathbf{J} - \mathrm{e}^2\frac{\mathrm{mm_L}}{\mathrm{M}}\mathbf{B}\mathbf{S}(\vec{z}_{\mathcal{B}})\right)\dot{\vec{\Omega}} + \mathbf{S}(\vec{\Omega})\mathbf{J}\vec{\Omega} + \mathrm{e}\frac{\mathrm{mm_L}}{\mathrm{M}}\mathbf{B}\mathbf{S}^2(\vec{\Omega})(\mathrm{e}\vec{z}_{\mathcal{B}} + \mathrm{l}\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}}) = \vec{\Gamma} - \mathrm{e}\frac{\mathrm{m_L}}{\mathrm{M}}\mathbf{B}\vec{T} \\ &\qquad + \mathrm{el}\frac{\mathrm{mm_L}}{\mathrm{M}}\mathbf{S}(\vec{z}_{\mathcal{B}})\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\left(\omega_z\mathbf{S}^2(\vec{z}_{\mathcal{P}})\vec{\omega} - \mathbf{S}^2(\vec{\omega})\vec{z}_{\mathcal{P}}\right) - 2\mathrm{el}\frac{\mathrm{mm_L}}{\mathrm{M}}\mathbf{B}\mathbf{S}(\vec{\Omega})\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\mathbf{S}(\vec{\omega})\vec{z}_{\mathcal{P}} \end{aligned} \tag{7.73}$$

Therefore, we introduce the inertia of the global system and prove the following result:

Proposition 7.4.1. *The inertia of the global system:*

$$\mathbf{J}(\mathbf{R}_{\mathcal{B}}^{\mathcal{P}}) = \left(\mathbf{J} - \mathrm{e}^2\frac{\mathrm{mm_L}}{\mathrm{M}}\left(\mathbf{S}^2(\vec{z}_{\mathcal{B}}) + \mathbf{S}(\vec{z}_{\mathcal{B}})\mathbf{S}^2(\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}})\mathbf{S}(\vec{z}_{\mathcal{B}})\right)\right) \tag{7.74}$$

is invertible for any orientation $\mathbf{R}_{\mathcal{B}}^{\mathcal{P}}$ *of the pendulum load and for any mass* $\mathrm{m_L}$ *and offset* e.

Proof. Using the formalism of equation (7.52), when $\vec{z}_{\mathcal{P}}$ is aligned with $\vec{z}_{\mathcal{B}}$, the angle δ is null and $\mathbf{R}_{\mathcal{B}}^{\mathcal{P}} = \mathbf{I}$. The additional inertia term is then null and the inertia matrix is invertible. Furthermore, there exists a neighborhood of $\delta = 0$ where $\mathbf{J}(\mathbf{R}_{\mathcal{B}}^{\mathcal{P}})$ is invertible. For any values of δ and γ, the varying term of the inertia matrix depends on:

$$\mathbf{S}^2(\vec{z}_{\mathcal{B}}) + \mathbf{S}(\vec{z}_{\mathcal{B}})\mathbf{S}^2(\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}\vec{z}_{\mathcal{P}})\mathbf{S}(\vec{z}_{\mathcal{B}}) = \sin^2(\delta)\begin{pmatrix} -\sin^2(\gamma) & \sin(\gamma)\cos(\gamma) & 0 \\ \sin(\gamma)\cos(\gamma) & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{7.75}$$

This matrix is real symmetric. The new inertia matrix $\mathbf{J}(\mathbf{R}_{\mathcal{B}}^{\mathcal{P}})$ is then also real symmetric and thus diagonalizable. We notice that this additional matrix is diagonal for $\gamma \in \{0, \pm\frac{\pi}{2}, \pi\}$. To ease the analysis, we neglect the non-diagonal terms of the matrix $\mathbf{J}$ and study the matrix in the xy plane. Then the eigenvalues of the upper left block of the inertia matrix $\mathbf{J}(\mathbf{R}_{\mathcal{B}}^{\mathcal{P}})\dot{\vec{\Omega}}$ are $\{\mathrm{J_{xx}} + \mathrm{e}^2\frac{\mathrm{mm_L}}{\mathrm{M}}\sin^2(\delta), \mathrm{J_{xx}}\}$. The first eigenvalue does not depend on γ and is always strictly

positive. When considering the non-diagonal terms, the two first eigenvalues of the inertia matrix are:

$$J_{xx} + \frac{\varkappa}{2}\sin^2(\delta) \pm \sqrt{J_{xy}^2 - 2J_{xy}\varkappa\sin^2(\delta)\sin(\gamma)\cos(\gamma) + \frac{\varkappa^2}{4}\sin^4(\delta)} \tag{7.76}$$

where $\varkappa = e^2\frac{mm_L}{M}$. The term under the square root verifies:

$$\begin{aligned} J_{xy}^2 - 2J_{xy}\varkappa\sin^2(\delta)\sin(\gamma)\cos(\gamma) + \frac{\varkappa^2}{4}\sin^4(\delta) &< J_{xy}^2 + 2J_{xy}\varkappa\sin^2(\delta) + \frac{\varkappa^2}{4}\sin^4(\delta) \\ &< \left(\frac{\varkappa}{2}\sin^2(\delta) + J_{xy}\right)^2 \end{aligned} \tag{7.77}$$

This leads to eigenvalues λ_1, λ_2 between $J_{xx} - J_{xy} < \lambda_1, \lambda_2 < J_{xx} + J_{xy} + e^2\frac{mm_L}{M}\sin^2(\delta)$ which is in any case positive. Therefore, the matrix $\mathbf{J}(\mathbf{R}_{\mathcal{B}}^{\mathcal{P}})\vec{\Omega}$ is always invertible. □

Finally, we may introduce the functions Ψ, Φ, Ξ to write the system in the form:

$$\begin{cases} \dot{\vec{\Omega}} &= \Psi(\vec{\Gamma}, \vec{T}, \mathbf{R}_{\mathcal{B}}^{\mathcal{P}}, \vec{\omega}, \vec{\Omega}) \\ \dot{\vec{\omega}} &= \Phi(\vec{\Gamma}, \vec{T}, \mathbf{R}_{\mathcal{B}}^{\mathcal{P}}, \vec{\omega}, \vec{\Omega}) \\ \ddot{\vec{p}} &= \Xi(\vec{\Gamma}, \vec{T}, \mathbf{R}_{\mathcal{B}}^{\mathcal{P}}, \vec{\omega}, \mathbf{R}_{\mathcal{I}}^{\mathcal{B}}, \vec{\Omega}) \end{cases} \tag{7.78}$$

In this form, the system reminds us of flat systems. When building a flat output, it appears from equation (7.78) that the position of the pendulum $\vec{p}$ only appears as its second derivative in the third equation. The position of the pendulum load should then be chosen as a part of the flat output. The remaining components of the chosen flat outputs are given in the following result:

Proposition 7.4.2. *The system composed of the tricopter with the pendulum load described by equation (7.78) is flat with a flat output given by* $(\vec{p}, \mathbf{R}_{\mathcal{B}}^{\mathcal{P}}, \mathbf{R}_{\mathcal{I}}^{\mathcal{C}})$ *where* $\mathbf{R}_{\mathcal{B}}^{\mathcal{P}}$ *has two degrees of freedom (the angles* γ *and* δ *in figure 7.26) and* $\mathbf{R}_{\mathcal{I}}^{\mathcal{C}}$ *has one (the yaw angle as depicted in figure 7.25).*

Proof. The tension in the link is first obtained based on equation (7.60) as:

$$m_L\|\ddot{\vec{p}} - \vec{g}\| = |s| \tag{7.79}$$

In the following, we admit that s is strictly positive. That is, we assume that the load never experiences free fall. In this case, both the load and the tricopter are independent flat systems forming a differentially flat hybrid system [Sreenath et al., 2013]. The case where s is strictly negative is similar to the positive case. We define the two following vectors, first:

$$\vec{t} = \mathbf{R}_{\mathcal{I}}^{\mathcal{C}}\frac{\ddot{\vec{p}} - \vec{g}}{\|\ddot{\vec{p}} - \vec{g}\|} \tag{7.80}$$

According to equation (7.60), we have:

$$\vec{t} = \mathbf{R}_{\mathcal{P}}^{\mathcal{C}}\vec{z}_{\mathcal{P}} \tag{7.81}$$

m_L	=	2.5×10^{-1} kg
l	=	2.0×10^{-1} m
e	=	5.0×10^{-2} m

Table 7.3 – Pendulum load parameters used in the open-loop simulations.

In other words, $\vec{t}$ represents the position of the pendulum load expressed in the yaw reference frame but depends only upon $\mathbf{R}_{\mathcal{I}}^{\mathcal{C}}$ and $\vec{p}$. The second vector is:

$$\vec{u} = \mathbf{R}_{\mathcal{P}}^{\mathcal{B}} \vec{z}_{\mathcal{P}} \tag{7.82}$$

which is the position of the load in the reference frame of the tricopter. These two vectors are linked by equation (7.60). Using the previously introduced vectors, this equation reads:

$$\vec{t} = \mathbf{R}_{\mathcal{B}}^{\mathcal{C}} \vec{u} \tag{7.83}$$

In other words, we have the same vector ($\vec{z}_{\mathcal{P}}$) expressed in two different bases ($\mathcal{C}$ and $\mathcal{B}$) and we want to construct a compatible rotation matrix $\mathbf{R}_{\mathcal{C}}^{\mathcal{B}}$. The first lemma in [Piovan and Bullo, 2012] states that every solution to this equation in $\mathbf{R}_{\mathcal{B}}^{\mathcal{C}}$ is written as:

$$\mathbf{R} = \exp(\beta \mathbf{S}(\vec{t})) \exp(\alpha \mathbf{S}(\vec{v})) \tag{7.84}$$

where $\exp(\alpha \mathbf{S}(\vec{v}))$ is a rotation of angle α around axis $\vec{v}$. The angle β is arbitrary in $[-\pi, \pi[$ and the angle $\alpha \in [0, \pi]$ and the unit-length vector $\vec{v}$ are defined by:

$$\begin{aligned} \alpha &= \operatorname{atan2}(\|\vec{t} \times \vec{u}\|, \vec{t}^{\mathrm{T}} \vec{u}), \\ \vec{v} &= \begin{cases} \frac{\vec{t} \times \vec{u}}{\|\vec{t} \times \vec{u}\|} & \text{if } \vec{t} \times \vec{u} \neq 0, \\ \text{any unit-length vector} \perp \vec{t}, & \text{otherwise} \end{cases} \end{aligned} \tag{7.85}$$

The degree of liberty β can be uniquely determined for example as in [Konz and Rudolph, 2013] to minimize a chosen value. This allows to compute $\mathbf{R}_{\mathcal{B}}^{\mathcal{C}}$. Using the knowledge of $\mathbf{R}_{\mathcal{P}}^{\mathcal{B}}$, $\mathbf{R}_{\mathcal{B}}^{\mathcal{C}}$ and $\mathbf{R}_{\mathcal{C}}^{\mathcal{I}}$ and considering equation (7.47), one can compute $\vec{r}$. The angular velocity $\vec{\Omega}$ is obtained as $(\mathbf{R}_{\mathcal{I}}^{\mathcal{B}} \dot{\mathbf{R}}_{\mathcal{B}}^{\mathcal{I}})^{\vee}$. The controls $\vec{T}$ and $\vec{\Gamma}$ can then be computed. □

Remark. In comparison to the flat output proposed in [Sreenath et al., 2013], the orientation of the link with respect to the UAV is added resulting in the creation of trajectories compatible with the link mechanism.

This flat output allows to compute open-loop controls for the tricopter. We show this possibility by computing such controls for the following application. The pendulum has to track the trajectory of figure 7.3 with the tricopter tracking the same yaw. The pendulum load has to remain vertical with respect to the tricopter (*i.e.* $\mathbf{R}_{\mathcal{P}}^{\mathcal{B}} = \mathbf{I}$) with the parametrization $\beta = 0$ and the physical parametres given in table 7.3. This result in the trajectory depicted in figure 7.27.

The corresponding open-loop controls are given in figure 7.28. It appears that these controls are compatible with the actuators saturations as they do not exceed the maximum values given in table 6.1 on page 120. This open-loop feedforward shall be completed by a closed-loop

