Flatness-based constrained control and model-free control applications to quadrotors and cloud computing

Maria Bekcheva

université PARIS-SACLAY

NNT : 2019SACLS218

Thèse de doctorat

UNIVERSITÉ PARIS SUD

Comprendre le monde, construire l'avenir

Flatness-based Constrained Control and Model-Free Control Applications to Quadrotors and Cloud Computing

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'Université Paris-Sud

École doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)

Spécialité de doctorat: Automatique

Thèse présentée et soutenue à Gif-sur-Yvette, le 11/07/2019, par

Maria BEKCHEVA

Composition du Jury :

Silviu Iulian NICULESCU, Directeur de Recherche CNRS Université Paris-Saclay – L2S	Président
Emmanuel DELALEAU, Professeur des Universités École nationale d'ingénieurs de Brest – Dépt. de Mécatronique	Rapporteur
Didier THEILLIOL, Professeur des Universités Université de Lorraine – CRAN	Rapporteur
Mireille BAYART, Professeur des Universités Université de Lille - CRISTAL	Examinatrice
Michel FLIESS, Directeur de Recherche CNRS Emérite Ecole Polytechnique -LIX	Examinateur
Cédric JOIN, Professeur des Universités Université de Lorraine – CRAN	Examinateur
Hugues MOUNIER, Professeur des Universités Université Paris-Sud (Université Paris-Saclay) – L2S	Directeur de thèse
Luca GRECO, Maître de Conférences Université Paris-Sud (Université Paris-Saclay) – L2S	Co-Directeur de thèse

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

Flatness-based Constrained Control and Model-Free Control Applications to Quadrotors and Cloud Computing

Maria Bekcheva

Université Paris-Saclay

Université Paris-Sud

A thesis submitted for the degree of

PhD

July 2019

I dedicate this thesis to my dear parents and Alireza.

Remerciements

Mes premiers remerciements sont adressés à mon directeur de thèse Hugues Mounier. Je lui suis très reconnaissante pour ces années, durant lesquelles il m'a guidée sur plusieurs sujets, m'a encouragée dans les moments de doute et m'a conseillée avec patience et pédagogie. Je remercie Hugues d'être un directeur tolérable avec un esprit moderne, libre et innovant. Ce fut un privilège de travailler avec lui. Je remercie également mon co-directeur Luca Greco pour ses précieux conseils, son aide et ses encouragements durant ces années de collaboration.

Je voudrais remercier Michel Fliess et Cédric Join. Cédric m'a guidée et encouragée durant ces deux dernières années. J'ai beaucoup appris à son contact et je tiens à le remercier très chaleureusement pour ses précieux conseils et bienveillance. Je remercie très chaleureusement Michel Fliess pour sa bonne humeur, et ses paroles toujours encourageantes. Ma recherche a été profondément influencée par ses travaux et on peut déceler cette influence tout au long de ce mémoire. Je suis très heureuse d'avoir pu bénéficier de ses lumières. Il m'a fait partager avec enthousiasme son ample culture scientifique et son expérience.

Je suis honorée d'avoir eu comme rapporteurs Emmanuel Delaleau et Didier Theilliol. Un grand merci pour leur relecture attentive de mon manuscrit, leurs suggestions et corrections avisées.

Je remercie Silviu Niculescu de m'avoir fait l'honneur de présider mon jury de thèse, et pour ses précieux conseils et son aide tout au long de ma thèse.

Je tiens aussi à remercier Mireille Bayart, qui a accepté d'examiner mon travail.

J'aimerai remercier toute l'équipe de L2S et l'école doctorale STIC pour leurs aides et sympathies et tout particulièrement Gilles Duc pour ses conseils administratives et pédagogiques éclairés.

Je remercie mes collègues docteurs et doctorants de L2S. Ce fut un grand plaisir de vous connaître, de partager nos recherches et de vous retrouver autour des

différents repas et pauses-café.

Je remercie infiniment mes parents et mon frère Branko, pour leur soutien et amour.

Enfin, je voudrais exprimer toute ma gratitude à Alireza qui, jour après jour, partage mes joies et mes peines, me supporte, m'encourage, m'inspire, me conseille, m'aime.

Résumé

Motivation

L'Automatique, avec son sous-domaine mathématique Théorie du Contrôle, étudie les propriétés et la commande des systèmes dynamiques en ingénierie. Chaque système contrôlé est composé de commandes (on dit aussi entrée ou contrôle) u, d'états x et de sorties y. L'objectif de la théorie du contrôle est de concevoir une commande de telle sorte que les états et les sorties atteignent un objectif défini. Les commandes agissent sur les états qui représentent habituellement la dynamique interne du système. Alors que les sorties représentant les composants mesurables, sont liés aux états. Lorsque la commande dépend uniquement de la mesure de sortie et que seule l'erreur entre la référence et la sortie mesurée est utilisée dans la conception, la commande est appelée *une rétro-action* (on dit aussi retour d'état ou feedback). La commande de rétroaction ou le cadre avec *un degré-de-liberté* échoue souvent pour les systèmes non linéaires, où il n'existe pas de solution globale asymptotiquement stable qui satisfait les contraintes du *système* (comme indiqué dans [95]).

Dans cette thèse, nous travaillons dans le cadre de la *platitude différentielle* ou le cadre avec *deux degrés de liberté*, qui est composé de deux parties : un générateur (ou une planification) des trajectoires et une commande de rétroaction pour réduire les erreurs dues aux perturbations et incertitudes. Les travaux de recherche de Fliess et de ses collègues [54–56] sur les systèmes différentiellement plats et leurs propriétés ont permis d'approfondir la compréhension du suivi de trajectoire par le paramétrage des trajectoires du système. Pour un système plat [55], tous les états et les entrées peuvent être paramétrés via une *sortie plate* et la planification de trajectoire est obtenue sans résoudre les équations différentielles. De plus, dans le cadre des systèmes plats, le comportement de chaque variable du système peut être facilement analysé. De ce point de vue, la conception de la commande peut donc être décomposée en deux étapes :

- Conception des trajectoires de référence pour les sorties plates. Des calculs hors ligne des commandes en boucle ouverte *(feedforward)*.

- Calcul en ligne des commandes en boucle fermée afin de stabiliser le système autour des trajectoires de référence *(stabilisation ou feedback)*.

Cette conception en deux étapes convient mieux qu'une commande classique de rétro-action (schéma de stabilisation) c'est-à-dire un cadre avec *un degré de liberté*. La première étape consiste à obtenir une solution du premier ordre au problème de suivi de trajectoire en tenant compte du modèle du système (comme dans un schéma classique de stabilisation pure). La deuxième étape est une étape de raffinement, où l'erreur entre les valeurs mesurées réelles et les références suivies sera beaucoup plus petite que dans le cas de la stabilisation pure. Le cadre avec *deux degrés de liberté* est une solution plus attrayante compte tenu de l'effort de calcul, de la complexité et de la stabilité. En outre, de nombreuses classes de systèmes non linéaires sont différentiellement plates [55], ce qui les rend plus faciles à analyser.
Cependant, lorsque les systèmes plats sont confrontés au problème de la *commande avec contraintes*, il y a encore de grandes difficultés qui subsistent.

Contraintes

Les contraintes sont présentes dans tous les systèmes contrôlés et peuvent avoir des effets néfastes sur la performance du système si elles ne sont pas prises en compte dans la conception de la commande. En général, les contraintes découlent des relations constitutives et physiques existantes entre les composants du système et son environnement de travail. Le plus souvent, les contraintes sont identifiées par des *contraintes d'entrée, contraintes d'état et/ou contraintes de sortie* :

- *Contraintes d'entrée*
  Les contraintes d'entrée dépendent des contraintes de l'actionneur. L'actionneur (par exemple, moteur, vanne, interrupteur) est un composant mécanique qui reçoit un signal de commande et une source d'énergie (par exemple, électricité, liquide, air), et convertit l'énergie du signal en mouvement ou force. Les actionneurs sont soumis à des saturations d'amplitude et de vitesse. Ils ne sont capables de fournir qu'une quantité limitée de mouvement ou de force. L'actionneur peut prendre une valeur entre une limite inférieure et supérieure $u_{\min} < u < u_{\max}$. Dans certaines applications, les saturations sont évitées en choisissant des actionneurs plus puissants, mais cela peut ne pas être une solution permanente. Un exemple courant d'actionneur est le moteur à courant continu qui est présent dans chaque système de véhicule aérien sans pilote (UAV) pour fournir un mouvement de rotation. Le moteur à courant continu est limité en tension et en courant. D'autres exemples de contraintes d'entrée sont : le couple maximal disponible pour les dispositifs mécatroniques ou la puissance de refroidissement/chauffage limitée pour les réacteurs chimiques.

- *Contraintes d'état*
  Des contraintes d'état peuvent provenir de la structure du système. On peut considérer les exemples suivants : dans les robots mobiles non-holonomes (par exemple, un chariot avec deux roues motrices avant et deux roues arrière qui ne peuvent pas se déplacer latéralement), les contraintes sur les vitesses longitudinales et angulaires; dans les systèmes sous-actionnés (par exemple, le quadrirotor), les contraintes sur les angles; dans les processus chimiques, les contraintes sur les variables du processus.

- *Contraintes de sortie*
  Les contraintes de sortie sont généralement définies par l'environnement de travail. Dans la planification des mouvements, les robots mobiles se déplacent d'un point à l'autre tout en évitant les obstacles de l'environnement. Dans certaines applications, pour des raisons de sécurité ou de confort, des contraintes de sortie sont imposées par la demande des utilisateurs. Par exemple, limitations sur les vitesses longitudinales, les vitesses angulaires, les températures.

La trajectoire de référence doit tenir compte des contraintes du système. Une telle trajectoire qui se trouve à l'intérieur du domaine d'état admissible et qui respecte les contraintes s'appelle une *trajectoire réalisable*. Si les contraintes du système ne sont pas prises en compte, la trajectoire peut être irréalisable et la mission définie peut ne pas être accomplie. L'échec d'une mission se produit par exemple lorsque les actionneurs atteignent leurs limites et ne peuvent pas fournir les entrées d'actionnement souhaitées par le contrôleur.

Il est donc susceptible d'avoir *un générateur de trajectoires* qui définit un ensemble de trajectoires de référence réalisables c-à-d feed-forwarding trajectoires qui répondront aux contraintes définies. Une approche systématique capable de satisfaire les contraintes au préalable et d'intégrer la satisfaction des contraintes directement dans la formulation de la commande est un avantage clé. Ce fait nous motive à intégrer systématiquement les contraintes dans la conception de trajectoire.

Perturbations

Pour un système physique, à part la satisfaction des contraintes, le traitement des perturbations est essentiel à son bon fonctionnement. Par exemple, un quadrirotor devrait suivre sa trajectoire malgré les perturbations dues au vent ou la variation de masse (lorsqu'il transporte une charge). Cependant, lors de la compensation des perturbations, les trajectoires de référence dans le temps doivent induire un mouvement suffisamment lent, c'est-à-dire diminuer sa vitesse ou changer sa trajectoire de référence initiale afin d'éviter les saturations des moteurs du quadrirotor.

Par conséquent, un problème de contrôle important est de savoir comment concevoir la trajectoire de référence dans la *présence de perturbations* de manière à ce que les contraintes soient respectées. Dans la pratique de l'automatique, pour annuler les perturbations, un *contrôleur en boucle fermée* ou un *contrôleur de rétroaction* est utilisé pour surmonter les limites du contrôleur en boucle ouverte. Les perturbations du système peuvent être divisées en deux groupes :

- *perturbations internes*, par exemple, le modèle du système est inconnu ou en partie inconnu, les incertitudes des paramètres du modèle, et
- *pertubations externes*, par exemple, le bruit des capteurs, les rafales de vent ou toute autre perturbation de l'environnement.

Pour traiter les perturbations du système, dans cette thèse, nous utilisons la Commande sans modèle. La *Commande sans modèle* (CSM), présenté par Michel Fliess et Cédric Join [47, 48], a déjà prouvé sa puissance à travers un large éventail d'applications réussies[1, 49, 93, 96, 105] et même, avec des résultats expérimentaux [59, 60] où le modèle du système et ses perturbations sont inconnus. Les tentatives réussies pour une commande sans modèle non-linéaire et pour une commande sans modèle pour les systèmes à retard sont présentées dans [26, 106] et dans [37] respectivement. La première preuve détaillée de la stabilité de la commande sans modèle qui fournit des informations sur les paramètres de contrôle, a été donnée dans [35].

Organisation de la thèse et contributions

Cette thèse est consacrée à deux problèmes : le premier est la commande des systèmes différentiellement plats avec contraintes, et le second est l'application de la Commande sans modèle en utilisant sa robustesse par rapport aux dynamiques et pertubations non modélisées du système et de son environnement (voir Figure 1.1).

Le but ultime de ce projet de thèse est de montrer que les contraintes peuvent être satisfaites par les entrées c-à-d les feedforwarding nominales, et que les perturbations internes et externes peuvent être gérées par la Commande sans modèle. Puisque l'objectif derrière ces deux problèmes de contrôle (les contraintes et les perturbations), est essentiel pour presque tous les systèmes de contrôle, il est intéressant de noter à quel point peu de recherche est faite pour traiter simultanément ces deux problèmes. La difficulté vient de la nécessité de replanifier rapidement la partie feedforwarding lorsqu'une perturbation importante se produit, si la partie feedback augmente. Lorsque de telles perturbations se produisent, une replanification rapide de la trajectoire est essentielle pour éviter la saturation de l'actionneur. Dans [127] est présenté une commande composée de deux parties : la première partie est construite selon la platitude differentielle et la seconde partie suit la commande sans modèle pour un cas particulier où la perturbation et sa dérivée sont supposées bornées.

Mais que se passe-t-il lorsqu'une perturbation importante se produit soudainement ? Comment replanifier rapidement une nouvelle trajectoire de référence ?

Pour ce faire, nous avons étudié la planification de trajectoire sous contraintes présentée dans la *première partie* de la thèse, où nous définissons un ensemble de trajectoires réalisables.

- Dans le chapitre 2, nous nous concentrons sur la génération de trajectoires réalisables plutôt que sur la stabilisation ou le feedback. Notre but est d'écrire les *contraintes d'entrée et d'état* en termes de la sortie plate et ses dérivés. Nous proposons d'utiliser les courbes de Bézier comme trajectoires de référence pour la sortie plate en raison de leurs propriétés utiles (une des clés dans la conception de notre contrôle en boucle ouverte sous contraintes). Nous montrons comment les entrées exprimées par les sorties plates, représentées chacune par une courbe de Bézier, peuvent être exprimées sous la forme d'une combinaison de courbes de Bézier. On obtient ainsi des équations explicites pour les points de contrôle d'entrée et d'état en fonction des points de contrôle des sorties plates. Nous recherchons ensuite les régions réalisables des points de contrôle de Bézier, c'est-à-dire un ensemble de trajectoires de référence réalisables. Il existe un certain nombre d'approches connexes [43, 62, 63, 79, 80, 120, 128, 129, 136] qui reposent toutes sur des procédures d'optimisation pour obtenir une trajectoire de référence réalisable. De plus, les systèmes différentiellement plats avec contraintes sont généralement attaqués en utilisant la commande optimale (voir, par exemple, [38, 42, 108, 111, 126]).

- Dans le chapitre 3, nous étudions les systèmes avec des contraintes en présence de retards sur l'état ou l'entrée. Une grande partie de la littérature d'automatique est consacrée à l'étude des systèmes linéaires et non linéaires avec des retards [119], mais peu de résultats sont capables de traiter les contraintes. Les contraintes sur l'entrée ou les variables d'état d'un système à retard sont généralement traitées avec des méthodes numériques impliquant le calcul d'intervalles invariants de trajectoire [103] et d'ensembles invariants positifs [32, 70], ou plus complexes, mais similaires dans leur esprit, les approches de la commande prédictive (voir par exemple [87, 107]). Dans le cadre des systèmes à retards, une approche systématique capable d'intégrer directement dans la formulation de la commande, la satisfaction des contraintes sur les entrées/états reste manquante.

La *seconde partie* de la thèse présente la Commande sans modèle (CSM) [48] qui estime et annule les perturbations et la dynamique inconnue du système en se basant sur une modélisation ultra-locale. L'apport de la deuxième partie de la thèse

réside dans deux applications de la Commande sans modèle à deux systèmes de nature différente :

- Dans le chapitre 4, nous proposons une commande qui évite les procédures de modélisation/identification du système de quadrirotor tout en restant robuste par rapport aux perturbations endogènes (la performance de contrôle est indépendante de tout changement de masse, inertie, effets gyroscopiques ou aérodynamiques) et exogènes (vent, bruit des mesures). Pour atteindre notre objectif, en se basant sur la structure en cascade du quadrirotor, nous divisons le système en sous-systèmes de position et d'attitude, chacun contrôlé par une *CSM* indépendante de la dynamique du système. Ensuite, nous donnons des résultats probants sur la stabilité pratique de la commande proposée. Nous validons notre approche de contrôle dans trois scénarios réalistes : en présence d'un bruit de mesure inconnu, avec des perturbations du vent variant dans le temps et des variations de masse inconnues.

- Dans le chapitre 5, nous utilisons la CSM pour contrôler l'élasticité horizontale d'un système Cloud Computing. Comparé aux algorithmes commerciaux de "mise à l'échelle automatique", notre approche facilement implémentable se comporte mieux, même avec de fortes fluctuations de charge de travail. Ceci est confirmé par des expérimentations sur le cloud public Amazon Web Services (AWS).

- Enfin, au chapitre 6, nous résumons les travaux de recherche de la thèse et nous proposons, dans une perspective, une commande de type « boîte grise » à l'aide des deux études présentées ci-dessus.

Contents

1 Introduction

1.1 Motivation

Control System Engineering, together with its mathematical sub-field Control Theory, studies the properties of dynamical systems in engineering. Each control system is composed of inputs u, states x and outputs y. Control theory's aim is to design a control input such that the states/outputs reach a defined goal. The inputs act on the states which usually represent the internal dynamics of the system. While the outputs representing the measurable components, are related to the states. When the control input solely depends on the output measure, and only the error between the reference goal and the measured output is used in the design, the control is called *a feedback control*. The feedback controller or the *one-degree-of-freedom* framework often fails for nonlinear systems, where a global asymptotically stable solution that satisfies the *system constraints* does not exist (as stated in [95]).

In this thesis, we work in the framework of *differential flatness* or the *two-degree-of-freedom* framework, which is a combination of a trajectory generator and a feedback controller. Fliess and his coworkers research work [54–56] on differentially flat systems and their properties led to a deeper understanding of trajectory tracking through the system trajectories parametrization. For a differentially flat system [55], all the states and the inputs can be parametrized through a so-called *flat output* and the trajectory planning can be obtained immediately without solving differential equations. Moreover, with the flatness property, the behaviour of each system variable can be easily analyzed. From this perspective, the control design can thus be decomposed in two steps:

- Design of flat outputs reference trajectory; off-line computation of the open-loop controls *(feedforward part)*.
- On-line computation of the complementary closed-loop controls in order to stabilize the system around the reference trajectories *(feedback part)*.

This two-step design is better suited than a classical feedback controller (stabilization scheme) *i.e. one-degree of-freedom* framework. The first step obtains a first-order solution to the tracking problem while following the model instead of forcing it (like in a usual pure stabilization scheme). The second step is a refinement one, where the error between the actual measured values and the tracked references will be much smaller than in the pure stabilization case. The *two-degree-of-freedom* framework is a more attractive solution considering the computational effort, tuning complexity and stability. In addition, many classes of nonlinear systems are differentially flat [55], which makes them easier to analyse.
However, when the flat systems are faced with the problem of *constrained control*, there are still great difficulties that remain.

1.1.1 Constraints

Constraints are present in all control systems and may lead in damaging effects on the system performance unless they are accounted in the control design. In general, constraints arise from the constitutive and physical relationships existing between the components of the control system, and its working environment. Most often constraints are identified as *input, state and/or output constraints*:

- *Input constraints:*
  The inputs constraints depend on the actuator constraints. Actuator (for *e.g.* , motors, valves, switches) is a mechanical component of a system that receives a control signal and a source of energy (*e.g.* , electricity, liquid, air), and converts the signal energy into motion or force. Actuators are subject to magnitude and rate saturations. They are able to deliver only a limited amount of motion or force. The actuator can take a value between a lower and upper limit $u_{\min} < u < u_{\max}$. In some applications, the saturations are avoided by choosing more powerful actuators but this may not be a permanent solution. A common example of actuator is the DC motor which is present in every unmanned aerial vehicle (UAV) system to deliver a rotational motion. The DC motor is constrained in input voltage and current (which without load is equivalent to a velocity or acceleration). Other examples for input constraints are: maximum available torque for mechatronic devices or limited cooling/heating power for chemical reactors.