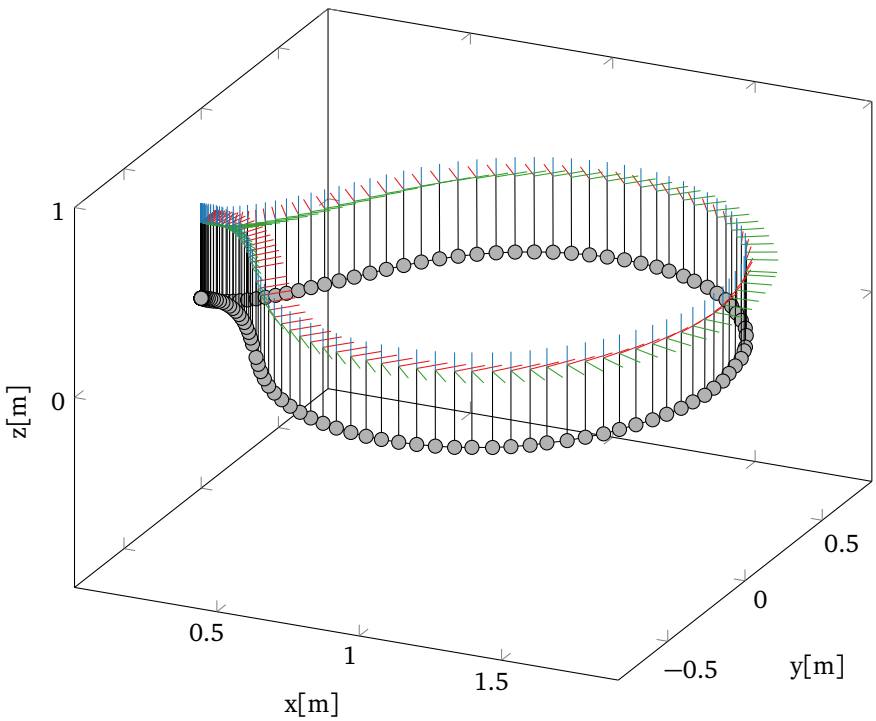


Figure 7.27 – Trajectory followed by the tricopter with pendulum load. The orientation of the tricopter is given by the blue axis (yaw), green (roll), red (pitch). The load is symbolized by the grey circle. (timestamp: $\Delta t = 1.0 \times 10^{-1}$ s).

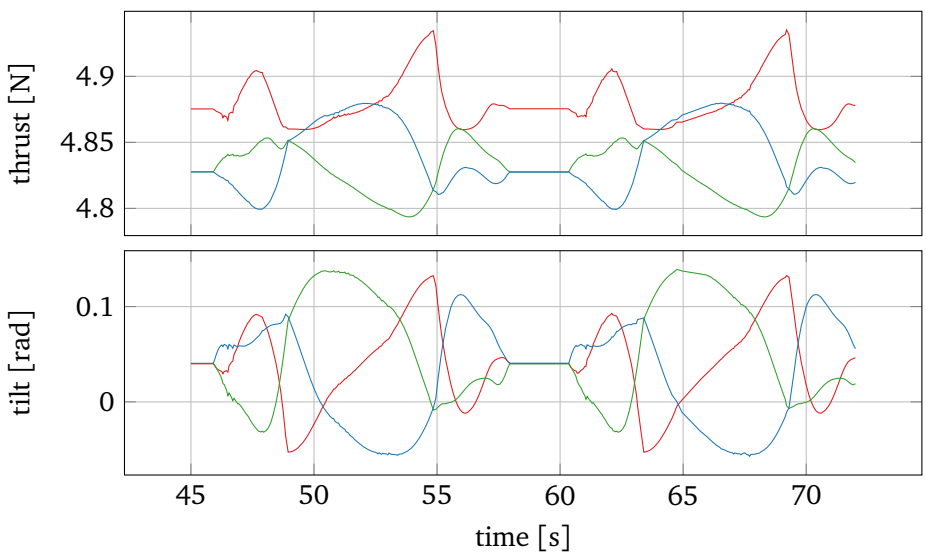


Figure 7.28 – Feedforward for the thrusts (upper part) and the tilt angles (lower part) when flying the pendulum on two revolutions of the trajectory depicted in figure 7.27. The controls for nacelle 1 are plotted in red, green for nacelle 2 and blue for nacelle 3.

controller to correct the deviations from the model. However, this closed-loop controller is still to be constructed. Furthermore, a better construction of the parameter β, for example to minimize the inclination of the tricopter with respect to the ground, could prove to be useful. In the example depicted in figure 7.27, this parametrization results in an appropriate attitude. With more aggressive trajectories, this parametrization could prove to be unsatisfying. Nonetheless, the reasonable open-loop controls validate the chosen approach.

7.5 Perspectives

In this chapter, we presented a flatness-based closed-loop controller and its application to the control of the tricopter in two scenarios. The first application, where the tricopter had to follow a trajectory on the ground rolling on ball casters is a great illustration of the total actuation of the platform. It has allowed to carefully test the platform and, especially, the controller for any defaults. Furthermore, this approach is interesting for indoor exploration, when the ground is compatible with the ball casters, as it appears to be particularly efficient in term of displacement. In our case, we used the third of the energy needed in the case of flight. The controller has proved to be efficient in terms of trajectory tracking, both in term of orientation as in term of position.

The second application is the “natural” case of flight. The controller proved to correctly stabilize the tricopter but some minor flaws could be corrected in the future. For example, adding an integral term to the position tracker could help eliminating the steady-state error. Furthermore, as our test flights were conducted in a small test room, the tricopter was particularly affected by turbulences. Notably, the altitude tracker was perturbed by the ground effect and autonomous take-off was not possible. A better comprehension or correction of these various aerodynamic effects would help increasing the efficiency of the control approach.

In the last scenario, the tricopter is equipped with a pendulum load. A precise model is introduced taking into account the influence of the load on the tricopter. A flatness-based open-loop control approach is suggested for the load to track an absolute reference trajectory with respect to the ground while tracking a reference trajectory relative to the tricopter. This approach could benefit from various improvements and studies. Notably, the parametrization of the roll-pitch matrix of the tricopter could be improved in the proof of the flatness of the system, following what was done for example by [Konz and Rudolph, 2013]. The open-loop controller could be used as a first step toward a closed-loop controller. Furthermore, other trajectories and application cases could be studied, for example going through a window smaller than the pendulum length or the cooperative transportation of the pendulum load.

Conclusion

The aim of this thesis was to investigate and present a trajectory planning framework and the control of collaborative systems applied to a special case: a novel trirotor UAV.

In the first part of the thesis, we created a framework to generate trajectories for multi-agent systems. After a review of various existing solutions of control of collaborative systems, the case of solution generation for certain partial differential equation was examined.

Our work relies on Burgers' equation, a classical one-dimensional and nonlinear equation of fluid dynamics. Solutions to Burgers' equation might be transformed with the help of Hopf-Cole transformation into solutions to the heat equation. We prove, under certain assumptions, that the heat equation with control on both sides is flat. Therefore, we may create solutions to the heat equation by choosing the trajectories of the two agents on the formation edge. We prove various results on the admissible structure of the controls and show that these controls can be used to drive the heat equation in finite time between various states, notably, to null-state. In a second time, we use particles swarm optimization to optimize the controls and create an adapted solution to Burgers' equation. This solution defines the trajectories of the collaborative system. This system is composed of leaders and followers. The positions of leaders can be freely chosen and set as constraints of the optimizer. The positions of followers are consequences of the positions of the leaders and of the optimization criteria. In our framework, the trajectories are constructed such that the trajectories of the leaders are the shortest possible. We combine several one-dimensional trajectories into multi-dimensional trajectories and show an example in two dimensions for two leaders with nineteen followers.

In the second part of the thesis, we consider the problem of tracking the generated trajectories by UAVs. After a review of aerodynamics with a special emphasis on multirotor UAVs, we present various current problems in the field of UAVs such as pendulum load transportation and cooperative transportation.

Our work is dedicated to the special case of a trirotor UAV: the tricopter. This design is the result of studies conducted under the supervision of Prof. Rudolph at CSTCE (Saarland University, Germany). It has the particular advantage of six independent controls. Therefore, it is totally actuated and may track any reasonable trajectory. We present the architecture of the tricopter and our work in modeling and simulating this platform. A control approach is presented based on flatness of this dynamical system and a closed-loop trajectory tracker is presented.

Energy efficiency of UAVs is an ongoing problem. A possible solution in the case of indoor exploration is to let the UAV roll on the ground to spare vertical thrust. This is particularly easy for our totally-actuated tricopter. Thus, the controller is first tested on the ground and tracks a determined path and yaw angles. The results of the experiments are conclusive and the tricopter is then tested with the tracking of aerial trajectories. The controller performs accurately in this case also. A last application is suggested with the transportation of a pendulum load. We show that the tricopter carrying a pendulum load is flat. We present an open-loop controller to travel the desired trajectories and present the load-orientation measurement unit that will be necessary to perform closed-loop tracking of the trajectories.

In conclusion, this thesis presents a complete and coherent framework for collaborative

systems. A higher-level trajectory planner is introduced and completed with the control of the lower-level agent: a totally-actuated trirotor UAV. Nonetheless, various improvements might be foreseen. For example, in the case of the higher-level trajectory planner, the optimizer could be adapted to allow collision or obstacle avoidance. Furthermore, the dynamics of the agents could be taken into account, for example using different viscosities (the μ parameter) or control stiffnesses (the γ parameter) for vertical dynamics and lateral dynamics.

The control of the lower-level agent could also be improved. Autonomous takeoff and landing is one of the most obvious improvements that could be added. That would necessitate a better investigation and modeling of various aerodynamic effects. For example, the ground effect could be taken into account, using the model suggested by [Johnson, 1994]:

$$\frac{T}{T_\infty} = \frac{1}{1-\alpha/z^2}.$$

or the more sophisticated models suggested therein. Moreover, an integral term could be added to the position tracker to improve the quality of the tracking in flight. Last but not least, the dynamics of the servomotors and propellers could be taken into account in the model. This would lead to a higher order dynamical model that would more accurately describe the tricopter.

Regarding the problem of pendulum load transportation, one of the first improvements is to suggest a closed-loop controller. It could be also interesting to propose a controller for an inverted pendulum that would use only the rotational dynamics of the tricopter while stabilizing the position. This is possible with our platform and is reflected in the equation of motion of the model. This would be an interesting improvement over the current UAVs carrying an inverted pendulum. Ultimately, suggesting a cooperative load transportation framework using the tricopter would be an impressive breakthrough!

Le vent se lève, il est temps de vivre...

Bibliography

Abramowitz, M. and Stegun, I. A. (1965). Handbook of mathematical functions: with formulas, graphs, and mathematical tables, vol. 55,. Dover publications. Cited on pages 56, 57, 59, 61, 64, 70, 71 et 72.

Adams, Z., Benedict, M., Hrishikeshavan, V. and Chopra, I. (2013). Design, Development, and Flight Test of a Small-Scale Cyclogyro UAV Utilizing a Novel Cam-Based Passive Blade Pitching Mechanism. International Journal of Micro Air Vehicles *5*, 145–162. Cited on page 103.

Akiyama, S. and Tanigawa, Y. (2001). Multiple zeta values at non-positive integers. The Ramanujan Journal *5*, 327–351. Cited on page 62.

Alexis, K., Nikolakopoulos, G. and Tzes, A. (2010). Design and experimental verification of a Constrained Finite Time Optimal control scheme for the attitude control of a Quadrotor Helicopter subject to wind gusts. In Robotics and Automation (ICRA), 2010 IEEE International Conference on pp. 1636–1641, IEEE. Cited on pages 107 et 111.

Amiri, N., Serrano, A. R. and Davies, R. (2011). Modelling of opposed lateral and longitudinal tilting dual-fan unmanned aerial vehicle. In 18th IFAC World Congress vol. 28, pp. 2054–2059,. Cited on page 116.

Balch, T. and Arkin, R. C. (1998). Behavior-based formation control for multirobot teams. Robotics and Automation, IEEE Transactions on *14*, 926–939. Cited on page 21.

Beard, R. W., Lawton, J., Hadaegh, F. Y. et al. (2001). A coordination architecture for spacecraft formation control. IEEE Transactions on control systems technology *9*, 777–790. Cited on pages 3, 18 et 32.

Bisgaard, M., la Cour-Harbo, A. and Bendtsen, J. (2009). Guidance, Navigation, and Control and Co-located Conferences chapter Swing Damping for Helicopter Slung Load Systems Using Delayed Feedback. American Institute of Aeronautics and Astronautics. Cited on pages 8 et 113.

Bouabdallah, S., Noth, A. and Siegwart, R. (2004). PID vs LQ control techniques applied to an indoor micro quadrotor. In Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, vol. 3,, pp. 2451–2456, IEEE. Cited on pages 8, 107 et 110.

Bouabdallah, S. and Siegwart, R. (2005). Backstepping and Sliding-mode Techniques Applied to an Indoor Micro Quadrotor. In Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on pp. 2247–2252, IEEE. Cited on pages 8 et 110.

Bouabdallah, S. and Siegwart, R. (2007). Full control of a quadrotor. In Intelligent robots and systems, 2007. IROS 2007. IEEE/RSJ international conference on pp. 153–158, IEEE. Cited on pages 8 et 110.

Bullo, F. and Murray, R. M. (1999). Tracking for fully actuated mechanical systems: a geometric framework. Automatica *35*, 17–34. Cited on page 137.

Chamseddine, A., Li, T., Zhang, Y., Rabbath, C. A. and Theilliol, D. (2012). Flatness-based trajectory planning for a quadrotor Unmanned Aerial Vehicle test-bed considering actuator and system constraints. In American Control Conference (ACC), 2012 pp. 920–925,. Cited on page 111.

Chamseddine, A., Theilliol, D., Sadeghzadeh, I., Zhang, Y. and Weber, P. (2014). Optimal reliability design for over-actuated systems based on the MIT rule: Application to an octocopter helicopter testbed . Reliability Engineering & System Safety *132*, 196–206. Cited on page 107.

Chang, D. E., Shadden, S. C., Marsden, J. E. and Olfati-Saber, R. (2003). Collision avoidance for multiple agent systems. In Decision and Control, 2003. Proceedings. 42nd IEEE Conference on IEEE. Cited on page 26.

Chartrand, R. (2011). Numerical differentiation of noisy, nonsmooth data. ISRN Applied Mathematics *2011*. Cited on pages 11, 145, 146 et 147.