- *State constraints:*
  State constraints may arise from the system's structure. We can considered the following examples: in non-holonomic mobile robots (*e.g.* , a cart with two forward driving wheels and two back wheels that cannot move sideways), constraints on forward and angular velocities; in underactuated systems (*e.g.* , a quadrotor), constraints on the angles; in chemical processes, constraints on the process variables.

- *Output constraints:*
  The output constraints are usually defined by the working environment. In motion planning, mobile robots steer from one point to another while avoiding the obstacles in the environment. In some applications, for safety or comfort reasons, output constraints are imposed by the user demand. For instance, limitations on the longitudinal velocities, angular velocities, temperatures.

The reference trajectory should consider the system's constraints. Such a trajectory that lies inside the admissible state domain and that does not violate the constraints is called a *feasible trajectory*. If system's constraints are not considered, the trajectory may be infeasible and the defined task may not be accomplished. Task unaccomplishment occurs for example when actuators hit their limits, and cannot deliver the actuation inputs desired by the controller.

It is therefore likely to have *a trajectory generator* that defines a set of feasible reference trajectories *i.e.* feed-forwarding trajectories that will fulfil the defined constraints. A systematic approach able to satisfy the constraints beforehand and to embed the constraint satisfaction directly in the control formulation is a key advantage. This fact motivates us to embed systematically the constraints in the trajectory design based on the flatness property.

1.1.2 Disturbances

For a physical system, besides constraints fulfilments, dealing with uncertain disturbances is essential to its well functioning. For example, a quadrotor should track its trajectory despite the wind perturbations or mass variation (when carrying a payload). However, when compensating the disturbances, the reference trajectories in time must induce a sufficiently slow motion *i.e.* decreasing its velocity or change its initial reference trajectory in order to avoid saturations of the quadrotors motors.

Consequently, an important control problem is how to design the reference trajectory in the *presence of disturbances* such that the constraints are fulfilled. In

control engineering practice, to cancel the disturbances, a *closed-loop controller* or a *feedback controller* is used to overcome the limitations of the open-loop controller. Moreover, the so-called *robust controllers* are able to compensate for disturbances that may affect the nominal evolution of the system. The system disturbances may be divided in two groups:

- *internal disturbances*, for, *e.g.,* unknown model or partly model mismatching, model parameter uncertainties, and
- *external disturbances*, for, *e.g.,* sensor noise, wind gusts or any other environmental disturbance.

To deal with the system disturbances, in this thesis, we employ the Model-Free Control. The *Model-Free Control* (MFC), introduced by Michel Fliess and Cédric Join [47, 48], already proved its power through a wide range of successful applications[1, 49, 93, 96, 105] and even, with experimental results [59, 60], where the system model and disturbances are unknown. Successful attempts for nonlinear MFC and for delay systems are presented in [26, 106] and in [37] respectively. The first detailed proof of stability of the MFC that provides insights to the tuning of the control parameters, was given in [35].

1.2 Thesis Organization and Contributions

This thesis is devoted to two problems: the first is the control of differentially flat systems with constraints, and the second is the application of the Model-Free Control using its robustness with respect to unmodelled dynamics and uncertainty of the system and its environment (see Figure 1.1).

Remark 1.1 *Each thesis chapter is self-explanatory and contains all the necessary introduction and preliminaries to its presented contents. In this introduction, we define the main lines of thought which we explain in more details in the corresponding chapters.*

The ultimate goal of this thesis project is to show that constraints can be fulfilled through the nominal feedforwarding inputs, and that internal and external disturbances can be managed by the MFC. Since the goal behind these two control problems (the constraints and the disturbances), are essential for almost every control system, it is interesting to note how little research is done in simultaneously dealing with these two problems. The difficulty arises from the need to quickly re-plan the feedforwarding part when a big disturbance occurs *i.e.* the feedback part

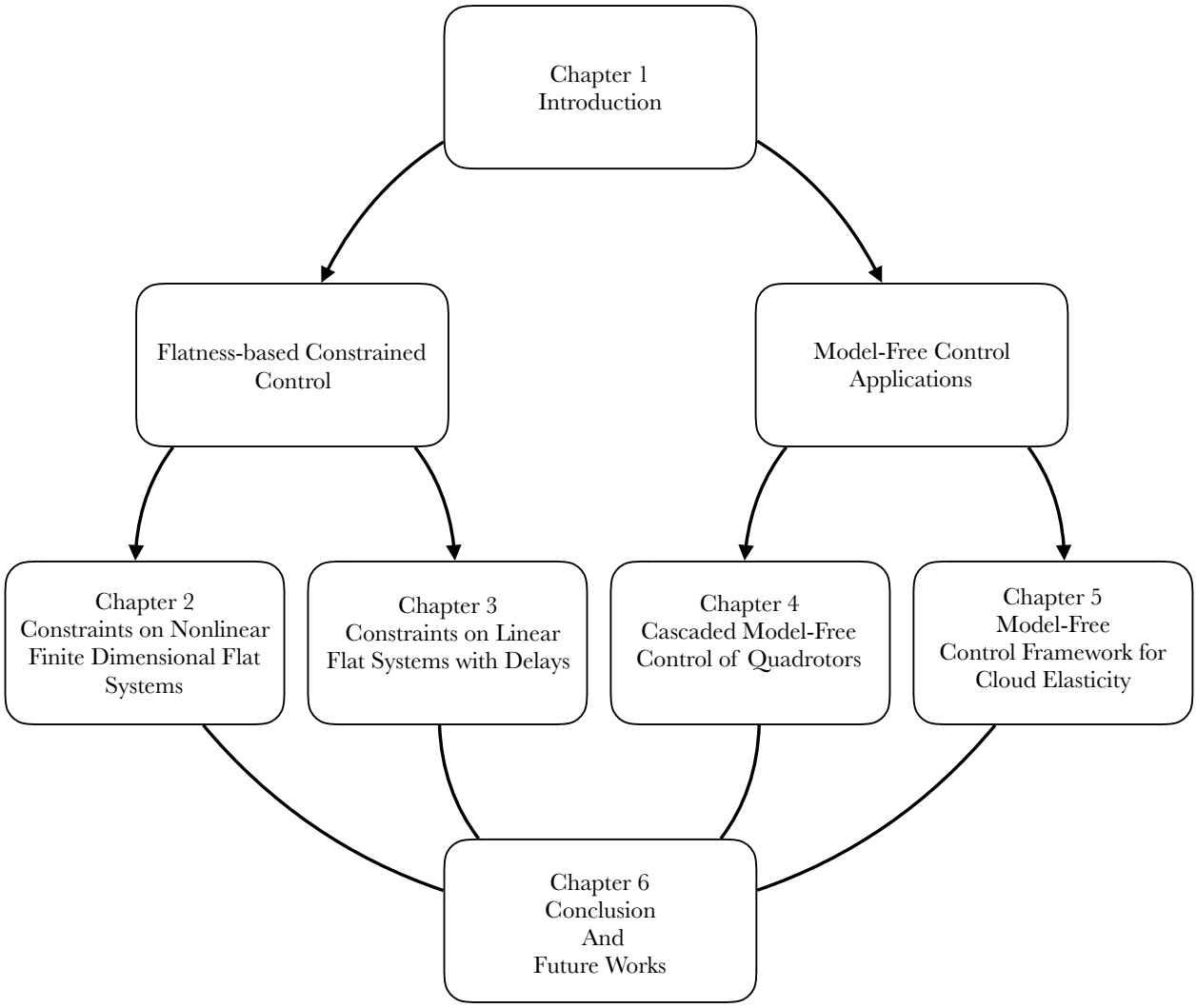


Figure 1.1: Organization of the thesis

increases. When such disturbances occur, a fast trajectory re-planning is essential to avoid actuator saturation. In [127], is discussed the control law composed of two parts: first part is constructed following differential flatness and second part is following the model-free control law for a special case where the disturbance and its derivative are assumed to be bounded.

But what happens when a big disturbance suddenly occurs? How to quickly re-plan a new reference trajectory ?

To that purpose, we studied the constrained trajectory approach presented in the *first part* of the thesis, where we define a set of feasible trajectories.

- In Chapter 2, we focus on trajectories generation rather than feedback transformations. Our goal is to write the *state/input constraints* in terms of

flat output and its derivatives. We propose to use Bézier curves as reference trajectories because of their useful properties (one of the keys in the design of our constrained open-loop control). We show how the inputs expressed by the flat outputs, each represented in terms of a Bézier curve, can be expressed as a combination of Bézier curves. We thus obtain explicit equations for the input/state control points as functions of the flat output control points. We then search for feasible regions of the Bézier control points *i.e.* a set of feasible reference trajectories. There are a number of existing related approaches [43, 62, 63, 79, 80, 120, 128, 129, 136], which all rely on optimization procedures to obtain a flat output reference trajectory. Moreover, differentially flat systems with constraints are generally attacked through the use of optimal control problems (see, e.g. [38, 42, 108, 111, 126])).

- In Chapter 3, we study the presence of delays in the state or the input that characterizes many natural as well as artificial systems. A large part of the control literature is thus devoted to the study of linear and nonlinear delay systems [119], but few results are able to handle constraints. Constraints in the input or the state variables of a delay system are usually tackled with numerical methods entailing the computation of trajectory invariant intervals [103] and positively invariant sets [32, 70], or more complex, but similar in spirit, Model Predictive Control approaches (see for instance [87, 107]). A systematic approach capable of embedding the constraint satisfaction directly in the control formulation is still lacking for delay systems.

The *second part* of the thesis presents the Model-Free Control framework [48] which estimates and cancels the unknown disturbances and/or unknown system dynamics. The contribution of the second part of the thesis lies in two applications of the Model-Free Control (MFC) of different nature:

- In Chapter 4, we propose a controller design that avoids the quadrotor's system identification procedures while staying robust with respect to the endogenous (the control performance is independent of any mass change, inertia, gyroscopic or aerodynamic effects) and exogenous disturbances (wind, measurement noise). To reach our goal, based on the cascaded structure of a quadrotor, we divide the system into positional and attitude subsystems each controlled by an independent *Model-Free controller* of second-order dynamics. Then, we give proof results on the practical stability of the proposed control

design. We validate our control approach in three realistic scenarios: in presence of unknown measurement noise, with unknown time-varying wind disturbances and mass variation.

- In Chapter 5, we use Model-Free Control to control the "horizontal elasticity" of a Cloud Computing system. When compared to the commercial "Auto-Scaling" algorithms, our easily implementable approach behaves better, even with sharp workload fluctuations. This is confirmed by experiments on the Amazon Web Services (AWS) public cloud.

- Finally, in Chapter 6, we summarize the research work of the thesis and propose, as a perspective, a unified gray-box framework using the two above presented studies.

2

Constraints on Nonlinear Finite Dimensional Flat Systems

Contents

Abstract: This chapter presents an approach to embed the input/state/output constraints in a unified manner into the trajectory design for differentially flat systems. To that purpose, we specialize the flat outputs (or the reference trajectories) as Bézier curves. Using the flatness property, the system's inputs/states can be expressed as a combination of Bézier curved flat outputs and their derivatives. Consequently, we explicitly obtain the expressions of the control points of the inputs/states Bézier curves as a combination of the control points of the flat outputs. By applying desired constraints to the latter control points, we find the feasible regions for the output Bézier control points i.e. *a set of feasible reference trajectories.*

2.1 Chapter overview

Motivation

The control of nonlinear systems subject to *state and input constraints* is one of the major challenges in control theory. Traditionally, in the control theory literature, the reference trajectory to be tracked is specified in advance. Moreover for some applications, for instance, the quadrotor trajectory tracking, selecting the *right trajectory* in order to avoid obstacles while not damaging the actuators is of crucial importance.

In the last few decades, Model Predictive Control (MPC) [22, 89] has achieved a big success in dealing with constrained control systems. Model predictive control is a form of control in which the current control law is obtained by solving, at each sampling instant, a finite horizon open-loop optimal control problem, using the current state of the system as the initial state; the optimization yields an optimal control sequence and the first control in this sequence is applied to the system. It has been widely applied in petro-chemical and related industries where satisfaction of constraints is particularly important because efficiency demands operating points on or close to the boundary of the set of admissible states and controls.

The optimal control or MPC maximize or minimize a defined performance criterion chosen by the user. The optimal control techniques, even in the case without constraints are usually discontinuous, which makes them less robust and more dependent of the initial conditions. In practice, this means that the delay formulation renders the numerical computation of the optimal solutions difficult.

A large part of the literature working on constrained control problems is focused on optimal trajectory generation [42, 79]. These studies are trying to find feasible trajectories that optimize the performance following a specified criterion. Defining

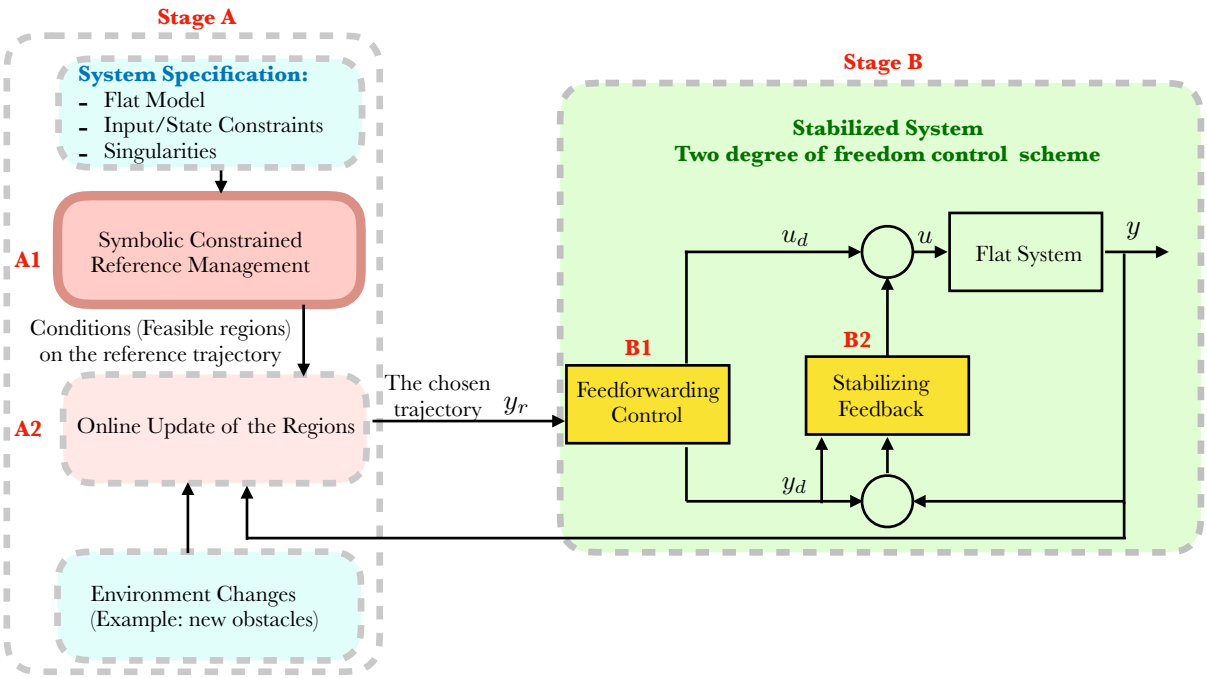


Figure 2.1: Two degrees of freedom control scheme overview

the right criterion to optimize may be a difficult problem in practice. Usually, in such cases, the feasible and the optimal trajectory are not too much different. For example, in the case of autonomous vehicles [76], due to the dynamics, limited curvature, and under-actuation, a vehicle often has few options for how it changes lines on highways or how it travels over the space immediately in front of it. Regarding the complexity of the problem, searching for a feasible trajectory is *easier*, especially in the case where we need *real-time re-planning* [68, 69]. Considering that the evolution of transistor technologies is reaching its limits, low-complexity controllers that can take the constraints into account are of considerable interest. The same remark is valid when the system has sensors with limited performance.

Research objective and contribution

In this chapter, we propose a novel trajectory-based framework to deal with system constraints. We are answering the following question:

Question 2.1 *How to design a set of the reference trajectories (or the feed-forwarding trajectories) of a nonlinear system such that the input, state and/or output constraints are fulfilled?*

For that purpose, we divide the control problem in two stages (see Figure 2.1). Our objective will be to elaborate a *constrained reference trajectory management* (Stage A) which is meant to be applied to already pre-stabilized systems (Stage B).

Unlike other receding horizon approaches which attempt to solve stabilization, tracking, and constraint fulfilment at the same time, we assume that in *Stage B*, a primal controller has already been designed to stabilize the system which provide nice tracking properties in the absence of constraints. In stage B, we employ the two-degree of freedom design consisting of a constrained trajectory design (constrained feedfowarding) and a feedback control.

In *Stage A*, the constraints are embedded in the flat output trajectory design. Thus, *our constrained trajectory generator* defines a feasible open-loop reference trajectory satisfying the states and/or control constraints that *a primal feedback controller* will track and stabilize around.

To construct Stage A we first take advantage of the *differential flatness property* which serves as a base to construct our method. The differential flatness property yields exact expressions for the state and input trajectories of the system through trajectories of a flat output and its derivatives without integrating any differential equation. The latter property allows us to map the state/input constraints into the flat output trajectory space.

Then, in our symbolic approach (stage A1), we assign a Bézier curve to each flat output where the parameter to be chosen are the so-called *control points* (yielding a finite number of variables on a finite time horizon) given in a symbolic form. This kind of representation naturally offers several algebraic operations like the sum, the difference and multiplication, and affords us to preserve the explicit functions structure without employing discrete numerical methods. The advantage to deal with the constraints *symbolically*, rather than numerically, lies in that the symbolic solution explicitly depends on the control points of the reference trajectory. This allows to study how the input or state trajectories are influenced by the reference trajectory.

We find symbolic conditions on the trajectory control points such that the states/inputs constraints are fulfilled.

We translate the state/input constraints into constraints on the reference trajectory control points and we wish to reduce the solution of the systems of equations/inequations into a simpler one. Ideally, we want to find the exact set of solutions *i.e.* the constrained subspace.

We explain how this symbolic constrained subspace representation can be used for *constrained feedforwarding trajectory selection*. The stage A2 can be done in two different ways.

- When a system should track a trajectory in a *static known environment*, then the exact set of feasible trajectories is found and the trajectory is fixed by our choice. If the system's environment changes, we only need to re-evaluate the exact symbolic solution with new numerical values.
- When a system should track a trajectory in an *unknown environment with moving objects*, then, whenever necessary, the reference design modifies the reference supplied to a primal control system so as to enforce the fulfilment of the constraints. This second problem is not addressed in the thesis.

Our approach is *not based on any kind of optimization* nor does it need computations for a given numerical value at each sampling step. We determine a set of feasible trajectories through the system constrained environment that enable a controller to make quick real-time decisions. For systems with singularities, we can isolate the singularities of the system by considering them as additional constraints.

Existing Methods

- Considering actuator constraints based on the derivatives of the flat output (for instance, the jerk [58, 143], snap [91]) can be too conservative for some systems. The fact that a feasible reference trajectory is designed following the system model structure allows to choose a quite aggressive reference trajectory.
- In contrast to [136], we characterize the whose set of viable reference trajectories which take the constraints into account.
- In [131], the problem of constrained trajectory planning of differentially flat systems is cast into a simple quadratic programming problem ensuing computational advantages by using the flatness property and the B-splines curve's properties. They simplify the computation complexity by taking advantage of the B-spline minimal (resp. maximal) control point. The simplicity comes at the price of having only minimal (resp. maximal) constant constraints that eliminate the possible feasible trajectories and renders this approach conservative.
- In [63], an inversion-based design is presented, in which the transition task between two stationary set-points is solved as a two-point boundary value problem. In this approach, the trajectory is defined as polynomial where only the initial and final states can be fixed.
- The thesis of Bak [7] compared existing methods to constrained controller design (anti-windup, predictive control, nonlinear methods), and introduced a nonlinear gain scheduling approach to handle actuator constraints.