Cole, J. D. et al. (1951). On a quasi-linear parabolic equation occurring in aerodynamics. Quart. Appl. Math *9*, 225–236. Cited on page 44.

Coron, J. M. (2007). Some open problems on the control of nonlinear partial differential equations. In Perspectives in Nonlinear Partial Differential Equations: In Honor of Haim Brezis (Henri Berestycki, Michiel Bertsch, Bert Peletier and Laurent Véron eds.) vol. 446, pp. 215–243. American Mathematical Society. Cited on pages 46 et 47.

Couceiro, M. S., Rocha, R. P. and Ferreira, N. M. F. (2011). A novel multi-robot exploration approach based on Particle Swarm Optimization algorithms. In Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on pp. 327–332,. Cited on page 24.

Cowling, I. D., Yakimenko, O. A., Whidborne, J. F. and Cooke, A. K. (2007). A prototype of an autonomous controller for a quadrotor UAV. In European Control Conference pp. 1–8,. Cited on pages 8 et 110.

Crépeau, E. and Prieur, C. (2008). Approximate controllability of a reaction-diffusion system. Systems & Control Letters *57*, 1048–1057. Cited on page 48.

Dai, S., Lee, T. and Bernstein, D. S. (2014). Adaptive Control of a Quadrotor UAV Transporting a Cable-Suspended Load with Unknown Mass. In 19th IFAC World Congress, Cape Town. Cited on page 114.

Desai, J. P., Ostrowski, J. P. and Kumar, V. (2001). Modeling and control of formations of nonholonomic mobile robots. Robotics and Automation, IEEE Transactions on *17*, 905–908. Cited on page 27.

Diebel, J. (2006). Representing Attitude: Euler Angles, Unit Quaternions and Rotation Vectors. Technical report Stanford University Stanford, California 94301-9010. Cited on pages 127, 163 et 164.

Dunbar, W. B. and Murray, R. M. (2006). Distributed receding horizon control for multi-vehicle formation stabilization . Automatica *42*, 549–558. Cited on pages 32 et 33.

d'Andréa-Novel, B., Boustany, F. and Conrad, F. (1992). Control of an overhead crane: Stabilization of flexibilities. In Boundary Control and Boundary Variation, (Zoléesio, J., ed.), vol. 178, chapter Lecture Notes in Control and Information Sciences, pp. 1–26. Springer Berlin Heidelberg. Cited on page 113.

d'Andréa-Novel, B. and Coron, J. M. (2000). Exponential stabilization of an overhead crane with flexible cable via a back-stepping approach. Automatica *36*, 587–593. Cited on page 113.

Eberhart, R. C. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In Proceedings of the sixth international symposium on micro machine and human science vol. 1, pp. 39–43,. Cited on page 22.

Egerstedt, M. and Hu, X. (2001). Formation constrained multi-agent control. Robotics and Automation, IEEE Transactions on *17*, 947–951. Cited on page 28.

Escareño, J., Sanchez, A., Garcia, O. and Lozano, R. (2008). Triple tilting rotor mini-UAV: modeling and embedded control of the attitude. In American Control Conference, 2008 pp. 3476–3481,. Cited on pages 9, 116 et 118.

Fan, P., Wang, X. and Cai, K.-Y. (2010). Design and control of a tri-rotor aircraft. In Control and Automation (ICCA), 2010 8th IEEE International Conference on pp. 1972–1977,. Cited on page 116.

Fax, J. A. and Murray, R. M. (2002). Graph Laplacians and stabilization of vehicle formations. In World Congress vol. 15, pp. 88–88,. Cited on page 27.

Figueroa, R., Faust, A., Cruz, P., Tapia, L. and Fierro, R. (2014). Reinforcement learning for balancing a flying inverted pendulum. In Proc. The 11th World Congress on Intelligent Control and Automation. Cited on pages 8 et 114.

Fliess, M. and Join, C. (2013). Model-free control. International Journal of Control *86*, 2228–2252. Cited on page 112.

Fliess, M., Lévine, J., Martin, P. and Rouchon, P. (1995). Flatness and defect of nonlinear systems: Introductory theory and examples. International Journal of Control *61*, 1327–1361. Cited on pages 10, 130 et 131.

Frihauf, P. and Krstic, M. (2011). Leader-enabled deployment onto planar curves: A PDE-based approach. Automatic Control, IEEE Transactions on *56*, 1791–1806. Cited on pages 4 et 43.

Gazi, V. and Fidan, B. (2007). Coordination and control of multi-agent dynamic systems: Models and approaches. In Swarm Robotics, (Şahin, E., Spears, W. and Winfield, A. T., eds), pp. 71–102. Springer Berlin Heidelberg. Cited on page 18.

Gevrey, M. (1918). Sur la nature analytique des solutions des équations aux dérivées partielles. Premier mémoire. In Annales Scientifiques de l'École Normale Supérieure vol. 35, pp. 129–190,. Cited on pages 42 et 47.

Girard, M. and Amkraut, S. (1990). Eurhythmy: Concept and process. The journal of Visualization and computer animation *1*, 15–17. Cited on pages 2 et 17.

Glass, O. and Guerrero, S. (2007). On the uniform controllability of the Burgers equation. SIAM Journal on Control and Optimization *46*, 1211–1238. Cited on page 46.

Goodarzi, F. A., Lee, D. and Lee, T. (2013). Geometric Stabilization of Quadrotor UAV with a Payload Connected by Flexible Cable. In American Control Conference, 2014. Cited on page 114.

Gorguis, A. (2006). A comparison between Cole–Hopf transformation and the decomposition method for solving Burgers'equations. Applied Mathematics and Computation *173*, 126–136. Cited on page 45.

Gregoire, J., Bonnabel, S. and de La Fortelle, A. (2013). Priority-based coordination of robots. arXiv preprint arXiv:1306.0785 *1*. Cited on page 39.

Guerrero, S. and Imanuvilov, O. Y. (2007). Remarks on global controllability for the Burgers equation with two control forces. In Annales de l'Institut Henri Poincare (C) Non Linear Analysis vol. 24, pp. 897–906,. Cited on page 47.

Harvey, D. (2010). A multimodular algorithm for computing Bernoulli numbers. Mathematics of Computation *79*, 2361–2370. Cited on page 62.

Hehn, M. and D'Andrea, R. (2011). A flying inverted pendulum. In Robotics and Automation (ICRA), 2011 IEEE International Conference on pp. 763–770,. Cited on pages 8 et 114.

Hoffmann, G. M., Huang, H., Waslander, S. L. and Tomlin, C. J. (2011). Precision flight control for a multi-vehicle quadrotor helicopter testbed . Control Engineering Practice *19*, 1023–1036. Cited on page 111.