Outline

This chapter is organized as follows:

- In section 2.2, we recall the notions of differential flatness for finite dimensional systems.
- In section 2.3, we present our problem statement for the constraints fulfilment through the reference trajectory.
- In section 2.4, we detail the flat output parameterization given by the Bézier curve, and its properties.
- In section 2.5, we give the whole procedure in establishing reference trajectories for constrained open-loop control. We illustrate the procedure through two applications in section 2.7.
- In section 2.6, we present the two methods that we have used to compute the constrained set of feasible trajectories.

2.2 Differential flatness overview

The concept of *differential flatness* was introduced in [55, 56] for non-linear finite dimensional systems. By the means of differential flatness, a non-linear system can be seen as a controllable linear system through a dynamical feedback.

A model shall be described by a differential system as:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \tag{2.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ denote the *state variables* and $\boldsymbol{u} \in \mathbb{R}^m$ the *input vector*. Such a system is said to be *flat* if there exists a set of *flat outputs (or linearizing outputs)* (equal in number to the number of inputs) given by

$$\boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}, \dot{\boldsymbol{u}}, ..., \boldsymbol{u}^{(r)}) \tag{2.2}$$

with $r \in \mathbb{N}$ such that the components of $\boldsymbol{y} \in \mathbb{R}^m$ and all their derivatives are functionally independent and such that we can parametrize every solution $(\boldsymbol{x}, \boldsymbol{u})$ of (2.1) in some dense open set by means of the flat output $\boldsymbol{y}$ and its derivatives up to a finite order q:

$$\boldsymbol{x} = \psi(\boldsymbol{y}, \dot{\boldsymbol{y}}, ..., \boldsymbol{y}^{(q-1)}), \tag{2.3a}$$

$$\boldsymbol{u} = \zeta(\boldsymbol{y}, \dot{\boldsymbol{y}}, ..., \boldsymbol{y}^{(q)}) \tag{2.3b}$$

where (ψ, ζ) are *smooth functions* that give the trajectories of $\boldsymbol{x}$ and $\boldsymbol{u}$ as functions of the flat outputs and their time derivatives. The preceding expressions in (2.3), will be used to obtain the so called *open-loop controls*. The differential flatness found numerous applications, non-holonomic systems, among others (see [126] and the references therein).

In the context of feedforwarding trajectories, the "degree of continuity" or the smoothness of the reference trajectory (or curve) is one of the most important factors. The smoothness of a trajectory is measured by the number of its continuous derivatives. We give the definitions on the trajectory continuity when it is represented by a parametric curve in the Appendix 2.B.

2.3 Problem statement: Trajectory constraints fulfilment

Notation

Given the scalar function $z \in C^\kappa(\mathbb{R}, \mathbb{R})$ and the number $\alpha \in \mathbb{N}$, we denote by $\boldsymbol{z}^{\langle\alpha\rangle}$ the tuple of derivatives of z up to the order $\alpha \leqslant \kappa$: $\boldsymbol{z}^{\langle\alpha\rangle} = z,\, \dot{z}, \ddot{z}, \ldots,\, z^{(\alpha)}$. Given the vector function $\boldsymbol{v} = (v_1, \ldots, v_q)$, $v_i \in C^\kappa(\mathbb{R}, \mathbb{R})$ and the tuple $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_q)$, $\alpha_i \in \mathbb{N}$, we denote by $\boldsymbol{v}^{\langle\alpha\rangle}$ the tuple of derivatives of each component v_i of $\boldsymbol{v}$ up to its respective order $\alpha_i \leqslant \kappa$: $\boldsymbol{v}^{\langle\alpha\rangle} = v_1, \ldots, v_1^{(\alpha_1)},\, v_2, \ldots, v_2^{(\alpha_2)},\, \ldots\, ,\, v_q, \ldots, v_q^{(\alpha_q)}$.

2.3.1 General problem formulation

Consider the nonlinear system

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{2.4}$$

with state vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ and control input $\boldsymbol{u} = (u_1, \ldots, u_m)$, $x_i, u_j \in C^\kappa([0, +\infty), \mathbb{R})$ for a suitable $\kappa \in \mathbb{N}$. We assume the state, the input and their derivatives to be subject to both inequality and equality constraints of the form

$$C_i(\boldsymbol{x}^{\langle\alpha_i^x\rangle}(t), \boldsymbol{u}^{\langle\alpha_i^u\rangle}(t)) \leqslant 0 \qquad \forall t \in [0, T],\ \forall i \in \{1, \ldots, \nu^{\text{in}}\} \tag{2.5a}$$

$$D_j(\boldsymbol{x}^{\langle\beta_j^x\rangle}(t), \boldsymbol{u}^{\langle\beta_j^u\rangle}(t)) = 0 \qquad \forall t \in I_j,\ \forall j \in \{1, \ldots, \nu^{\text{eq}}\} \tag{2.5b}$$

with each I_j being either $[0, T]$ (continuous equality constraint) or a discrete set $\{t_1, \ldots, t_\gamma\}$, $0 \leq t_1 \leqslant \cdots \leqslant t_\gamma \leqslant T < +\infty$ (discrete equality constraint), and $\alpha_i^x, \beta_j^x \in \mathbb{N}^n$, $\alpha_i^u, \beta_j^u \in \mathbb{N}^m$. We stress that the relations (2.5) specify objectives (and

constraints) on the finite interval $[0, T]$. Objectives can be also formulated as a concatenation of sub-objectives on a union of sub-intervals, provided that some continuity and/or regularity constraints are imposed on the boundaries of each sub-interval. Here we focus on just one of such intervals.

Our aim is to characterise the set of input and state trajectories $(\boldsymbol{x}, \boldsymbol{u})$ satisfying the system's equations (2.4) and the constraints (2.5). More formally we state the following problem.

Problem 2.1 (Constrained trajectory set) *Let $\mathscr{C}$ be a subspace of $C^\kappa([0, +\infty), \mathbb{R})$. Constructively characterise the set $\mathscr{C}^{\text{cons}} \subseteq \mathscr{C}^{n+m}$ of all extended trajectories $(\boldsymbol{x}, \boldsymbol{u})$ satisfying the system (2.4) and the constraints (2.5).*

Problem 2.1 can be considered as a generalisation of a constrained reachability problem (see for instance [43]). In such a reachability problem the stress is usually made on initial and final set-points and the goal is to find a suitable input to steer the state from the initial to the final point while possibly fulfilling the constraints. Here, we wish to give a functional characterisation of the overall set of extended trajectories $(\boldsymbol{x}, \boldsymbol{u})$ satisfying some given differential constraints. A classical constrained reachability problem can be cast in the present formalism by limiting the constraints C_i and D_j to $\boldsymbol{x}$ and $\boldsymbol{u}$ (and not their derivatives) and by forcing two of the equality constraints to coincide with the initial and final set-points.

Problem 2.1 is difficult to be addressed in its general setting. To simplify the problem, in the following we make some restrictions to the class of systems and to the functional space $\mathscr{C}$. As a first assumption we limit the analysis to differentially flat systems [55].

2.3.2 Constraints in the flat output space

Let us assume that system (2.4) is differentially flat with flat output[1]

$$\boldsymbol{y} = (y_1, \ldots, y_m) = h(\boldsymbol{x}, \boldsymbol{u}^{\langle \rho^u \rangle}), \tag{2.6}$$

with $\rho^u \in \mathbb{N}^m$. Following Equation (2.3), the parameterisation or the feedforwarding trajectories associated to the reference trajectory $\boldsymbol{y_r}$ is

$$\boldsymbol{x}_r = \psi(\boldsymbol{y_r}^{\langle \eta^x \rangle}) \tag{2.7a}$$

$$\boldsymbol{u}_r = \zeta(\boldsymbol{y_r}^{\langle \eta^u \rangle}), \tag{2.7b}$$

[1] We recall that the flat output $\boldsymbol{y}$ has the same dimension m as the input vector $\boldsymbol{u}$.

with $\eta^x \in \mathbb{N}^n$ and $\eta^u \in \mathbb{N}^m$.

Through the first step of the dynamical extension algorithm [44], we get the flat output dynamics

$$\begin{cases} y_1^{(k_1)} = \phi_1(\boldsymbol{y}^{\langle \mu_1^y \rangle}, \boldsymbol{u}^{\langle \mu_1^u \rangle}) \\ \quad \vdots \\ y_m^{(k_m)} = \phi_m(\boldsymbol{y}^{\langle \mu_m^y \rangle}, \boldsymbol{u}^{\langle \mu_m^u \rangle}), \end{cases} \tag{2.8}$$

with $\mu_i^y = (\mu_{i1}^y, \ldots, \mu_{im}^y) \in \mathbb{N}^m$, $\mu_i^u = (\mu_{i1}^u, \ldots, \mu_{im}^u) \in \mathbb{N}^m$ and $k_i > \max_j \mu_{ji}^y$. The original n-dimensional dynamics (2.4) and the K-dimensional flat output dynamics (2.8) ($K = \sum_i k_i$) are in one-to-one correspondence through (2.6) and (2.7). Therefore, the constraints (2.5) can be re-written as

$$\Gamma_i(\boldsymbol{y_r}^{\langle \omega_i^{\text{in}} \rangle}) \leqslant 0 \qquad \forall t \in [0, T],\ \forall i \in \{1, \ldots, \nu^{\text{in}}\} \tag{2.9a}$$

$$\Delta_j(\boldsymbol{y_r}^{\langle \omega_j^{\text{eq}} \rangle}) = 0 \qquad \forall t \in I_j,\ \forall j \in \{1, \ldots, \nu^{\text{eq}}\} \tag{2.9b}$$

with

$$\Gamma_i(\boldsymbol{y_r}^{\langle \omega_i^{\text{in}} \rangle}) = C_i((\psi(\boldsymbol{y_r}^{\langle \eta^x \rangle}))^{\langle \alpha_i^x \rangle}, \zeta(\boldsymbol{y_r}^{\langle \eta^u \rangle})^{\langle \alpha_i^u \rangle}),$$
$$\Delta_j(\boldsymbol{y_r}^{\langle \omega_j^{\text{eq}} \rangle}) = D_j((\psi(\boldsymbol{y_r}^{\langle \eta^x \rangle})^{\langle \beta_j^x \rangle}, \zeta(\boldsymbol{y_r}^{\langle \eta^u \rangle})^{\langle \beta_j^u \rangle})$$

and $\omega_i^{\text{in}}, \omega_j^{\text{eq}} \in \mathbb{N}^m$.

Remark 2.1 *We may use the same result to embed an input rate constraint* $\dot{\boldsymbol{u}}_{\boldsymbol{r}}$.

Thus, Problem 2.1 can be transformed in terms of the flat output dynamics (2.8) and the constraints (2.9) as follows.

Problem 2.2 (Constrained flat output set) [2] *Let* $\mathscr{C}_y$ *be a subspace of* $C^p([0, +\infty), \mathbb{R})$ *with* $p = \max((k_1, \ldots, k_m), \omega_1^{\text{in}}, \ldots, \omega_{\nu^{\text{in}}}^{\text{in}}, \omega_1^{\text{eq}}, \ldots, \omega_{\nu^{\text{eq}}}^{\text{eq}})$. *Constructively characterise the set* $\mathscr{C}_y^{\text{cons}} \subseteq \mathscr{C}_y^m$ *of all flat outputs satisfying the dynamics (2.8) and the constraints (2.9).*

Working with differentially flat systems allows us to translate, in a unified fashion, all the state and input constraints as constraints in the flat outputs and their derivatives (See (2.9)). We remark that ψ and ζ in (2.7) are such that $\psi(\boldsymbol{y}^{\langle \eta_x \rangle})$ and $\zeta(\boldsymbol{y}^{\langle \eta_u \rangle})$ satisfy the dynamics of system (2.4) by construction. In other words, the extended trajectories $(\boldsymbol{x}, \boldsymbol{u})$ of (2.4) are in one-to-one correspondence with $\boldsymbol{y} \in \mathscr{C}_y^m$ given by (2.6). Hence, choosing $\boldsymbol{y}$ solution of Problem 2.2 ensures that $\boldsymbol{x}$ and $\boldsymbol{u}$ given by (2.7) are solutions of Problem 2.1.

[2] Here the max operator is applied elementwise on each vector.

2.3.3 Problem specialisation

For any practical purpose, one has to choose the functional space $\mathscr{C}_y$ to which all components of the flat output belong. Instead of making reference to the space $\mathscr{C}^{\text{gen}} := C^p([0,+\infty),\mathbb{R})$, mentioned in the statement of Problem 2.1, we focus on the space $\mathscr{C}_T^{\text{gen}} := C^p([0,T],\mathbb{R})$. Indeed, the constraints (2.9) specify finite-time objectives (and constraints) on the interval $[0,T]$. Still, the problem exhibits an infinite dimensional complexity, whose reduction leads to choose an approximation space $\mathscr{C}^{\text{app}}$ that is dense in $\mathscr{C}_T^{\text{gen}}$. A possible choice is to work with parametric functions expressed in terms of basis functions like, for instance, Bernstein-Bézier, Chebychev or Spline polynomials.

A scalar Bézier curve of degree $N \in \mathbb{N}$ in the Euclidean space $\mathbb{R}$ is defined as

$$P(s) = \sum_{j=0}^{N} \alpha_j B_{jN}(s), \qquad s \in [0,1]$$

where the $\alpha_j \in \mathbb{R}$ are the control points and $B_{jN}(s) = \binom{N}{j}(1-s)^{N-j}s^j$ are Bernstein polynomials [34]. For sake of simplicity, we set here $T = 1$ and we choose as functional space

$$\mathscr{C}^{\text{app}} = \left\{ \sum_{0}^{N} \alpha_j B_j N | N \in \mathbb{N}, (\alpha_j)_0^N \in \mathbb{R}^{N+1}, B_j \in \mathcal{C}^0([0,1],\mathbb{R}) \right\} \tag{2.10}$$

The set of Bézier functions of generic degree has the very useful property of being closed with respect to addition, multiplication, degree elevation, derivation and integration operations (see section 2.4). As a consequence, any polynomial integro-differential operator applied to a Bézier curve, still produces a Bézier curve (in general of different degree). Therefore, if the flat outputs $\boldsymbol{y}$ are chosen in $\mathscr{C}^{\text{app}}$ and the operators $\Gamma_i(\cdot)$ and $\Delta_j(\cdot)$ in (2.9) are integro-differential polynomials, then such constraints can still be expressed in terms of Bézier curves in $\mathscr{C}^{\text{app}}$. We stress that, if some constraints do not admit such a description, we can still approximate them up to a prefixed precision ε as function in $\mathscr{C}^{\text{app}}$ by virtue of the denseness of $\mathscr{C}^{\text{app}}$ in $\mathscr{C}_1^{\text{gen}}$. Hence we assume the following.

Assumption 2.1 *Considering each flat output $y_r \in \mathscr{C}^{\text{app}}$ defined as*

$$y_r = \sum_{j=0}^{N} \alpha_j B_{jN}(s),$$

the constraints (2.9) can be written as

$$\Gamma_i(\boldsymbol{y_r}^{\langle \omega_i^{\text{in}} \rangle}) = \sum_{k=0}^{N_i^{\text{in}}} \lambda_{ik} B_{kN}(s), \tag{2.11}$$

$$\Delta_j(\boldsymbol{y_r}^{\langle \omega_j^{\text{eq}} \rangle}) = \sum_{k=0}^{N_i^{\text{eq}}} \delta_{jk} B_{kN}(s) \tag{2.12}$$

where

$$\begin{aligned} \lambda_{ik} &= r_{ik}^{\text{in}}(\alpha_0, \ldots, \alpha_N) \\ \delta_{jk} &= r_{jk}^{\text{eq}}(\alpha_0, \ldots, \alpha_N) \\ r_{ik}^{\text{in}}, r_{jk}^{\text{eq}} &\in \mathbb{R}[\alpha_0, \ldots, \alpha_N] \end{aligned}$$

i.e. *the λ_{ik} and δ_{jk} are polynomials in the $\alpha_0, \ldots, \alpha_N$.*■

Set the following expressions asν^{in}

$$\begin{aligned} r^{\text{in}} &= (r_{1,0}^{\text{in}}, \ldots, r_{\nu^{\text{in}}, N_{\nu^{\text{in}}}^{in}}^{\text{in}}), \\ r^{\text{eq}} &= (r_{1,0}^{\text{eq}}, \ldots, r_{\nu^{\text{eq}}, N_{\nu^{\text{eq}}}^{eq}}^{\text{eq}}), \\ r &= (r^{\text{in}}, r^{\text{eq}}), \end{aligned}$$

the control point vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)$, and the basis function vector $\boldsymbol{B} = (B_{1N}, \ldots, B_{NN})$. Therefore, we obtain a semi-algebraic set defined as:

$$\mathscr{I}(r, \mathbb{A}) = \left\{ \boldsymbol{\alpha} \in \mathbb{A} \mid r^{\text{in}}(\boldsymbol{\alpha}) \leqslant 0, r^{\text{eq}}(\boldsymbol{\alpha}) = 0 \right\}$$

for any parallelotope

$$\mathbb{A} = [\underline{\alpha}_0, \bar{\alpha}_0] \times \cdots \times [\underline{\alpha}_N, \bar{\alpha}_N], \underline{\alpha}_i, \bar{\alpha}_i \in \mathbb{R} \cup \{-\infty, \infty\}, \underline{\alpha}_i < \bar{\alpha}_i \tag{2.13}$$

Thus $\mathscr{I}(r, \mathbb{A})$ is a semi-algebraic set associated to the constraints (2.9). The parallelotope $\mathbb{A}$ represents the trajectory sheaf of available trajectories, among which the user is allowed to choose a reference. The semi-algebraic set $\mathscr{I}(r, \mathbb{A})$ represents how the set $\mathbb{A}$ is transformed in such a way that the trajectories fulfill the constraints (2.9). Then, picking an $\boldsymbol{\alpha}$ in $\mathscr{I}(r, \mathbb{A})$ ensures that $y_r = \boldsymbol{\alpha}\boldsymbol{B}$ automatically satisfies the constraints (2.9).
The Problem 2.2 is then reformulated as :

Problem 2.3 *For any fixed parallelotope $\mathbb{A}$, constructively characterise the semi-algebraic set $\mathscr{I}(r, \mathbb{A})$.*

This may be done through exact, symbolic techniques (such as, *e.g.* the Cylidrical Algebraic Decomposition) or through approximation techniques yielding outer approximations $\mathscr{I}_l^{out}(r,\mathbb{A}) \supseteq \mathscr{I}(r,\mathbb{A})$ and inner approximations $\mathscr{I}_l^{inn}(r,\mathbb{A}) \subseteq \mathscr{I}(r,\mathbb{A})$ with $\lim_{l\to\infty} \mathscr{I}_l^{out} = \lim_{l\to\infty} \mathscr{I}_l^{inn} = \mathscr{I}$. ■
This characterisation shall be useful to extract inner approximations of a special type yielding trajectory sheaves included in $\mathscr{I}(r,\mathbb{A})$. A specific example of this type of approximations will consist in disjoint unions of parallelotopes:

$$\mathscr{I}_l^{inn}(r,\mathbb{A}) = \bigcup_{j\in I_l} \mathbb{B}_{l,j}, \quad \forall i,j \in I_l, \mathbb{B}_{l,i} \cap \mathbb{B}_{l,j} = \emptyset \tag{2.14}$$

This class of inner approximation is of practical importance for end users, as the applications in Section 2.7 illustrate.

2.3.4 Closed-loop trajectory tracking

So far this chapter has focused on the design of open-loop trajectories while assuming that the system model is perfectly known and that the initial conditions are exactly known. When the reference open-loop trajectories $(\boldsymbol{x_r}, \boldsymbol{u_r})$ are well-designed *i.e.* respecting the constraints and avoiding the singularities, as discussed above, the system is close to the reference trajectory. However, to cope with the environmental disturbances and/or small model uncertainties, the tracking of the constrained open-loop trajectories should be made robust using *feedback control*. The feedback control guarantees the stability and a certain robustness of the approach, and is called the second degree of freedom of the primal controller (Stage B2 in figure 2.1).

We recall that some flat systems can be transformed via endogenous feedback and coordinate change to a linear dynamics [55, 126]. To make this chapter self-contained, we briefly discuss the *closed-loop trajectory tracking* as presented in [88].

Consider a differentially flat system with flat output $\boldsymbol{y} = (y_1, \ldots, y_m)$ (m being the number of independent inputs of the system). Let $\boldsymbol{y}_r(t) \in C^\eta(\mathbb{R})$ be a reference trajectory for $\boldsymbol{y}$. Suppose the desired open-loop state/ input trajectories $(x_r(t), u_r(t))$ are generated offline. We need now a feedback control to track them.