Holmgren, E. (1908). Sur l'équation de propagation de la chaleur. Ark. Mat. Astr. Fys. *4*, 1–28. Cited on page 53.

Hopf, E. (1950). The partial differential equation $u_t + uu_x = \mu u_{xx}$. Communications on Pure and Applied Mathematics *3*, 201–230. Cited on page 44.

Hua, M.-D., Hamel, T., Morin, P. and Samson, C. (2009). A Control Approach for Thrust-Propelled Underactuated Vehicles and its Application to VTOL Drones. Automatic Control, IEEE Transactions on *54*, 1837–1853. Cited on page 112.

Hua, M.-D., Hamel, T. and Samson, C. (2012). Control of VTOL Vehicles with Thrust-Tilting Augmentation. In IFAC World Congress. Cited on pages 9, 115 et 116.

Jadbabaie, A., Lin, J. and Morse, A. S. (2003). Coordination of groups of mobile autonomous agents using nearest neighbor rules. Automatic Control, IEEE Transactions on *48*, 988–1001. Cited on pages 20 et 21.

Johnson, W. (1994). Helicopter Theory. Dover Books on Aeronautical Engineering Series, Dover Publications. Cited on pages 14, 147 et 174.

Kastelan, D., Konz, M. and Rudolph, J. (2015). Fully Actuated Tricopter with Pilot-Supporting Control. In ACNAAV 15, Sevilla. Cited on pages vii, 10, 117, 124, 128 et 137.

Kendoul, F., Fantoni, I. and Lozano, R. (2005). Modeling and control of a small autonomous aircraft having two tilting rotors. In Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on pp. 8144–8149,. Cited on page 115.

Knüppel, T. and Woittennek, F. (2010). Flatness Based Control Design for a Nonlinear Heavy Chain Model. In Nonlinear Control Systems pp. 701–706,. Cited on page 114.

Konz, M. and Rudolph, J. (2013). Quadrotor tracking control based on a moving frame. In 9th IFAC Symposium on Nonlinear Control Systems, Toulouse vol. 1, pp. 80–85,. Cited on pages 8, 12, 111, 137, 169 et 171.

Krstić, M., Magnis, L. and Vazquez, R. (2008). Nonlinear stabilization of shock-like unstable equilibria in the viscous Burgers PDE. Automatic Control, IEEE Transactions on *53*, 1678–1683. Cited on pages 4 et 43.

Krstić, M., Magnis, L. and Vazquez, R. (2009). Nonlinear control of the viscous burgers equation: Trajectory generation, tracking, and observer design. Journal of Dynamic Systems, Measurement, and Control *131*, 021012. Cited on page 45.

Krstić, M. and Smyshlyaev, A. (2008). Boundary Control of PDEs: A Course on Backstepping Designs. Advances in Design and Control, Society for Industrial and Applied Mathematic. Cited on pages 4 et 43.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. Naval research logistics quarterly *2*, 83–97. Cited on page 33.

Laroche, B., Martin, P. and Rouchon, P. (2000). Motion planning for the heat equation. International journal of robust and nonlinear control *10*, 629–643. Cited on pages vi, 4, 5, 41, 42, 43, 47, 48, 55, 76 et 77.

LaValle, S. M. and Hutchinson, S. A. (1998). Optimal motion planning for multiple robots having independent goals. Robotics and Automation, IEEE Transactions on *14*, 912–925. Cited on page 39.

Lee, T., Leoky, M. and McClamroch, N. H. (2010). Geometric tracking control of a quadrotor UAV on SE (3). In Decision and Control (CDC), 2010 49th IEEE Conference on pp. 5420–5425,. Cited on page 111.

Lee, T., Sreenath, K. and Kumar, V. (2013). Geometric control of cooperating multiple quadrotor UAVs with a suspended payload. In Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on pp. 5510–5515,. Cited on pages 8, 38 et 114.

Lehmer, D. H. (1940). On the Maxima and Minima of Bernoulli Polynomials. The American Mathematical Monthly *47*, 533–538. Cited on page 68.

Lenoir, Y., Martin, P. and Rouchon, P. (1998). 2kπ, the juggling robot. In Decision and Control, 1998. Proceedings of the 37th IEEE Conference on vol. 2, pp. 1995–2000,. Cited on page 114.

Leonard, N. E. and Fiorelli, E. (2001). Virtual leaders, artificial potentials and coordinated control of groups. In Decision and Control, 2001. Proceedings of the 40th IEEE Conference on vol. 3, pp. 2968–2973,. Cited on pages 24 et 28.

Logan, J. D. (2008). An introduction to nonlinear partial differential equations, vol. 89,. John Wiley & Sons. Cited on page 85.

Lozano, R., Fantoni, I. and Block, D. J. (2000). Stabilization of the inverted pendulum around its homoclinic orbit . Systems & Control Letters *40*, 197–204. Cited on page 114.

Lutz, D. A., Miyake, M. and Schäfke, R. (1999). On the Borel summability of divergent solutions of the heat equation. Nagoya Mathematical Journal *154*, 1–29. Cited on page 73.

Malgrange, B. and Ramis, J.-P. (1992). Fonctions multisommables. Ann. Inst. Fourier (Grenoble) *42*, 353–368. Cited on page 73.

Martin, P., Rosier, L. and Rouchon, P. (2014). Null Controllability of the 1D Heat Equation Using Flatness. In 19th IFAC World Congress, Cape Town. Cited on page 49.

Martin, P. and Salaun, E. (2010). The true role of accelerometer feedback in quadrotor control. In Robotics and Automation (ICRA), 2010 IEEE International Conference on pp. 1623–1629,. Cited on pages 107 et 111.

Maza, I., Kondak, K., Bernard, M. and Ollero, A. (2010). Multi-UAV cooperation and control for load transportation and deployment. Journal of Intelligent and Robotic Systems *57*, 417–449. Cited on pages 36, 39 et 160.

Mellinger, D. and Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. In Robotics and Automation (ICRA), 2011 IEEE International Conference on pp. 2520–2525,. Cited on pages 22, 33, 38, 107 et 111.

Mellinger, D., Michael, N. and Kumar, V. (2012). Trajectory generation and control for precise aggressive maneuvers with quadrotors. The International Journal of Robotics Research *31*, 664–674. Cited on pages 107 et 110.

Mellinger, D., Shomin, M., Michael, N. and Kumar, V. (2013). Cooperative grasping and transport using multiple quadrotors. In Distributed autonomous robotic systems pp. 545–558. Springer. Cited on pages 8, 36 et 113.

Menabrea, L. F. and Lovelace, A. K. C. o. (1843). Sketch of the analytical engine invented by Charles Babbage, Esq. Richard and John E. Taylor. Cited on page 61.

Mesbahi, M. (2005). On State-dependent dynamic graphs and their controllability properties. Automatic Control, IEEE Transactions on *50*, 387–392. Cited on page 28.