Since the nominal open-loop control (or the feedforward input) linearizes the system, we can take a simple linear feedback, yielding the following closed-loop error dynamics:

$$\boldsymbol{e}^{(\eta)} + \lambda_{\eta-1}\boldsymbol{e}^{(\eta-1)} + \cdots + \lambda_1\dot{\boldsymbol{e}} + \lambda_0\boldsymbol{e} = 0 \tag{2.15}$$

where $\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{y}_r$ is the tracking error and the coefficients $\Lambda = [\lambda_0, \ldots, \lambda_{\eta-1}]$ are chosen to ensure an asymptotically stable behaviour (see e.g. [56]).

Remark 2.2 *Note that this is not true for all flat systems, in [66] can be found an example of flat system with nonlinear error dynamics.*

Now let $(\boldsymbol{x}, \boldsymbol{u})$ be the closed-loop trajectories of the system. These variables can be expressed in terms of the flat output $\boldsymbol{y}$ as:

$$\boldsymbol{x} = \psi(\boldsymbol{y}^{\langle \eta-1 \rangle}), \quad \boldsymbol{u} = \zeta(\boldsymbol{y}^{\langle \eta \rangle}) \tag{2.16}$$

Then, the associated reference open-loop trajectories $(\boldsymbol{x_r}, \boldsymbol{u_r})$ are given by

$$\boldsymbol{x_r} = \psi({\boldsymbol{y_r}}^{\langle \eta-1 \rangle}), \quad \boldsymbol{u_r} = \zeta({\boldsymbol{y_r}}^{\langle \eta \rangle})$$

Therefore,

$$\boldsymbol{x} = \psi(\boldsymbol{y}^{\langle \eta-1 \rangle}) = \psi({\boldsymbol{y_r}}^{\langle \eta-1 \rangle} + \boldsymbol{e}^{\langle \eta-1 \rangle})$$

and

$$\boldsymbol{u} = \zeta(\boldsymbol{y}^{\langle \eta \rangle}) = \zeta({\boldsymbol{y_r}}^{\langle \eta \rangle} + \boldsymbol{e}^{\langle \eta \rangle}, -\Lambda \boldsymbol{e}^{\langle \eta \rangle}).$$

As further demonstrated in [88][See Section 3.3], since the tracking error $\boldsymbol{e} \to 0$ as $t \to \infty$ that means $\boldsymbol{x} \to \boldsymbol{x_r}$ and $\boldsymbol{u} \to \boldsymbol{u_r}$.

Besides the linear controller (Equation (2.15)), many different linear and nonlinear feedback controls can be used to ensure convergence to zero of the tracking error. For instance, sliding mode control, high-gain control, passivity based control, model-free control, among others.

Remark 2.3 *An alternative method to the feedback linearization, is the exact feedforward linearization presented in [67] where the problem of type "division by zero" in the control design is easily avoided. This control method removes the need for asymptotic observers since in its design the system states information is replaced by their corresponding reference trajectories. The robustness of the exact feedforwarding linearization was analyzed in [69].*

2.4 Preliminaries on Symbolic Bézier trajectory

To create a trajectory that passes through several points, we can use approximating or interpolating approaches. The interpolating trajectory that passes through the points is prone to oscillatory effects (more unstable), while the approximating trajectory like the Bézier curve or B-Spline curve is more convenient since it only approaches defined so-called *control points* [34] and have simple geometric

interpretations. The Bézier/B-spline curve can be handled by conveniently handling the curve's control points.
The main reason in choosing the Bézier curves over the B-Splines curves, is the simplicity of their arithmetic operators presented further in this Section. Despite the nice local properties of the B-spline curve, the direct symbolic multiplication[3] of B-splines lacks clarity and has partly known practical implementation [98].

In the following Section, we start by presenting the *Bézier curve* and its properties. Bézier curves are chosen to construct the reference trajectories because of their nice properties (smoothness, strong convex hull property, derivative property, arithmetic operations). They have their own type basis function, known as the Bernstein basis, which establishes a relationship with the so-called control polygon. A complete discussion about Bézier curves can be found in [116]. Here, some basic and key properties are recalled as a preliminary knowledge.

2.4.1 Definition of the Bézier curve

A Bézier curve is a parametric one that uses the Bernstein polynomials as a basis. An nth degree Bézier curve is defined by

$$f(t) = \sum_{j=0}^{N} c_j B_{j,N}(t), \quad 0 \leqslant t \leqslant 1 \tag{2.17}$$

where the c_j are *the control points* and the basis functions $B_{j,N}(t)$ are the *Bernstein polynomials* (see Figure 2.2). The $B_{j,N}(t)$ can be obtained explicitly by:

$$B_{j,N}(t) = \binom{N}{j}(1-t)^{N-j}t^j \text{ for } j = 0, \ldots, N.$$

or by recursion with the De Casteljau formula:

$$B_{j,N}(t) = (1-t)B_{j,N-1}(t) + tB_{j-1,N-1}(t).$$

2.4.2 Bézier properties

For the sake of completeness, we here list some important Bézier-Bernstein properties.

[3]The multiplication operator is essential when we want to work with polynomial systems.

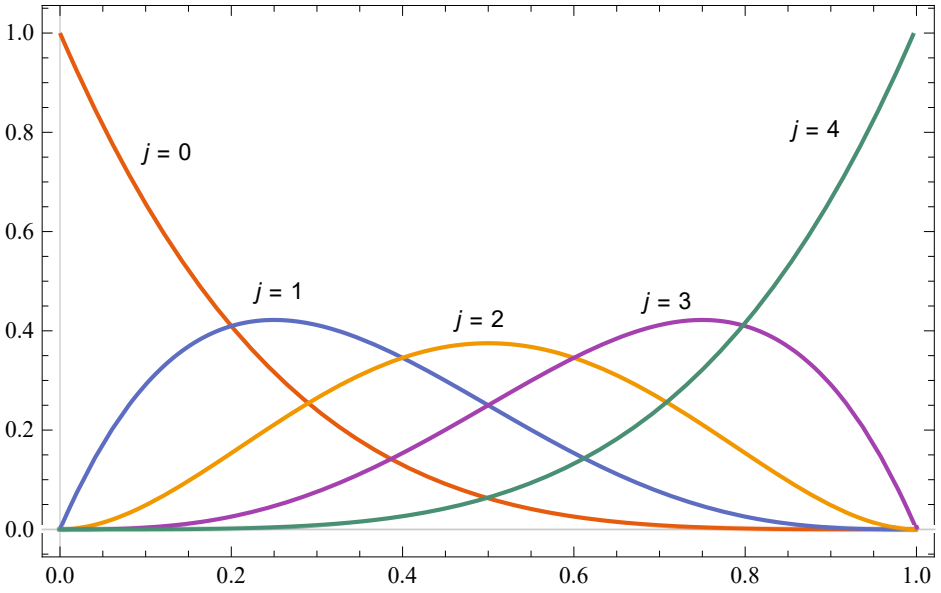


Figure 2.2: Bernstein Basis for degree $N = 4$.

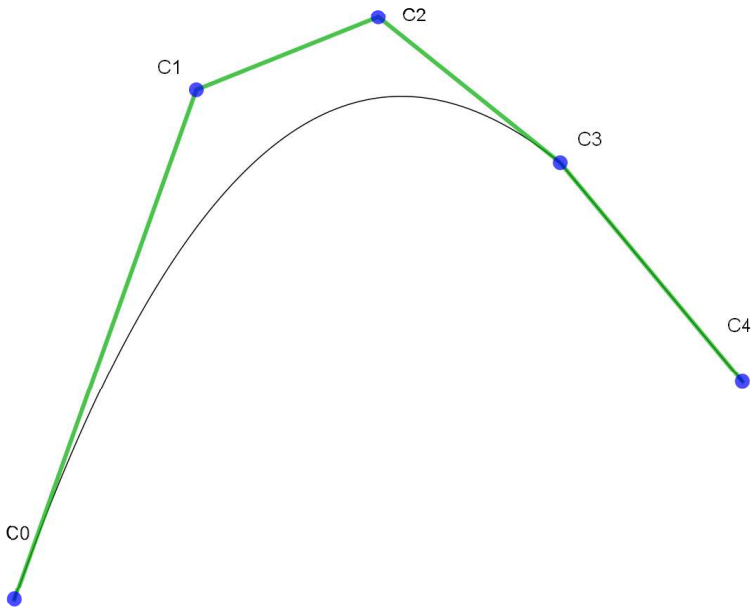


Figure 2.3: The convex hull property for Bézier curve ($N = 4$) with control points $c_j (j = 0, \ldots, 4)$.

Lemma 2.1 *Let n be a non-negative polynomial degree. The Bernstein functions have the following properties:*

1. *Partition of unity.* $\sum_{j=0}^{n} B_{j,N}(t) \equiv 1$
   This property ensures that the relationship between the curve and its defining Bézier points is invariant under affine transformations.

2. *Positivity. If* $t \in [0,1]$ *then* $B_{j,N}(t) > 0$.
   It guarantees that the curve segment lies completely within the convex hull of the control points (see Figure 2.3).

3. *Tangent property. For the start and end point, this guarantees* $f(0) = c_0$ *and* $f(1) = c_N$ *but the curve never passes through the intermediate control points.*

4. *Smoothness.* $B_{j,N}(t)$ *is* $N-1$ *times continuously differentiable. Hence, increasing degree increases regularity.*

2.4.3 Quantitative envelopes for the Bézier curve

Working with the Bézier curve control points in place of the curve itself allows a simpler explicit representation. However, since our framework is not based on the Bézier curve itself, we are interested in the localisation of the Bézier curve with respect to its control points, *i.e.* the control polygon. In this part, we review a result on *sharp quantitative bounds* between the Bézier curve and its control polygon [81, 104]. For instance, in the case of a quadrotor (discussed in Section 2.7.2), once we have selected the control points for the reference trajectory, these envelopes describe the exact localisation of the quadrotor trajectory and its distance from the obstacles. These quantitative envelopes may be of particular interest when avoiding corners of obstacles which traditionally in the literature [117] are modelled as additional constraints or introducing safety margin around the obstacle.

We start by giving the definition for the control polygon.

Definition 2.1 *(Control polygon for Bézier curves (see [104])). Let* $f = \sum_{j=0}^{N} c_j B_{j,N}(t)$ *be a scalar-valued Bézier curve. The control polygon* $\Gamma_f = \sum_{j=0}^{N} c_j H_j(t)$ *of* f *is a piecewise linear function connecting the points with coordinates* (t_j^*, c_j) *for* $j = 0, \ldots, N$ *where the first components* $t_j^* = \frac{j}{N}$ *are the Greville abscissae. The hat functions* H_j *are piecewise linear functions defined as:*

$$H_j(t) = \begin{cases} \frac{t - t_{j-1}^*}{t_j^* - t_{j-1}^*} & t \in [t_{j-1}^*, t_j^*] \\ \frac{t_{j+1}^* - t}{t_{j+1}^* - t_j^*} & t \in [t_j^*, t_{j+1}^*] \\ 0 \quad \textit{otherwise.} \end{cases}$$

An important detail is the *maximal distance* between a Bézier segment and its control polygon. For that purpose, we recall a result from [104], where sharp quantitative bounds of control polygon distance to the Bézier curve are given.

Theorem 2.1 *(See [104], Theorem 3.1) Let $f = \sum_{j=0}^{N} c_j B_{j,N}$ be a scalar Bézier curve and let Γ_f be its control polygon. Then the maximal distance from f to its control polygon is bounded as:*

$$\|f - \Gamma_f\|_{\infty,[0,1]} \leqslant \mu_\infty(N) \, \|\Delta_2 c\|_\infty = D_{\max} \tag{2.18}$$

where the constant $\mu_\infty(N) = \dfrac{\lfloor N/2 \rfloor \lceil N/2 \rceil}{2N}$ [4] *only depends on the degree N and the second difference of the control points $\|\Delta_2 c\|_\infty := \max_{0<j<N} |\Delta_2 c_j|$.*

The j^{th} second difference of the control point sequence c_j for $j = 0, \ldots, N$ is given by:

$$\Delta_2 c_j = c_{j-1} - 2c_j + c_{j+1}.$$

Based on this maximal distance, *Bézier curve's envelopes* are defined as two piecewise linear functions:

- the *lower envelope* $\underline{\Gamma}_f = \sum_{j=0}^{N} \underline{e}_j H_j = \sum_{j=0}^{N} (c_j - D_{\max}) H_j$ and,
- the *upper envelope* $\bar{\Gamma}_f = \sum_{j=0}^{N} \bar{e}_j H_j = \sum_{j=0}^{N} (c_j + D_{\max}) H_j$

such that $\underline{\Gamma}_f \leqslant f \leqslant \bar{\Gamma}_f$.
The envelopes are improved by taking $\underline{e}_0 = \bar{e}_0 = c_0$ and $\underline{e}_N = \bar{e}_N = c_N$ and then *clipped* with the standard Min-Max bounds [5]. The Min-Max bounds yield rectangular envelopes that are defined as

Definition 2.2 *(Min-Max Bounding box (see [116])). Let $f = \sum_{j=0}^{N} c_j B_{j,N}$ be a Bézier curve. As a consequence of the convex-hull property, a min-max bounding box is defined for the Bézier curve f as:*

$$\min_{0<j<N} c_j \leqslant \sum_{j=0}^{N} c_j B_{j,N} \leqslant \max_{0<j<N} c_j.$$

Remark 2.4 *As we notice, the maximal distance between a Bézier segment and its control polygon is bounded in terms of the second difference of the control point sequence and a constant that depends only on the degree of the polynomial. Thus, by elevating the degree of the Bézier control polygon,* i.e. *the subdivision (without modifying the Bézier curve), we can arbitrary reduce the distance between the curve and its control polygon.*

[4] Note that the notation$\lceil x \rceil$ means the ceiling of x, i.e. the smallest integer greater than or equal to x, and the notation$\lfloor x \rfloor$ means the floor of x, i.e. the largest integer less than or equal to x.

[5] Unfortunately the simple Min-Max bounds define very large envelopes when applied solely.

2.4.4 Symbolic Bézier operations

In this section, we present the Bézier operators needed to find the Bézier control points of the states and the inputs. Let the two polynomials $f(t)$ (of degree m) and $g(t)$ (of degree n) with *control points* f_j and g_j be defined as follows:

$$f(t) = \sum_{j=0}^{m} f_j B_{j,m}(t), \quad 0 \leqslant t \leqslant 1$$
$$g(t) = \sum_{j=0}^{n} g_j B_{j,n}(t), \quad 0 \leqslant t \leqslant 1$$

We now show how to determine the control points for the degree elevation and for the arithmetic operations (the sum, difference, and product of these polynomials). For further information on Bézier operations, see [40]. Some illustrations of the geometrical significance of these operations are included in the Appendix 2.A.

Degree elevation: To increase the degree from n to $n+r$ and the number of control points from $n+1$ to $n+r+1$ without changing the shape, the new control points b_j of the $(n+r)$th Bézier curve are given by:

$$b_j = \sum_{i=\max(0,j-r)}^{\min(n,j)} \frac{\binom{n}{i}\binom{r}{j-i}}{\binom{n+r}{j}} g_i \qquad j = 0, 1, \ldots, n+r \tag{2.19}$$

The latter constitutes the so-called *augmented control polygon*. The new control points are obtained as convex combinations of the original control points. This is an important operation exploited in addition/subtraction of two control polygons of different lengths and in approaching the curve to a new control polygon by refining the original one.

Addition and subtraction: If $m = n$ we simply add or subtract the coefficients

$$f(t) \pm g(t) = \sum_{j=0}^{m} (f_j \pm g_j) B_{j,m}(t) \tag{2.20}$$

If $m > n$, we need to first elevate the degree of $g(t)$ $m - n$ times using (2.19) and then add or subtract the coefficients.

Multiplication: Multiplication of two polynomials of degree m and n yields a degree $m + n$ polynomial

$$f(t)g(t) = \sum_{j=0}^{m+n} \underbrace{\left(\sum_{i=\max(0,j-n)}^{\min(m,j)} \frac{\binom{m}{i}\binom{n}{j-i}}{\binom{m+n}{j}} f_i g_{j-i} \right)}_{\text{Control points of the product}} B_{j,m+n}(t) \tag{2.21}$$

2.4.5 Bézier time derivatives

We give the derivative property of the Bézier curve in Proposition 2.1 which is crucial in establishing the constrained trajectory procedure.

Lemma 2.2 *(see [83]) The derivative of the jth Bernstein function of degree $n \geqslant 1$ is given by*

$$DB_{j,N}(t) = N\left(B_{j-1,N-1}(t) - B_{j,N-1}(t)\right) \text{ for } j = 0, \ldots, N. \tag{2.22}$$

for any real number t and where $B_{-1,N-1} = B_{N,N-1} = 0$.

Proposition 2.1 *If the flat output or the reference trajectory y is a Bézier curve, its derivative is still a Bézier curve and we have an explicit expression for its control points.*

Proof 2.1 *Let $y^{(q)}(t)$ denote the qth derivative of the flat output $y(t)$. We use the fixed time interval $T = t_f - t_0$ to define the time as $t = T\tau$, $0 \leqslant \tau \leqslant 1$. We can obtain $y^{(q)}(\tau)$ by computing the qth derivatives of the Bernstein functions.*

$$y^{(q)}(\tau) = \frac{1}{T^q} \sum_{j=0}^{N} c_j B_{j,N}^{(q)}(\tau) \tag{2.23}$$

Letting $c_j^{(0)} = c_j$, we write

$$y(\tau) = y^{(0)}(\tau) = \sum_{j=0}^{N} c_j^{(0)} B_{j,N}(\tau) \tag{2.24}$$

Then,

$$y^{(q)}(\tau) = \sum_{j=0}^{N-q} c_j^{(q)} B_{j,N-q}(\tau) \tag{2.25}$$

with derivative control points such that

$$c_j^{(q)} = \begin{cases} c_j, & q = 0 \\ \dfrac{(N-q+1)}{T^q}\left(c_{j+1}^{(q-1)} - c_j^{(q-1)}\right), & q > 0. \end{cases} \tag{2.26}$$

We can deduce the explicit expressions for all lower order derivatives up to order $N-1$. This means that if the reference trajectory $y_r(t)$ is a Bézier curve of degree $N > q$ (q is the derivation order of the flat output y), by differentiating it, all states and inputs are given in straightforward Bézier form.

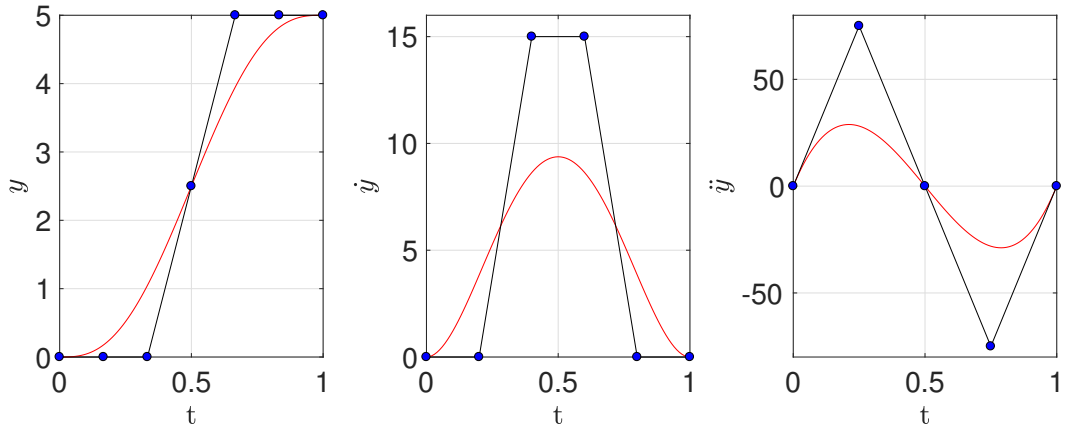


Figure 2.4: The time derivatives when $T = 1$

Example 2.1 *Through a simple example of a double integrator, we want to represent the link between the time interval and the time derivatives. For a changing position y, its time derivative $\dot{y}$ is its velocity, and its second derivative with respect to time $\ddot{y}$, is its acceleration. Even higher derivatives are sometimes also used: the third derivative of position with respect to time is known as the jerk.*

We here want to show the effect of the fixed time period T on the velocity, acceleration, etc. We remark the connection between the time scaling parameter appearing in the trajectory parameterization. We have a simple double integrator defined as:

$$\ddot{y} = u \tag{2.27}$$

As a reference trajectory, we choose a Bézier curve $y = \sum_{i=0}^{N} a_i B_{i,N}$ of order $N = 4$. Due to the Bézier derivative property, we can explicitly provide the link between the time interval T and control points of the Bézier curve's derivatives.

$$\dot{y} = \sum_{i=0}^{N-1} a_i^{(1)} B_{i,N-1} \tag{2.28a}$$

$$\ddot{y} = \sum_{i=0}^{N-2} a_i^{(2)} B_{i,N-2} \tag{2.28b}$$

where $a_i^{(1)}$ and $a_i^{(2)}$ are the control points of the first and the second derivative of the B-spline curve respectively. We have the expressions of the $a_i^{(1)}$ and $a_i^{(2)}$ in terms of the a_i. This fact allow us to survey when the desired reference trajectory will respect the input constraints i.e. $a_i^{(2)} = f_1(a_i^{(1)}) = f_2(a_i)$. That means that if $\forall a_i^{(2)} < K$ then $u < K$.

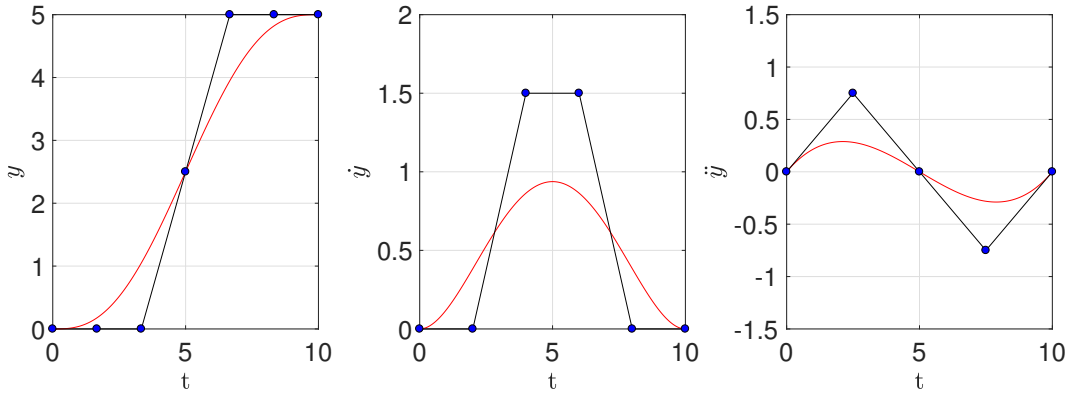


Figure 2.5: The time derivatives when $T = 10$

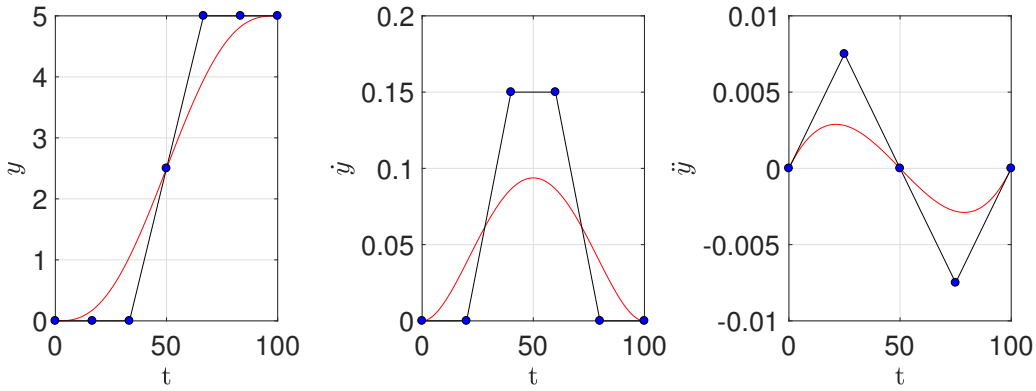


Figure 2.6: The time derivatives when $T = 100$

Proposition 2.2 *If we take a Bézier curve as reference trajectory* $y_r(t) = \sum_{j=0}^{N} c_j B_{j,N}(t)$ *for a flat system such that the input is a polynomial function of the flat output and its derivatives, then the open loop input is also a Bézier curve* $u_r = B(y_r, ..., y_r^{(q)}) = \sum_{i=0}^{m} U_i B_{i,m}(t)$.

Remark 2.5 *We should take a Bézier curve of degree* $N > q$ *to avoid introducing discontinuities in the control input.*

Example 2.2 *In the case of a chain of integrators* $u_r(t) = y_r^{(q)}(t)$ *by imposing for all* $K_l \leqslant c_j^{(q)} \leqslant K_h$, *we ensure an input constraint* $K_l \leqslant u_r(t) \leqslant K_h$.

2.5 Constrained feedforward trajectory procedure

We aim to find a feasible Bézier trajectory (or a set of feasible trajectories, and then make a suitable choice) $\boldsymbol{y_r}(t)$ between the initial conditions $\boldsymbol{y_r}(t_0) = \boldsymbol{y}_{\text{initial}}$ and the final conditions $\boldsymbol{y_r}(t_f) = \boldsymbol{y}_{\text{final}}$. We here show the procedure to obtain the *Bézier control points* for the constrained nominal trajectories $(\boldsymbol{y_r}, \boldsymbol{x_r}, \boldsymbol{u_r})$.

Given a *differentially flat system* $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$, the reference design procedure can be summarized as:

1. Assign to each flat output (trajectory) y_i a *symbolic Bézier curve* $y_r(t) = \sum_{j=0}^{N} \alpha_j B_{j,N}(t)$ of a suitable degree $N > q$ (q is the time derivatives of the flat output) and where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_N) \in \mathbb{R}^{N+1}$ are its control points.

2. Compute the needed derivatives of the flat outputs using Equation (2.25).

3. Use the Bézier operations to produce the system model relationships (2.11)-(2.12), and to find the *state reference Bézier curve* $\boldsymbol{x_r}(t) = \sum_{i=0}^{m} X_i B_{i,m}(t)$ and *input reference Bézier curve* $\boldsymbol{u_r}(t) = \sum_{j=0}^{m} U_j B_{j,m}(t)$ respectively, such that $(X_i, U_j) = r_k(\alpha_0, \ldots, \alpha_N), k = 0, \ldots, m + n + 2)$ are functions of the *output control points*.

4. If needed, calculate the corresponding *augmented control polygons* by elevating the degree of the original control polygons in order to be closer to the Bézier trajectory.

5. Specify the initial conditions, final conditions, or intermediate conditions on the flat output or on any derivative of the flat output that represent a direct equality constraint on the Bézier control points. Each flat output trajectory has its control points fixed as follows:

$$\alpha_0^{(i)} = y^{(i)}(t_0), \tag{2.29a}$$
$$\alpha_N^{(i)} = y^{(i)}(t_f), \text{ for } i = 0, \ldots q, \tag{2.29b}$$
$$\alpha_j \in [\underline{\alpha_j}, \bar{\alpha}_j] \text{ for } j = 1, \ldots N-1, \tag{2.29c}$$

where $\underline{\alpha_j}, \bar{\alpha}_j \in \mathbb{R}$ are the limits of the j^{th} control point. By using the Bézier properties, we will construct a set of constraints by means of its control points. We have a special case for the paralellotope where the first and last control point are fixed $\underline{\alpha_0} = \bar{\alpha}_0 = y(t_0)$ and $\underline{\alpha_N} = \bar{\alpha}_N = y(t_f)$ respectively.

6. We consider a constrained method based on the Bézier control points since the control point polygon captures important geometric properties of the Bézier curve shape. The conditions on the output Bézier control points α_j, the state Bézier control points X_i and the the input control points U_j result in a semi-algebraic set (system of polynomial equations and/or inequalities) defined as:

$$\mathscr{I}(r,\mathbb{A}) = \{\boldsymbol{\alpha} \in \mathbb{A} \mid r_k(\boldsymbol{\alpha}) *_k 0, k \in \{1,\ldots,l\}, *_k \in \{<,\leqslant,>,\geqslant,=,\neq\}\} \tag{2.30}$$

Depending on the studied system, the output constraints can be defined as in equation (2.13), or remain as $\mathbb{A} = \mathbb{R}^{N+1}$.

7. Find the regions of the *control points* α_j, $j = 1,\ldots N-1$, solving the system of equality/inequalities (2.30) by using an appropriate method. We present two kind of possible methods in Section 2.6.

2.6 Feasible control points regions

Once we transform all the system trajectories through the symbolic Bézier flat output, the problem is formulated as a system of functions (equations and inequalities) with Bézier control points as parameters (see equation (2.30)). Consequently the following question raises:

Question 2.2 *How to find the regions in the space of the parameters (Bézier control points) where the system of functions remains valid* i.e. *the constrained set of feasible feed-forwarding trajectories?*

This section has the purpose to answer the latter question by reviewing two methods from *semialgebraic geometry* [6] :

In *the first method*, we formulate the regions for the reference trajectory control points search as a *Quantifier Elimination (QE)* problem. The QE is a powerful procedure to compute an *equivalent* quantifier-free formula for a given first-order formula over the reals [30, 132]. Here we briefly introduce the QE method.
Let $f_i(X,U) \in \mathbb{Q}[X,U], i = 1,\ldots,l$ be polynomials with rational coefficients where:

[6] The theory that studies the real-number solutions to algebraic inequalities with-real number coefficients, and mappings between them, is called semialgebraic geometry.

- $X = (x_1, \ldots, x_n) \in \mathbb{R}^n$ is a vector of quantified variables
- $U = (u_1, \ldots, u_m) \in \mathbb{R}^m$ is a vector of unquantified (free) variables.

The *quantifier-free* Boolean formula $\varphi(X, U)$ is a combined expression of polynomial equations ($f_i(X, U) = 0$) , inequalities ($f_i(X, U) \leq 0$), inequations ($f_i(X, U) \neq 0$) and strict inequalities ($f_i(X, U) > 0$) that employs the logic operators $\wedge$ (and), $\vee$ (or), $\Rightarrow$ (implies) or $\Leftrightarrow$ (equivalence).
A prenex or *first-order formula* is defined as follows:

$$G(X, U) = (Q_1 x_1) \ldots (Q_n x_n)[\varphi(X, U)]$$

where Q_i is one of the quantifiers $\forall$(for all) and $\exists$ (there exists). Following the Tarski Seidenberg theorem (see [30]), for every prenex formula $G(X, U)$ there exists an equivalent quantifier-free formula $\psi(U)$ defined by the free variables.

The goal of the QE procedure is to compute an equivalent quantifier free formula $\psi(U)$ for a given first-order formula. It finds the feasible regions of free variables U represented as semialgebraic set where $G(X, U)$ is true. If the set U is non-empty, there exists a point $u \in \mathbb{R}^m$ which simultaneously satisfies all of the equations/inequalities. Such a point is called a feasible point and the set U is then called feasible. If the set U is empty, it is called unfeasible. In the case when $m = 0$, *i.e.* when all variables are quantified, the QE procedure decides whether the given formula is true or false (decision problem). For instance,

- given a first order formula $\forall x\,[x^2 + bx + c > 0]$, the QE algorithm gives the equivalent quantifier free formula $b - 4c < 0$;
- given a first order formula $\exists x\,[ax^2 + bx + c = 0]$, the QE algorithm gives the equivalent quantifier free formula $(a \neq 0 \wedge b^2 - 4ac \geq 0) \vee (a = 0 \wedge b \neq 0) \vee (a = 0 \wedge b = 0 \wedge c = 0)$.

As we can notice, the quantifier free formulas represent the semi-algebraic sets (the conditions) for the unquantified free variables verifying the first order formula is *true*. Moreover, given an input formula without quantifiers, the QE algorithm produces a *simplified formula*. For instance (for more examples, see [20]),

- given an input formula $(ab \leqslant 0) \wedge (a + b = 0) \wedge (b^2 + a^2 > 0) \vee (a^2 = -b^2)$, the QE algorithm gives the equivalent simplified formula $a + b = 0$.

On the other hand, given an input formula without unquantified free variables (usually called closed formula) is either *true* or *false*.

The symbolic computation of the *Cylindrical Algebraic Decomposition* (CAD) introduced by Collins [27] is the best currently known QE algorithm for solving real algebraic constraints (in particular parametric and *non-convex* case) (see [130]). This method gives us *an exact solution*, a simplified formula describing the semi-algebraic set.

The QE methods, particularly the CAD, have already been used in various aspects of control theory (see [3, 118] and the references therein): robust control design, finding the feasible regions of a PID controller, the Hurwitz and Schur stability regions, reachability analysis of nonlinear systems, trajectory generation [78].

Remark 2.6 *(On the complexity) Unfortunately the above method rapidly becomes slow due to its double exponential complexity [84]. Its efficiency strongly depends on the number and on the complexity of the variables (control points) used for a given problem. The computational complexity of the CAD is double exponential* i.e. *bounded by* $(sd)^{2^{\mathcal{O}(n)}}$ *for a finite set of* s *polynomials in* n *variables, of degree* d*. There are more computationally efficient QE methods than the CAD, like the Critical Point Method [11] (it has single exponential complexity in* n *the number of variables) and the cylindrical algebraic sub-decompositions [140] but to the author knowledge there are no available implementations.*

For more complex systems, the exact or symbolic methods are too computationally expensive. There exist methods that are numerical rather than exact.

As a second alternative method, we review one such method based on approximation of the exact set with more reasonable computational cost. The second method known as the Polynomial Superlevel Set (PSS) method, based on the paper [31] instead of giving us exact solutions tries to approximate the set of solutions by minimizing the L^1 norm of the polynomial. It can deal with more complex problems.

2.6.1 Cylindrical Algebraic Decomposition

In this section, we give a simple introduction to the Cylindrical Algebraic Decomposition.

Input of CAD: As an input of the CAD algorithm, we define a set of polynomial equations and/or inequations in n unknown symbolic variables (in our case, the control points) defined over real interval domains.

Definition of the CAD: The idea is to develop a sequence of projections that drops the dimension of the semi-algebraic set by one each time. Given a set S of polynomials in R^n, a *cylindrical algebraic decomposition* is a decomposition of R^n into finitely many connected semialgebraic sets called *cells*, on which each polynomial has constant sign, either $+$, $-$ or 0. To be cylindrical, this decomposition must satisfy the following condition: If $1 \leqslant k < n$ and π is the projection from R^n onto R^{n-k} consisting in removing the k last coordinates, then for every pair of cells c and d, one has either $\pi(c) = \pi(d)$ or $\pi(c) \cap \pi(d) = \emptyset$. This implies that the images by π of the cells define a cylindrical decomposition of R^{n-k}.

Output of CAD: As an output of this symbolic method, we obtain the total algebraic expressions that represent *an equivalent simpler form* of our system. Ideally, we would like to obtain a parametrization of all the control points regions as a closed form solution. Finally, in the case where closed forms are computable for the solution of a problem, one advantage is to be able to overcome any optimization algorithm to solve the problem for a set of given parameters (numerical values), since only an evaluation of the closed form is then necessary.

The execution runtime and memory requirements of this method depend of the dimension of the problem to be solved because of the computational complexity. For the implementation part, we will use its *Mathematica* implementation[7] (developed by Adam Strzebonski). Other implementations of CAD are QEPCAD, Redlog, SyNRAC, Maple.

Example 2.3 *From [74], we present an example in which we want to find the regions of the parameters $(a, b) \in \mathbb{R}^2$ where the following formula is true, not only answering if the formula is true or not.*
Having as input

$$F = \left\{(a,b) \in \mathbb{R}^2 : f_1(a,b) = \sqrt{a^2 - b^2} + \sqrt{ab - b^2} - a > 0, \quad f_2(a,b) = 0 < b < a\right\}$$

[7] see `https://reference.wolfram.com/language/ref/CylindricalDecomposition.html`

the corresponding CAD output is given by

$$\left\{a > 0 \wedge b < \frac{4}{5}a\right\}$$

As we notice, given a system of equations and inequalities formed by the control points relationship as an input, the CAD returns a simpler system that is equivalent over the reals.

2.6.2 Approximations of Semialgebraic Sets

Here we present a method based on the paper [31] that tries to approximate the set of solutions. Given a set

$$\mathcal{K} = \{x \in \mathbb{R}^n : g_i(x) \geqslant 0, i = 1, 2, \ldots, m\}$$

which is compact, with non-empty interior and described by given real multivariable polynomials $g_i(x)$ and a compact set $\mathcal{B} \supset \mathcal{K}$, we aim at determining a so-called *polynomial superlevel set* (PSS)

$$U(p) = \{x \in \mathcal{B} : p(x) \geqslant 1\}$$

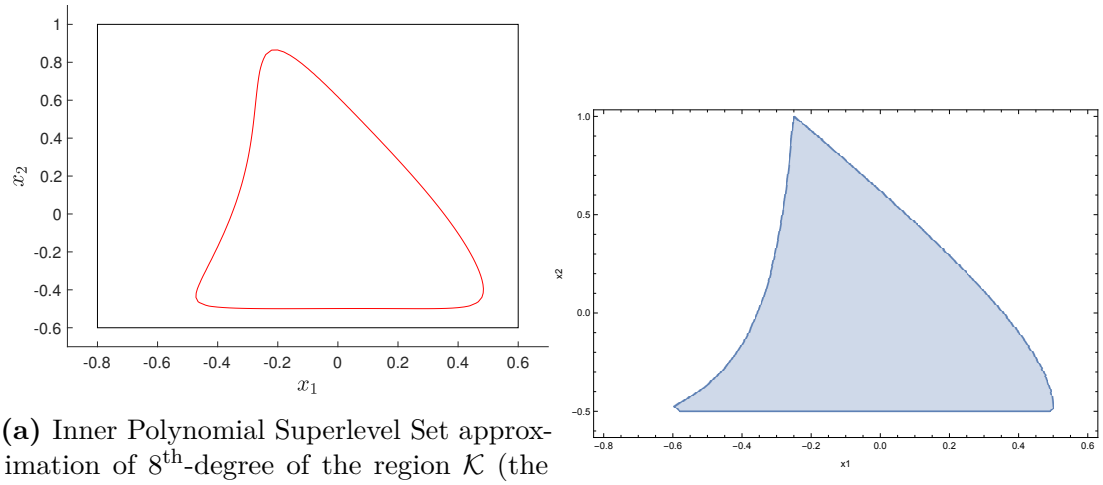
The set $\mathcal{B}$ is assumed to be an n-dimensional hyperrectangle. The PSS can capture the main characteristics of $\mathcal{K}$ (it can be non convex and non connected) while having at the same time a simpler description than the original set. It consists in finding a polynomial p of degree d whose 1-superlevel set $\{x \mid p(x) \geqslant 1\}$ contains a semialgebraic set $\mathcal{B}$ and has minimum volume. Assuming that one is given a simple set $\mathcal{B}$ containing $\mathcal{K}$ and over which the integrals of polynomials can be efficiently computed, this method involves searching for a polynomial p of degree d which minimizes $\int_{\mathcal{B}} p(x)dx$ while respecting the constraints $p(x) \geqslant 1$ on $\mathcal{K}$ and $p(x) \geqslant 0$ on $\mathcal{B}$. Note that the objective is linear in the coefficients of p and that these last two nonnegativity conditions can be made computationally tractable by using the *sum of squares relaxation*. The complexity of the approximation depends on the degree d. The advantage of such a formulation lies in the fact that when the *degree* of the polynomial p increases, the objective value of the problem converges to the true volume of the set $\mathcal{K}$.

Example 2.4 *To better review the latter method, we illustrate it with an example for a two dimensional set given in [31]. In order to compare the two presented*

methods, we also give its CAD solution. Having the following non-convex semi-algebraic set:

$$\mathcal{K} = x \in \mathbb{R}^2 : \begin{cases} f_1(x) = 1 + 2x_2 \geqslant 0, \\ f_1(x) = 2 - 4x_1 - 3x_2 \geqslant 0, \\ f_1(x) = 10 - 28x_1 - 5x_2 - 24x_1x_2 - 18x_2^2 \geqslant 0, \\ f_1(x) = 1 - x_2 - 8x_1^2 - 2x_1x_2 - x_2^2 - 8x_1^2x_2 - 6x_1x_2^2 \geqslant 0 \end{cases}$$

with a bounding box $\mathcal{B} = [-0.8, 0.6] \times [-0.6, 1.0]$*, and setting* $d = 8$ *, the degree of the polynomial* $p(x)$*. The algorithm yields the feasible region represented in Figure 2.7a.*



(a) Inner Polynomial Superlevel Set approximation of 8^{th}-degree of the region $\mathcal{K}$ (the inner surface of the red line).The black rectangle represents the bounding box.