Meurer, T. (2005). Feedforward and feedback tracking control of diffusion-convection-reaction systems using summability methods. PhD thesis, Universitätsbibliothek der Universität Stuttgart. Cited on page 73.

Meurer, T. and Krstić, M. (2011). Finite-time multi-agent deployment: A nonlinear PDE motion planning approach. Automatica *47*, 2534–2542. Cited on pages 4, 34, 43, 45, 48 et 87.

Michael, N., Fink, J. and Kumar, V. (2011). Cooperative manipulation and transportation with aerial robots. Autonomous Robots *30*, 73–86. Cited on pages 36 et 38.

Mogilner, A. and Edelstein-Keshet, L. (1999). A non-local model for a swarm. Journal of Mathematical Biology *38*, 534–570. Cited on pages 29 et 30.

Mohamed, M. K. and Lanzon, A. (2012). Design and control of novel tri-rotor UAV. In Control (CONTROL), 2012 UKACC International Conference on pp. 304–309,. Cited on pages 9, 116 et 118.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. Journal of the Society for Industrial & Applied Mathematics *5*, 32–38. Cited on page 33.

Murray, R. M. (1996). Trajectory generation for a towed cable system using differential flatness. In IFAC world congress pp. 395–400,. Cited on pages 36 et 114.

Murray, R. M., Rathinam, M. and Sluis, W. (1995). Differential flatness of mechanical control systems: A catalog of prototype systems. In ASME International Mechanical Engineering Congress and Exposition. Cited on pages 130 et 131.

Nörlund, N. E. (1922). Mémoire sur les polynômes de Bernoulli. Acta Mathematica *43*, 121–196. Cited on page 73.

Ogren, P., Egerstedt, M. and Hu, X. (2001). A control Lyapunov function approach to multi-agent coordination. In Decision and Control, 2001. Proceedings of the 40th IEEE Conference on vol. 2, pp. 1150–1155,. Cited on page 36.

Olfati-Saber, R. (2006). Flocking for multi-agent dynamic systems: Algorithms and theory. Automatic Control, IEEE Transactions on *51*, 401–420. Cited on page 26.

Olfati-Saber, R. and Murray, R. M. (2002). Distributed cooperative control of multiple vehicle formations using structural potential functions. In IFAC World Congress pp. 346–352,. Cited on page 28.

Olfati-Saber, R. and Murray, R. M. (2003). Flocking with obstacle avoidance: Cooperation with limited communication in mobile networks. In Decision and Control, 2003. Proceedings. 42nd IEEE Conference on vol. 2, pp. 2022–2028,. Cited on page 26.

Olfati-Saber, R. and Murray, R. M. (2004). Consensus problems in networks of agents with switching topology and time-delays. Automatic Control, IEEE Transactions on *49*, 1520–1533. Cited on pages 26 et 28.

Omari, S., Hua, M.-D., Ducard, G. and Hamel, T. (2013). Nonlinear control of VTOL UAVs incorporating flapping dynamics. In Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on pp. 2419–2425,. Cited on page 107.

Ostrowski, A. M. (1960). On the zeros of Bernoulli polynomials of even order. L'Enseignement Mathématique *6*. Cited on page 73.

Oung, R., Bourgault, F., Donovan, M. and D'Andrea, R. (2010). The Distributed Flight Array. In Robotics and Automation (ICRA), 2010 IEEE International Conference on pp. 601–607,. Cited on page 36.

Palunko, I., Cruz, P. and Fierro, R. (2012). Agile load transportation: Safe and efficient load manipulation with aerial robots. Robotics & Automation Magazine, IEEE *19*, 69–79. Cited on pages 8 et 113.

Palunko, I. and Fierro, R. (2011). Adaptive control of a quadrotor with dynamic changes in the center of gravity. In Proceedings 18th IFAC World Congress vol. 18, pp. 2626–2631,. Cited on pages 8 et 113.

Panagou, D., Turpin, M. and Kumar, V. (2014). Decentralized Goal Assignment and Trajectory Generation in Multi-Robot Networks: A Multiple Lyapunov Functions Approach. In Proc. of the 2014 IEEE Int. Conf. on Robotics and Automation, Hong Kong, China. Cited on page 34.

Petit, N. and Rouchon, P. (2001). Flatness of heavy chain systems. SIAM Journal on Control and Optimization *40*, 475–495. Cited on page 113.

Pillu, H. (2012). Conception, commande et contrôle d'un tricoptère. Master's thesis École Centrale de Lille. Cited on page 117.

Piovan, G. and Bullo, F. (2012). On Coordinate-Free Rotation Decomposition: Euler Angles About Arbitrary Axes. Robotics, IEEE Transactions on *28*, 728–733. Cited on page 169.

Pounds, P., Mahony, R., Hynes, P. and Roberts, J. (2002). Design of a four-rotor aerial robot. In Australasian Conference on Robotics and Automation pp. 145–150,. Cited on pages 8, 107 et 108.

Pounds, P. E., Bersak, D. R. and Dollar, A. M. (2012). Stability of small-scale UAV helicopters and quadrotors with added payload mass under PID control. Autonomous Robots *33*, 129–142. Cited on pages 8 et 113.

Powers, C., Mellinger, D., Kushleyev, A., Kothmann, B. and Kumar, V. (2012). Influence of Aerodynamics and Proximity Effects in Quadrotor Flight. In Proceedings of the International Symposium on Experimental Robotics. Cited on page 147.

Prodan, I., Olaru, S., Stoica, C. and Niculescu, S. I. (2011). Predictive control for tight group formation of multi-agent systems. In Proceedings of the 18th IFAC World Congress, Milano, Italy pp. 138–143,. Cited on pages 33 et 34.

Pucci, D., Hamel, T., Morin, P. and Samson, C. (2011). Nonlinear control of PVTOL vehicles subjected to drag and lift. In Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on pp. 6177–6183,. Cited on page 97.

Pugh, J. and Martinoli, A. (2007). Inspiring and modeling multi-robot search with particle swarm optimization. In Swarm Intelligence Symposium, 2007. SIS 2007. IEEE pp. 332–339,. Cited on pages 23 et 24.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In ACM SIGGRAPH Computer Graphics vol. 21, pp. 25–34,. Cited on pages 2 et 17.

Rimon, E. and Koditschek, D. E. (1992). Exact robot navigation using artificial potential functions. Robotics and Automation, IEEE Transactions on *8*, 501–518. Cited on pages 24 et 26.

Rodino, L. (1993). Linear partial differential operators in Gevrey spaces. World Scientific Publishing Company Incorporated. Cited on pages 42 et 54.

Rouchon, P., Fliess, M., Lévine, J. and Martin, P. (1993). Flatness, motion planning and trailer systems. In Decision and Control, 1993., Proceedings of the 32nd IEEE Conference on pp. 2700–2705,. Cited on page 130.