(b) The region found by the CAD algorithm (the inner surface of the blue line).

Figure 2.7: The feasible regions by the two methods

For the same set, even without specifying a particular box, the CAD algoririthm finds the following explicit solution:

$$\left(x_1 = -\frac{5}{8} \wedge x_2 = -\frac{1}{2}\right)$$
$$\vee \left(-\frac{5}{8} < x_1 < -\frac{1}{6} \wedge -\frac{1}{2} \leqslant x_2 \leqslant \frac{-8x_1^2 - 2x_1 - 1}{2(6x_1 + 1)} - \frac{1}{2}\sqrt{\frac{64x_1^4 - 160x_1^3 - 12x_1^2 + 28x_1 + 5}{(6x_1 + 1)^2}}\right)$$
$$\vee \left(x_1 = -\frac{1}{6} \wedge -\frac{1}{2} \leqslant x_2 \leqslant \frac{7}{8}\right)$$
$$\vee \left(-\frac{1}{6} < x_1 < \frac{1}{2} \wedge -\frac{1}{2} \leqslant x_2 \leqslant \frac{-8x_1^2 - 2x_1 - 1}{2(6x_1 + 1)} + \frac{1}{2}\sqrt{\frac{64x_1^4 - 160x_1^3 - 12x_1^2 + 28x_1 + 5}{(6x_1 + 1)^2}}\right)$$
$$\vee \left(x_1 = \frac{1}{2} \wedge x_2 = -\frac{1}{2}\right)$$

As we can observe, the PSS method (Figure 2.7a) gives us a good approximation of the feasible region, almost the same as the exact one obtained by the CAD algorithm (Figure 2.7b). However, in some cases, we observed that the PSS method may have some sensibilities when its bounding box is not well defined.

2.7 Applications

2.7.1 Longitudinal dynamics of a vehicle

The constraints are essentials in the design of vehicle longitudinal control which aims to ensure the passenger comfort, safety and fuel/energy reduction. The longitudinal control can be designed for a highway scenario or a city scenario. In the first scenario, the vehicle velocity keeps a constant form where the main objective is the vehicle inter-distance while the second one, deals with frequent stops and accelerations, the so-called Stop-and-Go scenario [135]. The inter-distance dynamics can be represented as an single integrator driven by the difference between the leader vehicle velocity V_l and the follower vehicle velocity V_x , i.e., $\dot{d} = V_l - V_x$.
In this example, suppose we want to follow the leader vehicle, and stay within a fixed distance from it (measuring the distance through a camera/radar system). Additionally, suppose we enter a desired destination through a GPS system, and suppose our GPS map contains all the speed information limits. Our goal is the follower longitudinal speed V_x to follow a reference speed $V_{xr}(t) \in [0, \min(V_l, V_{\max})], V_{\max} \in \mathbb{R} > 0$ given by the minimum between the leader vehicle speed and the speed limit.

The longitudinal dynamics of a follower vehicle is given by the following model:

$$M\dot{V}_x(t) = \frac{u(t)}{r} - C_a V_x^2(t) \tag{2.31}$$

where V_x is the longitudinal speed of the vehicle, u is the motor torque, taken as *control input* and the physical constants: M the vehicle's mass, r the mean wheel radius, and C_a the aerodynamic coefficient.
The model is differentially flat, with V_x as a flat output. An open loop control yielding the tracking of the reference trajectory V_{xr} by V_x, assuming the model to be perfect, is

$$u_r(t) = r\left(M\dot{V}_{xr}(t) + C_a V_{xr}^2(t)\right) \tag{2.32}$$

If we desire an open-loop trajectory $u_r \in C^0$, then for the flat output, we should assign a Bézier curve of degree $d > 1$. We take V_{xr} as reference trajectory, a Bézier curve of degree 4 *i.e.* C^4-function.

$$V_{xr}(t) = \sum_{i=0}^{4} a_i B_{i,4}(t),$$
$$V_{xr}(t_0) = V_i, \quad V_{xr}(t_f) = V_f$$

where the a_i's are the control points and the $B_{i,4}$ the Bernstein polynomials. Using the Bézier curve properties, we can find the control points of the open-loop control u_r in terms of the a_i's by the following steps:

1. First, we find the control points $a_i^{(1)}$ for $\dot{V}_{xr}$ by using the Equation (3.10):

$$\dot{V}_{xr} = \sum_{i=0}^{3} a_i^{(1)} B_{i,3}(t)$$

2. We obtain the term V_{xr}^2 by

$$V_{xr}^2 = \sum_{i=0}^{4} a_i B_{i,4}(t) \sum_{i=0}^{4} a_i B_{i,4}(t) = \sum_{i=0}^{8} p_i B_{i,8}(t)$$

   which is a Bézier curve of degree 8 and where the control points p_i are computed by the multiplication operation (see Equation (2.21)).

3. We elevate the degree of the first term up to 8 by using the Equation (2.19) and then, we find the sum of the latter with the Bézier curve for V_{xr}^2. We end up with u_r as a Bézier curve of degree 8 with nine control points U_i:

$$u_r(t) = rM\dot{V}_{xr} + rC_a V_{xr}^2 = rM \sum_{i=0}^{3} a_i B_{i,3}(t) + rC_a (\sum_{i=0}^{4} a_i B_{i,4})^2 = \sum_{i=0}^{8} U_i B_{i,8}(t)$$

   with $U_i = r_k(a_0, \ldots, a_4)$.

Symbolic input constraints

We want the input control points U_i to be

$$U_{\min} < U_i < U_{\max} \quad i = 0, \ldots, 8 \tag{2.33}$$

where $U_{\min} = 0$ is the lower input constraint and $U_{\max} = 10$ is the high input constraint. By limiting the control input, we indirectly constraint the fuel consumption. The initial and final trajectory control points are defined as $V_x(t_0) = a_0 = 0$ and $V_x(t_1) = a_4 = 1$ respectively.

The constraint (2.33) directly corresponds to the semi-algebraic set: The constraint (2.33) corresponds to the *semi-algebraic set i.e.* the following system

of nonlinear inequalities:

$$\begin{cases} 0 < U_0 = 4\,a_1 < 10 \\ 0 < U_1 = a_1 + \frac{3\,a_2}{2} < 10 \\ 0 < U_2 = \frac{4\,a_1^2}{7} - \frac{5\,a_1}{7} + \frac{12\,a_2}{7} + \frac{3\,a_3}{7} < 10 \\ 0 < U_3 = \frac{15\,a_2}{14} - \frac{10\,a_1}{7} + a_3 + \frac{6\,a_1\,a_2}{7} + \frac{1}{14} < 10 \\ 0 < U_4 = \frac{18\,a_2^2}{35} - \frac{10\,a_1}{7} + \frac{10\,a_3}{7} + \frac{16\,a_1\,a_3}{35} + \frac{2}{7} < 10 \\ 0 < U_5 = \frac{10\,a_3}{7} - \frac{15\,a_2}{14} - \frac{6\,a_1}{7} + \frac{6\,a_2\,a_3}{7} + \frac{5}{7} < 10 \\ 0 < U_6 = \frac{4\,a_3^2}{7} + \frac{5\,a_3}{7} - \frac{3\,a_1}{7} - \frac{9\,a_2}{7} + \frac{10}{7} < 10 \\ 0 < U_7 = \frac{5}{2} - \frac{3\,a_2}{2} < 10 \\ 0 < U_8 = 5 - 4\,a_3 < 10 \end{cases} \tag{2.34}$$

In order to solve symbolically the system of inequalities *i.e.* to find the regions of the intermediate control points a_i, we use the Mathematica function *CylidricalDecomposition*. The complete symbolic solution with three intemediate control points (a_1, a_2, a_3) is too long to be included. Since the latter is too long to be included, we illustrate the symbolic solution in the case of two intermediate control points (a_1, a_2) :

$$\begin{aligned} &(0 < a_1 \leqslant 0.115563 \wedge -a_1 < a_2 < 1.33333) \\ &\vee \left(0.115563 < a_1 \leqslant 0.376808 \wedge 0.142857\left(-3.a_1^2 + 2.a_1 - 1\right) < a_2 < 1.33333\right) \\ &\vee \left(0.376808 < a_1 \leqslant 1.52983 \wedge \frac{4a_1 - 2}{3.a_1 + 4} < a_2 < 1.33333\right) \\ &\vee \left(1.52983 < a_1 < 2 \wedge 0.333333\sqrt{15.a_1 - 17} - 0.333333 < a_2 < 1.33333\right) \end{aligned}$$

The latter solution describing the feasible set of trajectories can be used to make a choice for the Bézier control points: "First choose a_1 in the interval $(0, 0.115563]$ and then you may choose a_2 bigger than the chosen $-a_1$ and smaller than 1.33333. Or otherwise choose a_1 in the interval $(0.115563, 0.376808]$ and, then choose a_2 such that $0.142857\,(-3a_1^2 + 2.a_1 - 1) < a_2 < 1.33333$, etc."

In Figure 2.8, we illustrate the feasible regions for the three intermediate control points (a_1, a_2, a_3) by using the Mathematica function *RegionPlot3D*. We can observe how the flat outputs influences the control input *i.e.* which part of the reference trajectory influences which part of the control input. For instance in (2.34), we observe that the second control point a_1 influences more than a_2 and a_3 the beginning of the control input (the control points U_0, U_1, U_2). The previous inequalities can be used as a prior study to the sensibility of the control inputs with respect to the flat outputs.

It should be stressed that the goal here is quite different than the traditional one in optimisation problems. We do not search for the best trajectory according

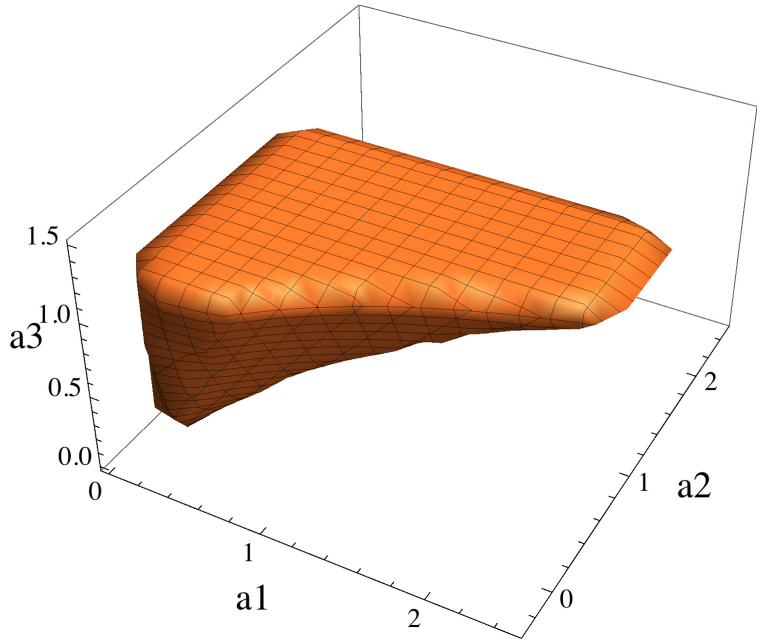


Figure 2.8: Feasible region for the control points of V_{xr} when $U_{\min} = 0$ and $U_{\max} = 10$.

to a certain criterion under the some constraints, but we wish to obtain the set of all trajectories fulfilling the constraints; this for an end user to be able to pick one or another trajectory in the set and to switch from one to another in the same set. The picking and switching operations aim to be really fast.

Simulation results

The proposed control approach has been successfully tested in simulation. For the physical parameters of the vehicle, academic values are chosen to test the constraint fulfilment. For the design of the Bézier reference trajectory, we pick values for a_1, a_2 and a_3 in the constrained region. As trajectory control points for V_{xr}, we take the possible feasible choice $a_0 = 0, a_1 = 2, a_2 = 2.3, a_3 = 1.3, a_4 = 1$. Simulation results for the constrained open-loop input are shown in Figure 2.9.

The form of the closed-loop input is

$$u = Mr\left(\dot{V}_{xr} - \lambda(V_x - V_{xr})\right) + rC_aV_x^2 \tag{2.35}$$

where $\lambda = 9$ is the proportional feedback gain chosen to make the error dynamics stable. Figure 2.10 shows the performance of the closed-loop control. For both schemes, the input respects the limits.

As shown in Figure 2.11, choosing a control point outside of the suitable region ($a_1 = 5.5$) can violate the closed-loop input limits.

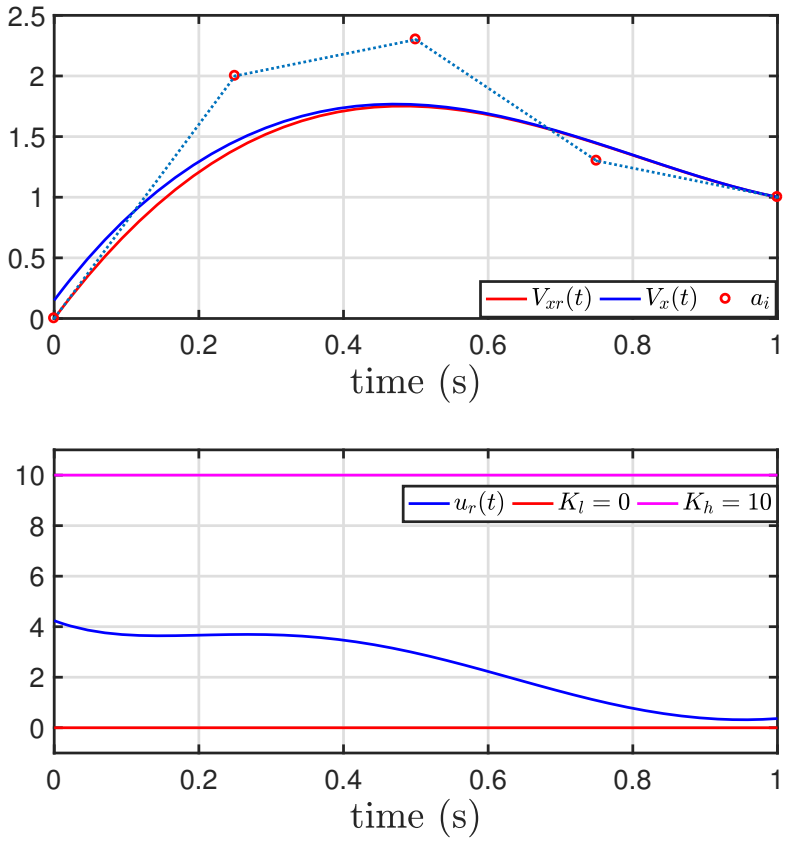


Figure 2.9: Open-loop input control

2.7.2 Quadrotor dynamics

Motivation

Over the last decade, the quadrotors have been a subject of extensive research study and have been used in a wide range of industrial and commercial applications. The quadrotors have become so popular due to their agility that allows them to hover as well as takeoff and land vertically while still being able to perform *agressive trajectories* [8].

However, during aggressive trajectory design, it is difficult to ensure trajectory feasibility while trying to exploit the entire range of feasible motor inputs. Moreover, in many applications, their role is to fly in complex cluttered environments, hence there is a necessity of output constraints. Therefore, the constraints on the inputs and states are one of the crucial issues in the control of quadrotors.

[8] A trajectory is considered as an aggressive one if during its tracking, one of the quadrotor motors is close to a saturation.

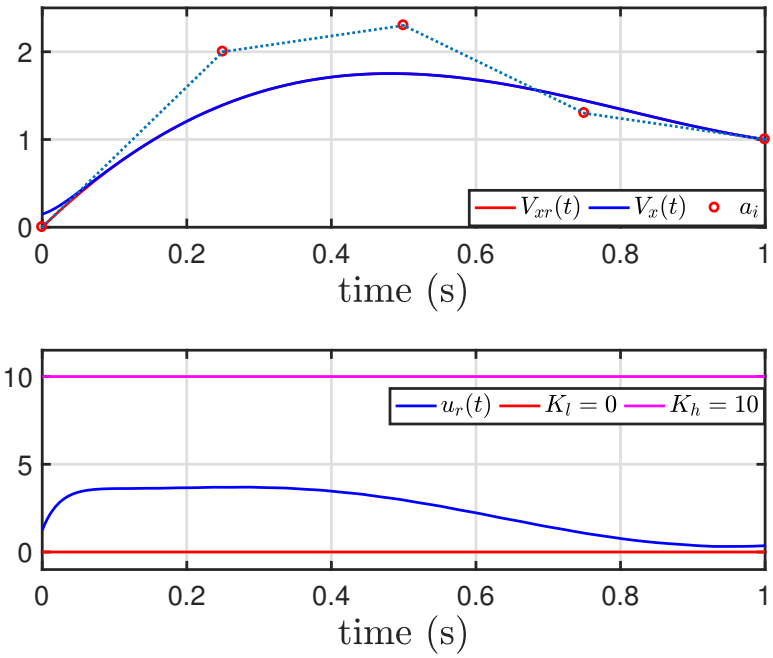


Figure 2.10: Closed-loop performance of trajectory tracking

Fortunately, with the hardware progress, today the quadrotors have speed limits of forty meters per second and more comparing to few meters per second in the past [41]. Therefore, it is important to conceive control laws for quadrotors to a level where they can exploit their full potential especially in terms of agility.

In the famous paper [91], is proposed an algorithm that generates optimal trajectories such that they minimize cost functionals that are derived from the square of the norm of the snap (the fourth derivative of position). There is a limited research investigating the quadrotor constraints (see [21] and the papers therein) without employing an online optimisation.

The following application on quadrotor is devoted to unify the dynamics constraints or demands constraints with the environmental constraints (*e.g.* , fixed obstacles).

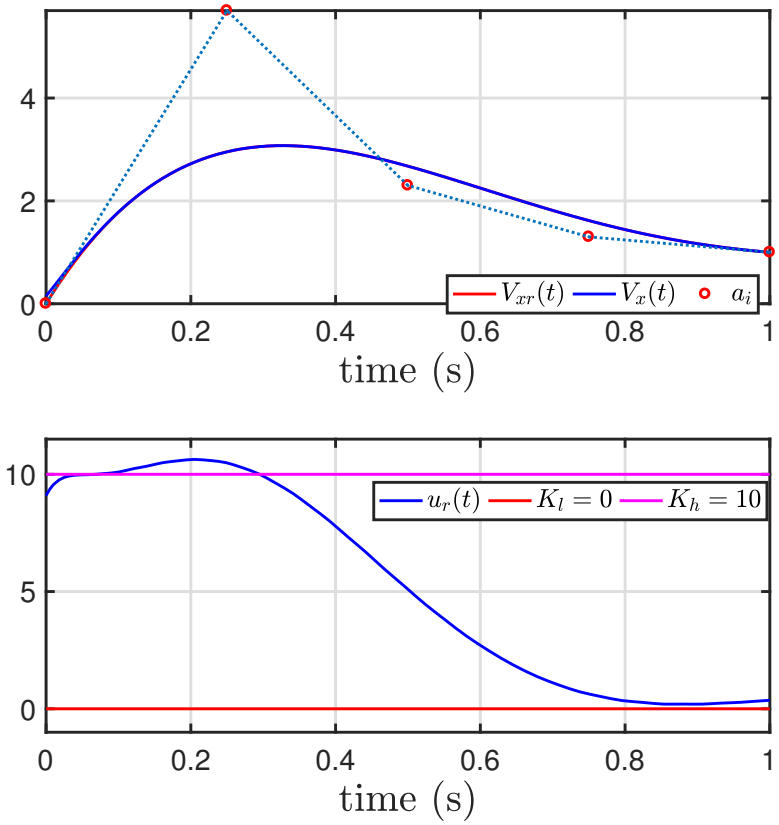


Figure 2.11: When control point a_1 is out of the its region

Simplified model of quadrotor

A (highly) simplified nonlinear model of quadrotor is given by the equations:

$$m\ddot{x} = \theta u_1 \tag{2.36a}$$
$$m\ddot{y} = -\phi u_1 \tag{2.36b}$$
$$m\ddot{z} = -mg + u_1 \tag{2.36c}$$
$$I_x\ddot{\theta} = u_2 \tag{2.36d}$$
$$I_y\ddot{\phi} = u_3 \tag{2.36e}$$
$$I_z\ddot{\psi} = u_4 \tag{2.36f}$$

where x, y and z are the position coordinates of the quadrotor in the world frame, and θ, ϕ and ψ are the pitch, roll and yaw rotation angles respectively. The constant m is the mass, g is the gravitation acceleration and I_x, I_y, I_z are the moments of inertia along the y, x directions respectively. The thrust u_1 is the total lift

generated by the four propellers applied in the z direction, and u_2, u_3 and u_4 are the torques in θ, ϕ and ψ directions respectively. As we can notice, the quadrotor is an under-actuated system *i.e.* it has six degrees of freedom but only four inputs.

A more complete presentation of the quadrotor model can be found in the Section 4.

Differential flatness of the quadrotor

Here, we describe the quadrotor differential parametrization on which its offline reference trajectory planning procedure is based. The model (2.36) is differentially flat. Having four inputs for the quadrotor system, the flat output has four components. These are given by the vector:

$$F = (x, y, z, \psi).$$

By equation (2.36c), we easily obtain expression of the thrust reference u_{1r}

$$u_{1r} = m(\ddot{z}_r + g) \tag{2.37}$$

Then, by replacing the thrust expression in (2.36a)–(2.36b), we obtain the angles θ_r and ϕ_r given by

$$\theta_r = \frac{m\ddot{x}_r}{u_{1r}} = \frac{\ddot{x}_r}{\ddot{z}_r + g} \tag{2.38a}$$

$$\phi_r = \frac{-m\ddot{y}_r}{u_{1r}} = \frac{-\ddot{y}_r}{\ddot{z}_r + g} \tag{2.38b}$$

We then differentiate (2.38a), (2.38b) and ψ_r twice to obtain (2.36d)–(2.36f) respectively. This operation gives us u_2 , u_3 and u_4.

$$u_{2r} = I_x\ddot{\theta}_r = \frac{I_x}{(g + \ddot{z}_r)} \left(x_r^{(4)} - 2\,\frac{x_r^{(3)}(\ddot{z}_r + g) - \ddot{x}_r z_r^{(3)})}{(\ddot{z}_r + g)^2}\, z_r^{(3)} - \frac{\ddot{x}_r z_r^{(4)}}{\ddot{z}_r + g} \right), \tag{2.39}$$

$$u_{3r} = I_y\ddot{\phi}_r = \frac{I_y}{(g + \ddot{z}_r)} \left(-y_r^{(4)} + 2\,\frac{y_r^{(3)}(\ddot{z}_r + g) - \ddot{y}_r z_r^{(3)})}{(\ddot{z}_r + g)^2}\, z_r^{(3)} + \frac{\ddot{y}_r z_r^{(4)}}{\ddot{z}_r + g} \right), \tag{2.40}$$

and

$$u_{4r} = I_z\ddot{\psi}_r. \tag{2.41}$$

A more complete model of a quadrotor and its flatness parametrization can be found in [125] and [57].

Constraints

Given an initial position and yaw angle and a goal position and yaw angle of the quadrotor, we want to find a set of smooth reference trajectories while respecting the dynamics constraints and the environmental constraints. Quadrotors have electric DC rotors that have limits in their rotational speeds, so input constraints are vital to avoid rotor damage. Besides the state and input constraints, to enable them to operate in constrained spaces, it is of great importance to impose output constraints.

We consider the following constraints:

1. *The thrust* u_1
   We set a maximum ascent or descending acceleration of 4g (g=9.8 m/s^2), and hence the thrust constraint is defined as:

$$0 < u_1 \leqslant U_1^{\max} = 4\, m \cdot g = 20.79\,\mathrm{N}, \tag{2.42}$$

   where m is the quadrotor mass which is set as 0.53 kg in the simulation. By the latter constraint, we also avoid the singularity for a zero thrust.

2. *The pitch and roll angle*
   In applications, the tilt angle is usually inferior to 14 degrees (0.25rad). We set

$$|\phi| \leqslant \Phi^{\max} = 0.25\mathrm{rad} \tag{2.43}$$

$$|\theta| \leqslant \Theta^{\max} = 0.25\mathrm{rad} \tag{2.44}$$

3. *The torques* u_2*,* u_3 *et* u_4
   With a maximum tilt acceleration of 48 rad/s^2, the limits of the control inputs are:

$$|u_2|,\ |u_3| \leqslant 48 I_{xx} = 0.3\,\mathrm{N{\cdot}m} \tag{2.45}$$

$$|u_4| \leqslant 48 I_{zz} = 0.5\,\mathrm{N{\cdot}m} \tag{2.46}$$

   where I_{xx}, I_{yy}, I_{zz} are the parameters of the moment of inertia, $I_{xx} = I_{yy} = 6.22 \times 10^{-3}\mathrm{kg \cdot m^2}$, $I_{zz} = 1.12 \times 10^{-2}\mathrm{kg \cdot m^2}$.

4. *Collision-free constraint*
   To avoid obstacles, constraints on the output trajectory x, y, z should be reconsidered.

Scenario 1: In this scenario, we want to impose constraints on the thrust, and on the roll and pitch angles.

Constrained open-loop trajectory u_{1r}

We specialize the flat output z_r to a sigmoid between two quasi constant altitudes, a situation frequently needed in practice:

$$z_r(t) = \frac{H_f - H_i}{2}\left(1 + \tanh(\gamma(t - t_m))\right) + H_i \tag{2.47}$$

where H_i is the initial altitude and H_f is the final altitude of the quadrotor; γ is the slope parameter of the tanh and t_m is the time when the quadrotor is taking off (see Figure 2.12). The maximum value for $z_r(t)$ is the final altitude H_f (see fig. 2.12).

The easy numerical implementation of the derivatives of $z_r(t)$ is due to the nice recursion. Let $R = \tanh(\gamma(t - t_m))$ and $C = \frac{H_f - H_i}{2}$. The first four derivatives of $z_r(t)$ are given as:

$$\begin{aligned}
\dot{z}_r &= \gamma C(1 - R^2) \\
\ddot{z}_r &= -2\gamma^2 CR(1 - R^2) \\
z_r^{(3)} &= 2\gamma^3 C(1 - R^2)(1 - 3R^2) \\
z^{(4)} &= -8\gamma^4 CR(3R^4 - 5R^2 + 2)
\end{aligned}$$

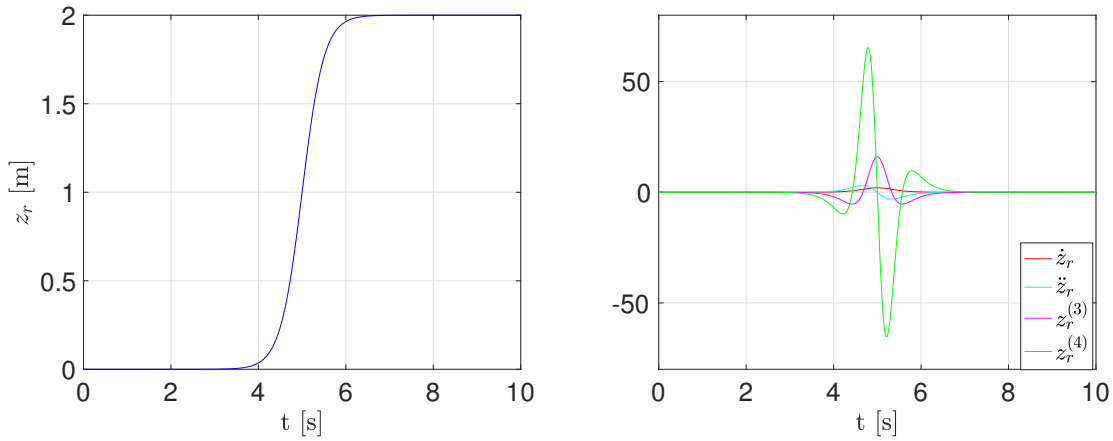


Figure 2.12: The reference trajectory for $z_r(t)$ (left) and its derivatives (right) with $H_i = 0$m and $H_f = 2$m, $t_m = 5s$ and parameter $\gamma = 2$.

The maximum values for its derivatives depend only on γ and C, and their values can be determined. We obtain their bounds as:

$$\begin{aligned} H_i \leqslant z_r &\leqslant H_f, & \\ 0 \leqslant \dot{z}_r &\leqslant b_1\gamma C, & b_1 &= 1; \\ -b_2\gamma^2 C \leqslant \ddot{z}_r &\leqslant b_2\gamma^2 C, & b_2 &= \frac{4\sqrt{3}}{9}; \\ -\underline{b_3}\gamma^3 C \leqslant z_r^{(3)} &\leqslant \overline{b_3}\gamma^3 C, & \underline{b_3} &= \frac{2}{3}, \quad \overline{b_3} = 2; \\ -b_4\gamma^4 C \leqslant z^{(4)} &\leqslant b_4\gamma^4 C, & b_4 &\approx 4.0849. \end{aligned}$$

Consequently, from the thrust limits (2.42), we have the following inequality

$$0 < m(-b_2\gamma^2 + g) \leqslant u_{1r} = m(\ddot{z}_r + g) \leqslant m(b_2\gamma^2 + g) < U_1^{\max}.$$

The input constraint of u_{1r} will be respected by choosing a suitable value of γ and C such that

$$\gamma^2 C < \min\left\{\frac{1}{b_2}\left(\frac{U_1^{\max}}{m} - g\right), \frac{g}{b_2}\right\}. \tag{2.48}$$

Figure 2.13 depicts the constrained open-loop trajectory u_{1r} that is well chosen by taking $\gamma = 2$ and $H_f = 2$m. On the other hand, in Figure 2.14 is shown the violation of the thrust constraints when $\gamma = 7$ is chosen out of the constrained interval (2.48).

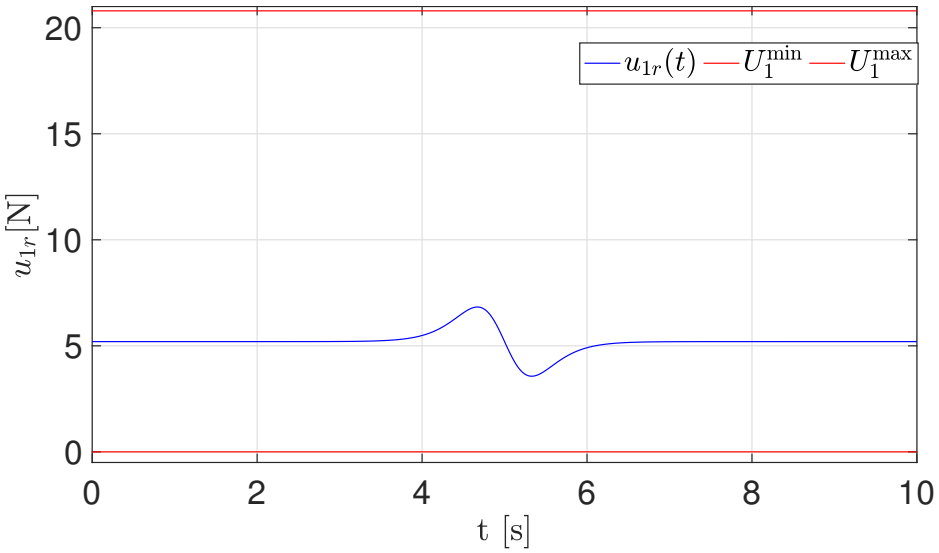


Figure 2.13: The reference trajectory for $u_{1r}(t)$ for a value of $\gamma = 2$ and $H_f = 2$m.

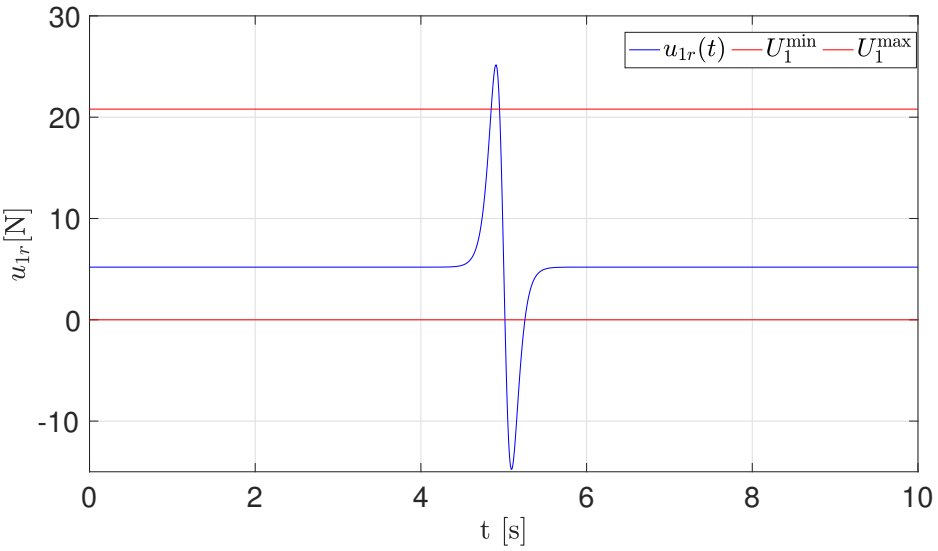


Figure 2.14: When the value for γ is out of the defined interval, the constraints on the open-loop trajectory $u_{1r}(t)$ are not respected. The reference trajectory for $u_{1r}(t)$ for a value of $\gamma = 7$.

Constrained open-loop trajectories θ_r et ϕ_r

In the rest of the study, we omit the procedure for the angle ϕ_r since is the same as for the angle θ_r.

1. In the first attempt, the reference trajectory x_r will be a Bézier curve of degree $d = 6$ with a predefined control polygon form as:

$$\boldsymbol{A_x} = \left\{ a, a, a, \frac{a+b}{2}, b, b, b \right\}.$$

The aim of the first and the final control point repetitions is to fix the velocity and acceleration reference equilibrium points as : $\dot{x}_r(t_0) = \dot{x}_r(t_f) = 0$ and $\ddot{x}_r(t_0) = \ddot{x}_r(t_f) = 0$.

The control polygon of the velocity reference trajectory $\dot{x}$ is :

$$\boldsymbol{A_{\dot{x}}} = \left\{ 0, 0, \frac{d}{T}\frac{b-a}{2}, \frac{d}{T}\frac{b-a}{2}, 0, 0 \right\}.$$

The control polygon of the acceleration reference trajectory $\ddot{x}$ is :

$$\boldsymbol{A_{\ddot{x}}} = \left\{ 0, \frac{d(d-1)}{T^2}\frac{a+b}{2}, 0, -\frac{d(d-1)}{T^2}\frac{a+b}{2}, 0 \right\}.$$

The proposed form of Bézier curve provide us the explicit bounds of its second derivative $\ddot{x}_r$ when $a = 0$ such that $\ddot{x}_r^{\min} = -\frac{144}{25}\frac{b}{T^2}$ and $\ddot{x}_r^{\max} = \frac{144}{25}\frac{b}{T^2}$.

From the Equations (2.43) and (2.38a), we get

$$\frac{-\frac{144}{25}\frac{b}{T^2}}{b_2\gamma^2 C + g} \leqslant \theta_r = \frac{\ddot{x}_r}{\ddot{z}_r + g} \leqslant \frac{\frac{144}{25}\frac{b}{T^2}}{-b_2\gamma^2 C + g} \tag{2.49}$$

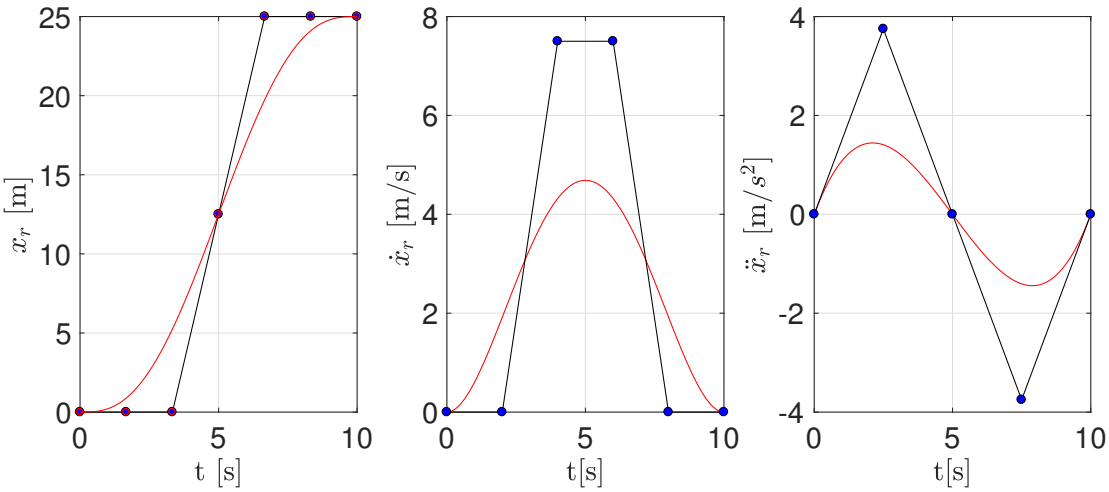


Figure 2.15: The Sigmoid Bézier trajectory x_r, the velocity trajectory $\dot{x}_r$ and the acceleration trajectory $\ddot{x}_r$ with their respective control polygons when $a = 0$ and $b = 25$.

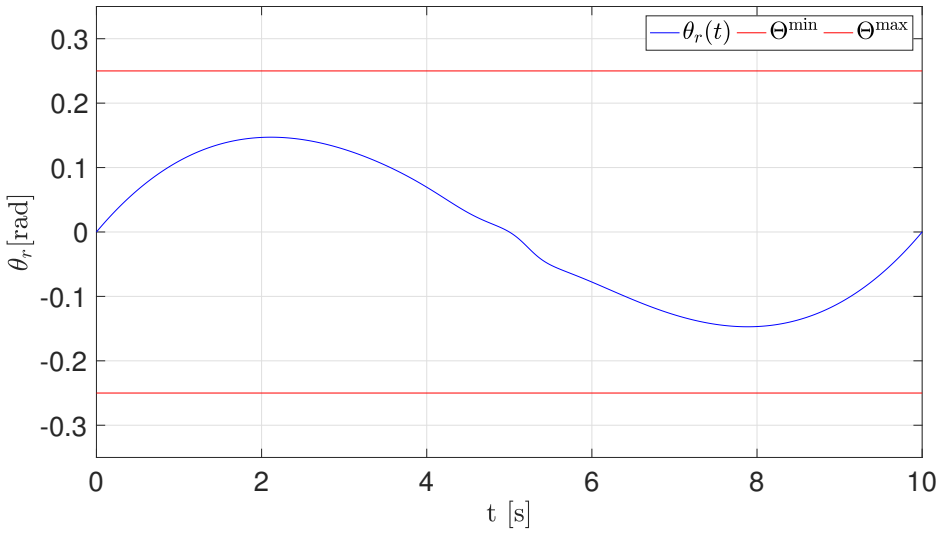


Figure 2.16: The open-loop trajectory $\theta_r(t)$ for Sigmoid Bézier trajectory

2. In a second case, the reference trajectory x_r can be any Bézier curve. However, we need to impose the first and last controls points in order to fix the initial and final equilibrium states. For the example, we take a Bézier trajectory of degree $d = 8$ with control polygon defined as:

$$\boldsymbol{A_x} = \{a, a, a, \alpha_1, \alpha_2, \alpha_3, b, b, b\}\,.$$

When $\gamma = 2$ and $H_i = 0$m, $H_f = 2$m are fixed, the minimum and maximum values for $\ddot{z}_r$ are also fixed. Therefore, to impose constraints on θ_r, it remains to determine $\ddot{x}_r$, *i.e.* the control points of x_r

$$\ddot{x}_r \leqslant (-b_2\gamma^2 C + g)\Theta^{\max} = X^{\max} \approx 1.682m/s^2, \tag{2.50}$$
$$\ddot{x}_r \geqslant -(b_2\gamma^2 C + g)\Theta^{\max} = X^{\min} \approx -3.222m/s^2. \tag{2.51}$$

The initial and final trajectory control points are defined as $x_r(t_0) = a = 0$ and $x_r(t_f) = b = 2$ respectively. Therefore, for $\ddot{x}_r$ where $T = t_f - t_0 = 10$, we obtain the following control polygon $\boldsymbol{A_{\ddot{x}}} = (a_{\ddot{x}i})_{i=0}^{6}$:

$$\boldsymbol{A_{\ddot{x}}} = \left\{0, \frac{14\alpha_1}{25}, \frac{14\alpha_2 - 28\alpha_1}{25}, \frac{14\alpha_1 - 28\alpha_2 + 14\alpha_3}{25}, \frac{14\alpha_2 - 28\alpha_3 + 28}{25}, \frac{14\alpha_3 - 28}{25}, 0\right\}.$$

As explained in the previous section, to reduce the distance between the control polygon and the Bézier curve, we need to elevate the degree of the control polygon $\boldsymbol{A_{\ddot{x}}}$. We elevate the degree of $\boldsymbol{A_{\ddot{x}}}$ up to 16 and we obtain a new augmented control polygon $\boldsymbol{A^A_{\ddot{x}}}$ by using the operation (2.19) (see Figure 2.17 (right)).