Rudin, W. (1976). Principles of mathematical analysis. Third Edition edition, McGraw-Hill New York. Cited on page 67.

Rudolph, J., Winkler, J. and Woittennek, F. (2003). Flatness Based Control of Distributed Parameter Systems. Shaker Verlag edition. Cited on pages 43, 48 et 49.

Ryll, M., Bulthoff, H. H. and Giordano, P. R. (2012). Modeling and control of a quadrotor UAV with tilting propellers. In Robotics and Automation (ICRA), 2012 IEEE International Conference on pp. 4606–4613,. Cited on pages 9, 115 et 140.

Salazar-Cruz, S., Kendoul, F., Lozano, R. and Fantoni, I. (2008). Real-time stabilization of a small three-rotor aircraft. Aerospace and Electronic Systems, IEEE Transactions on *44*, 783–794. Cited on page 116.

Sanchez, A., Escareno, J., Garcia, O. and Lozano, R. (2008). Autonomous hovering of a noncyclic tiltrotor UAV: Modeling, control and implementation. In Proc. of the 17th IFAC Wold Congress pp. 803–808,. Cited on page 116.

Servais, E., d'Andréa-Novel, B. and Mounier, H. (2015a). Ground control of a hybrid tricopter. In Unmanned Aircraft Systems (ICUAS), 2015 International Conference on pp. 945–950,. Cited on page 128.

Servais, E., d'Andréa-Novel, B. and Mounier, H. (August 24-27, 2015b). Trajectory tracking of Trirotor UAV with Pendulum Load. In MMAR'2015, Amber Baltic Hotel, Międzyzdroje, Poland. Cited on page 128.

Shi, Y. and Eberhart, R. (1998). A modified particle swarm optimizer. In Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on pp. 69–73,. Cited on page 22.

Smith, R. (1980). Communication and Control in Problem Solver. IEEE Transactions on computers *29*, 12. Cited on page 39.

Sreenath, K. and Kumar, V. (2013). Dynamics, Control and Planning for Cooperative Manipulation of Payloads Suspended by Cables from Multiple Quadrotor Robots. In Robotics: Science and Systems vol. 1,. Cited on pages 8, 37, 38 et 113.

Sreenath, K., Michael, N. and Kumar, V. (2013). Trajectory generation and control of a quadrotor with a cable-suspended load - A differentially-flat hybrid system. In Robotics and Automation (ICRA), 2013 IEEE International Conference on pp. 4888–4895,. Cited on pages 8, 113, 161, 168 et 169.

Su, H., Wang, X. and Lin, Z. (2009). Flocking of Multi-Agents With a Virtual Leader. Automatic Control, IEEE Transactions on *54*, 293–307. Cited on page 27.

Sultan, C., Seereram, S. and Mehra, R. K. (2007). Deep space formation flying spacecraft path planning. The International Journal of Robotics Research *26*, 405–430. Cited on page 32.

Tanaka, K., Suzuki, R., EMaru, T., Higashi, Y. and Wang, H. O. (2007). Development of a cyclogyro-based flying robot with variable attack angle mechanisms. Mechatronics, IEEE/ASME Transactions on *12*, 565–570. Cited on page 103.

Tayebi, A. and McGilvray, S. (2006). Attitude stabilization of a VTOL quadrotor aircraft. Control Systems Technology, IEEE Transactions on *14*, 562–571. Cited on page 107.

Thorel, S. (2014). Design and construction of an autonomous hybrid ground/air drone for indoor applications. PhD thesis, Ecole Nationale Supérieure des Mines de Paris. Cited on pages 10 et 138.

Thorel, S. and d'Andréa-Novel, B. (2014). Hybrid Terrestrial and Aerial Quadrotor Control. In 19th IFAC World Congress, Cape Town vol. 19, pp. 9834–9839,. Cited on pages 9, 107, 115 et 139.

Tillett, J., Rao, T., Sahin, F. and Rao, R. (2005). Darwinian particle swarm optimization. In Proc. Indian Int. Conf. Artif. Intell. pp. 1474–1487,. Cited on page 24.

Toner, J. and Tu, Y. (1998). Flocks, herds, and schools: A quantitative theory of flocking. Physical review E *58*, 4828. Cited on page 31.

Toner, J., Tu, Y. and Ramaswamy, S. (2005). Hydrodynamics and phases of flocks . Annals of Physics *318*, 170–244. Cited on page 31.

Topaz, C. M. and Bertozzi, A. L. (2004). Swarming patterns in a two-dimensional kinematic model for biological groups. SIAM Journal on Applied Mathematics *65*, 152–174. Cited on page 30.

Turduev, M., Cabrita, G., Kırtay, M., Gazi, V. and Marques, L. (2014). Experimental studies on chemical concentration map building by a multi-robot system using bio-inspired algorithms. Autonomous agents and multi-agent systems *28*, 72–100. Cited on page 24.

Turpin, M., Michael, N. and Kumar, V. (2012). Trajectory design and control for aggressive formation flight with quadrotors. Autonomous Robots *33*, 143–156. Cited on page 21.

Turpin, M., Mohta, K., Michael, N. and Kumar, V. (2013). Goal Assignment and Trajectory Planning for Large Teams of Aerial Robots. In Robotics: Science and Systems. Cited on page 33.

Vicsek, T., Czirók, A., Ben-Jacob, E., Cohen, I. and Shochet, O. (1995). Novel Type of Phase Transition in a System of Self-Driven Particles. Phys. Rev. Lett. *75*, 1226–1229. Cited on pages 20 et 21.

Wang, J., Mounier, H., Cela, A. and Niculescu, S.-I. (2011). Event driven intelligent PID controllers with applications to motion control. In 18th IFAC World Congress, Milan. Cited on pages 8 et 112.

Waslander, S. L., Hoffmann, G. M., Jang, J. S. and Tomlin, C. J. (2005). Multi-agent quadrotor testbed control design: integral sliding mode vs. reinforcement learning. In Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on pp. 3712–3717,. Cited on page 112.

Yamagami, T., Saito, Y., Matsuzaka, Y., Namiki, M., Toriumi, M., Yokota, R., Hirosawa, H. and Matsushima, K. (2004). Development of the highest altitude balloon . Advances in Space Research *33*, 1653–1659. Cited on page 96.

Yamanaka, T. (1989). A new higher order chain rule and Gevrey class. Annals of Global Analysis and Geometry *7*, 179–203. Cited on page 80.

Yoo, D.-W., Oh, H.-D., Won, D.-Y. and Tahk, M.-J. (2010). Dynamic Modeling and Stabilization Techniques for Tri-Rotor Unmanned Aerial Vehicles. International Journal of Aeronautical and Space Science *11*, 167–174. Cited on page 116.