The equation (2.50) translates into a system of linear inequalities *i.e. semi-algebraic set* defined as :

$$X^{\min} < a^A_{\ddot{x}i} = f(\alpha_1, \alpha_2, \alpha_3) < X^{\max} \quad i = 0, \ldots, 16. \tag{2.52}$$

We illustrate the feasible regions for the control points by using the Mathematica function *RegionPlot3D* (see Figure 2.18).

Scenario 2: In this scenario, we discuss the output constraints.

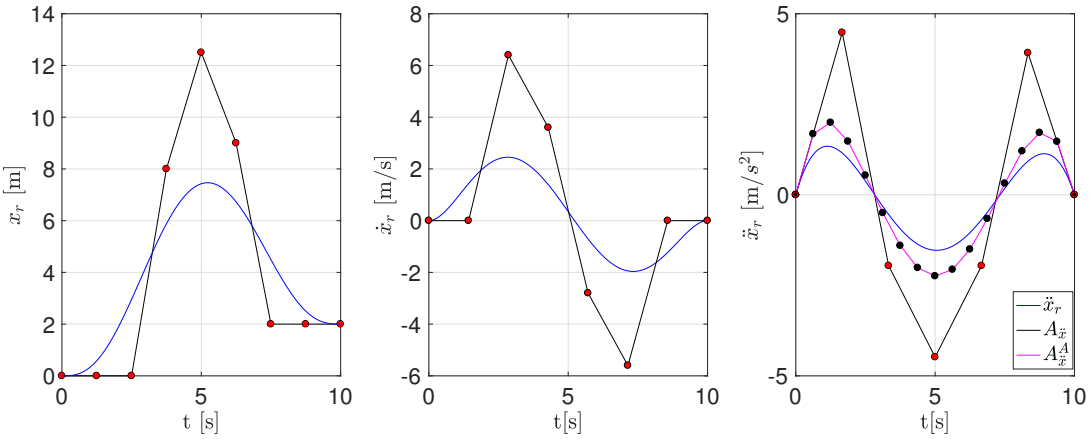


Figure 2.17: The Bézier curve x_r, $\dot{x}_r$, $\ddot{x}_r$ (blue lines) and their respective control polygons (black linear piecewise lines) with $a = 0$, $\alpha_1 = 8$, $\alpha_2 = 12.5$, $\alpha_3 = 9$ and $b = 2$. The augmented control polygon for $\ddot{x}_2$ is represented by the magenta line.

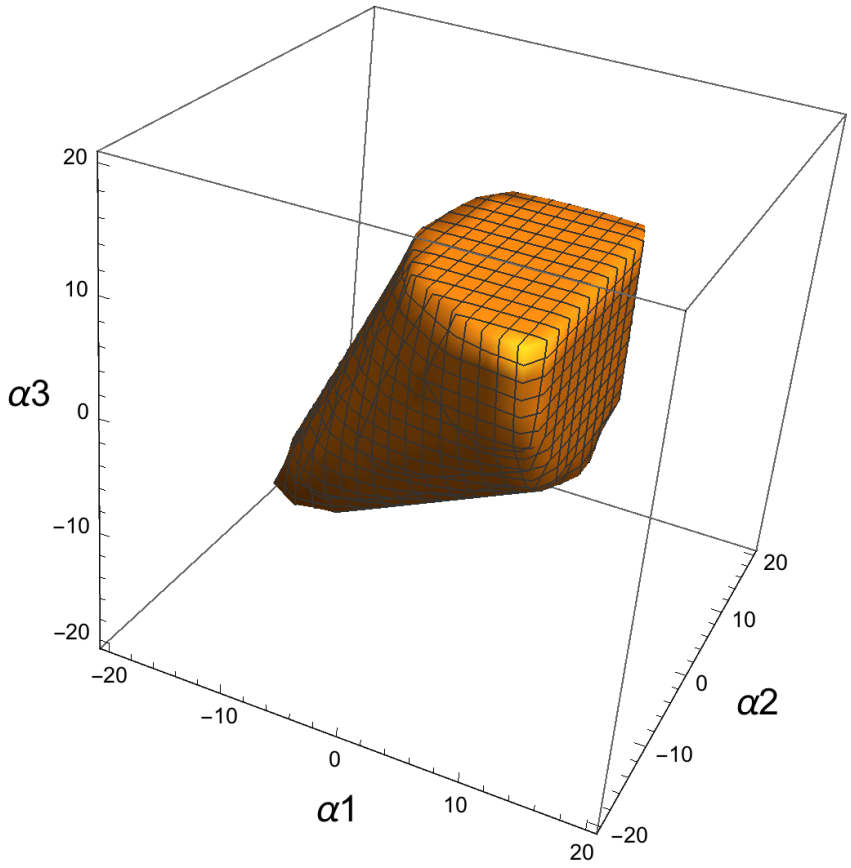


Figure 2.18: Feasible region for the intermediate control points of $x_r(t)$ while fulfilling the constraints on the roll angle.

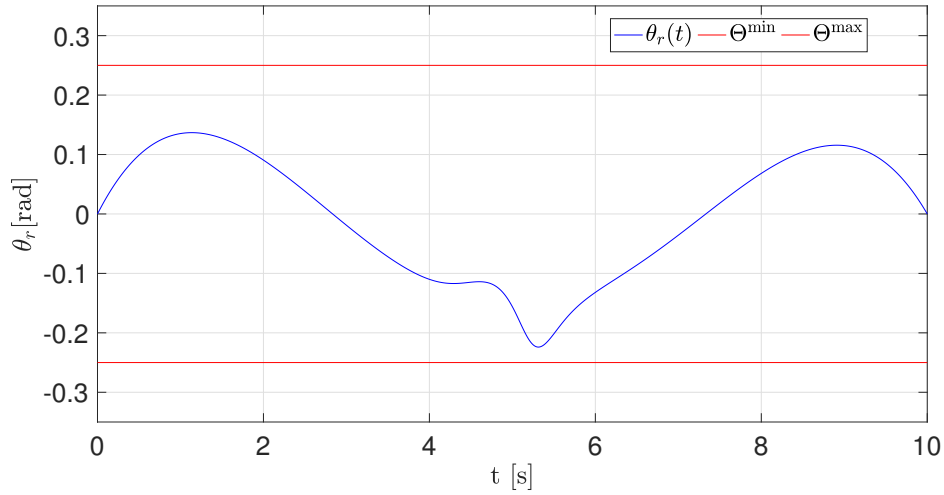


Figure 2.19: The constraints on the open-loop trajectory $\theta_r(t)$ are respected.

Constrained open-loop trajectories x_r and y_r

Here we discuss the scenario when the quadrotor has already been take off by an initial Bézier curve that fulfils the previous input/state constraints and avoids the known static obstacles. Then, suddenly appear new obstacle in the quadrotor environment. To decide, whether the quadrotor should change its trajectory or continue to follow the initial trajectory, we use the quantitative envelopes of the Bézier trajectory presented in Section 2.4.3 to verify if its envelope region overlaps with the regions of the new obstacle.

We construct the quantitative envelopes for x_r and y_r using Section 2.4.3. We find the maximal distance of the Bézier curve w.r.t. to the chosen control polygon. We choose as intermediate control points for x_r and y_r ($\alpha_1 = 8; \alpha_2 = 12.5; \alpha_3 = 9$ and $\beta_1 = 4; \beta_2 = 2.5; \beta_3 = 2$ respectively). The bounded region of the chosen reference trajectories x_r and y_r are depicts in Figure 2.21.

In particular, the figure 2.20 demonstrates the benefit of the bounded trajectory region. We can precisely determine the distance between the quadrotor pathway and the obstacles.

Scenario 3: In this scenario, we discuss the input constraints u_2 and u_3 when the quadrotor is in *hover mode i.e.* moving in a horizontal plane.

Constrained open-loop trajectories u_2 and u_3

By the previous constraints on θ_r and u_{1r}, we implicitly constrain the torque input u_{2r}. A more general case can also be treated if we assume that when the quadrotor

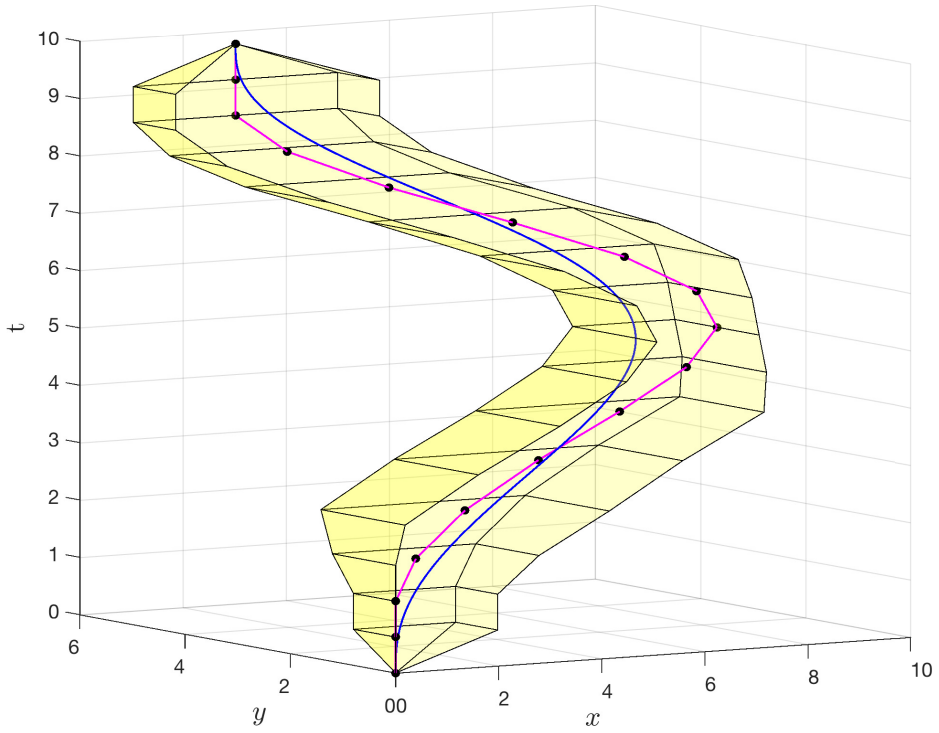


Figure 2.20: The constrained reference trajectories $x_r(t)$ and $y_r(t)$ and their quantitative bounded region w.r.t. to the chosen Bézier control polygon.

reaches the desired altitude, it moves in a horizontal plane. In that case by having slow dynamics for $z_r(t)$ such that $\ddot{z}_r \approx 0$, we therefore have:

$$u_{2r} = C_x x_r^{(4)} \tag{2.53a}$$

$$u_{3r} = C_y y_r^{(4)} \tag{2.53b}$$

where $C_x = \dfrac{I_x}{g}$ and $C_y = -\dfrac{I_y}{g}$ are constants. The latter forms a system of linear inequalities of the control points of x_r and y_r.

Constrained open-loop control for u_{4r}

For u_{4r}, we have a simple double integrator as:

$$u_{4r} = I_z \ddot{\psi}_r \tag{2.54}$$

To find the regions for control points $a_{\psi i}$, we proceed in the same way as in the previous Section 2.7.2.

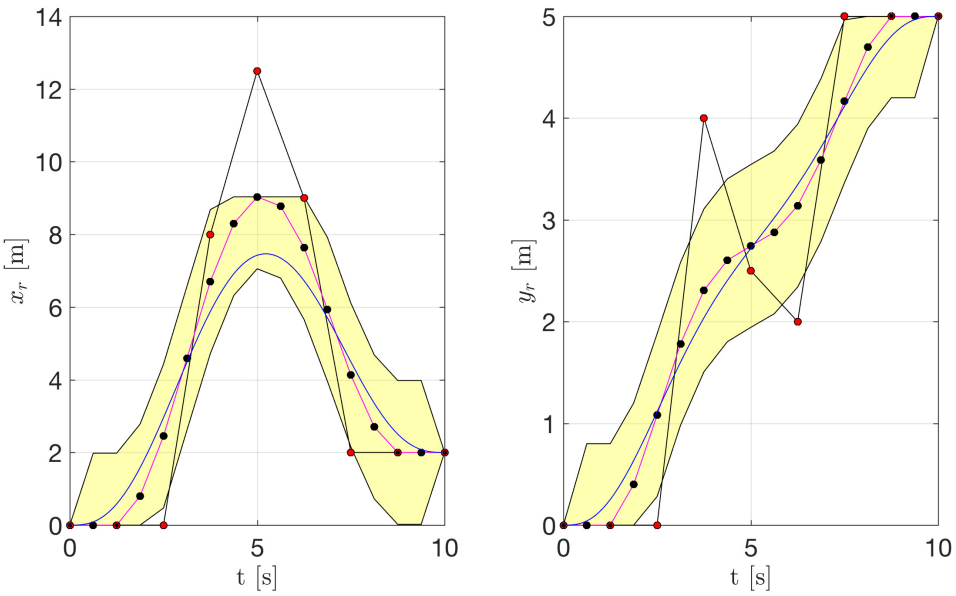


Figure 2.21: The quantitative envelopes for the reference trajectories $x_r(t)$ and $y_r(t)$ (the yellow highlighted regions). The augmented control polygons for $x_r(t)$ and $y_r(t)$ (magenta line). For the simulation, the intermediate control points for x_r and y_r are $\alpha_1 = 8; \alpha_2 = 12.5; \alpha_3 = 9$ and $\beta_1 = 4; \beta_2 = 2.5; \beta_3 = 2$ respectively.

Remark 2.7 *Our constrained trajectory reference study provides a set of feasible reference trajectories. Using the simplified models in the trajectory planning helps us to find the reference trajectory conform to the system dynamics constraints. On the other hand, these models can not serve as a basis for the feedback law design since it will increase the uncertainties and the mismatch with the system. For that purpose, in Chapter 4, we present the non-linear tracking of the aggressive reference trajectories by using a model-free controller.*

2.8 Closing remarks

We have presented a control design for non-linear flat systems handling input/state constraints through the reference trajectory design.

The state/input constraints are translated into a *system of inequalities and equalities* where the variables are the Bézier control points. This enables the input/state/output constraints to be considered into the trajectory design in a unified fashion. This allows us to develop a compact methodology to deal both with control limitations and space constraints as those arising in obstacle avoidance problems.

The core value of this work lies in two important advantages:

- The low complexity of the controller; fast real-time algorithms.
- The choice *i.e.* the user can select the desired feasible trajectory. The sub-optimality may be seen as a drawback.

In the context of trajectory design, we find a successful simpler or approximated semi-algebraic set defined off-line. The closed form solution of the CAD establishes an explicit relationship between the desired constraints and the trajectory parameters. This gives us a rapid insight into how the reference trajectory influences the system behaviour and the constraints fulfillment. Therefore, this method may serve as sensitivity analysis that reflects how the change in the reference trajectory influences the input reference trajectory. Also, for fault-tolerant systems, in spirit of the papers [24, 25, 85, 133], this approach may be useful for the control reconfiguration when an actuator *fault* occurs.

Our algorithm can deal with asymmetric constraints that may be useful in many situations e.g., for a vehicle where acceleration is created by a motor, while deceleration is achieved through the use of a mechanical brake. Increasing tracking errors and environment changes are signs that a re-planning of the reference trajectory is needed. Having the symbolic form of the exact solution, allows us a quick re-evaluation over a new range of output constraints, or with a new set of numerical values for the symbolic variables. In such case, the replanning initial conditions are equivalent to the system current state.

Appendix

2.A Geometrical signification of the Bezier operations

Here we present the geometrical signification of the degree elevation of the Bezier trajectory $y(t)$ (Figure 2.22), the addition (Figure 2.23) and the multiplication (Figure 2.24) of two Bézier trajectories.

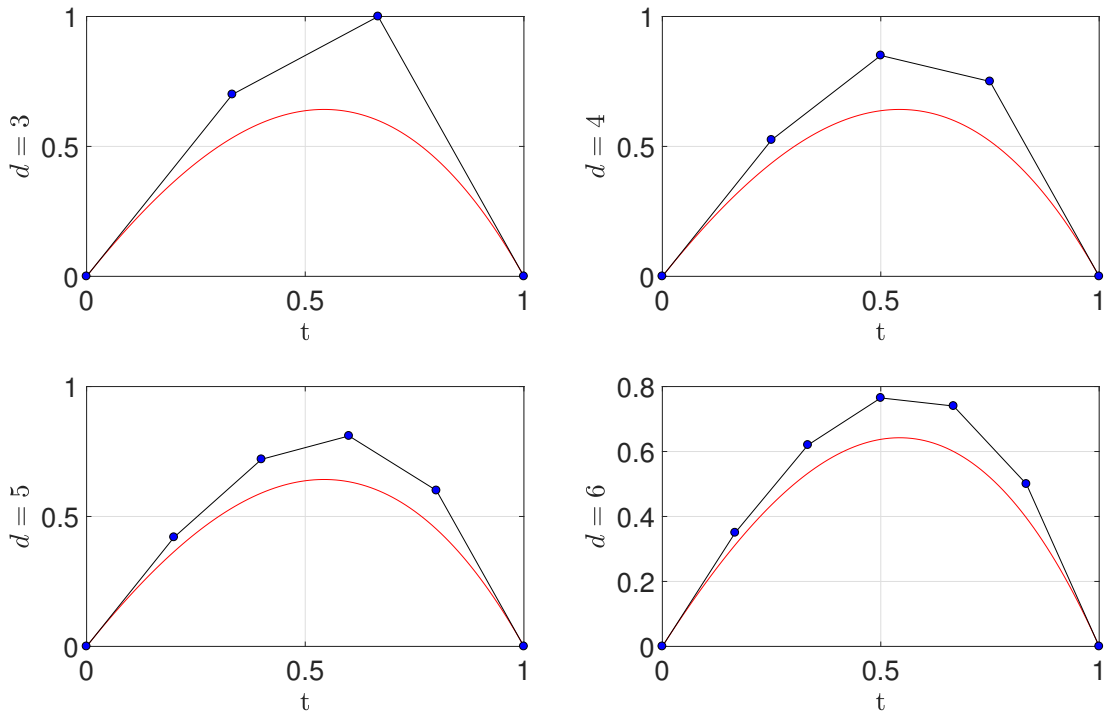


Figure 2.22: Degree Elevation of Bézier curve.

2.B Trajectory Continuity

In the context of feedforwarding trajectories, the "degree of continuity" or the smoothness of the reference trajectory (or curve) is one of the most important

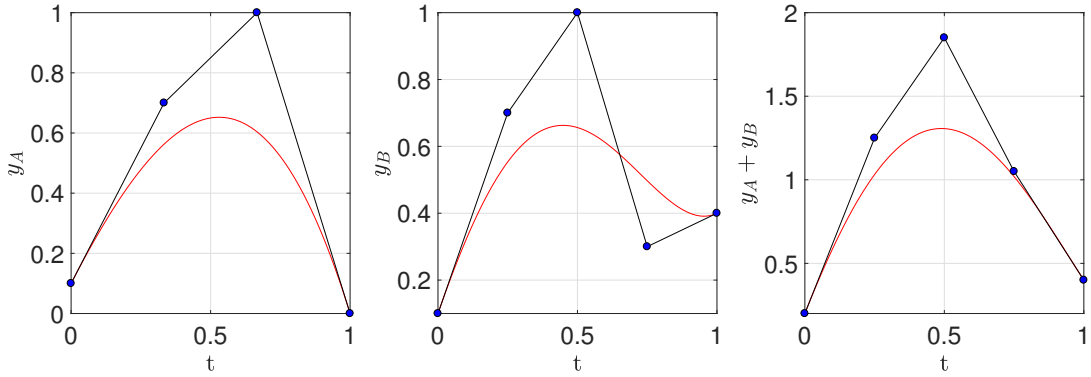


Figure 2.23: Addition of two Bézier curves.

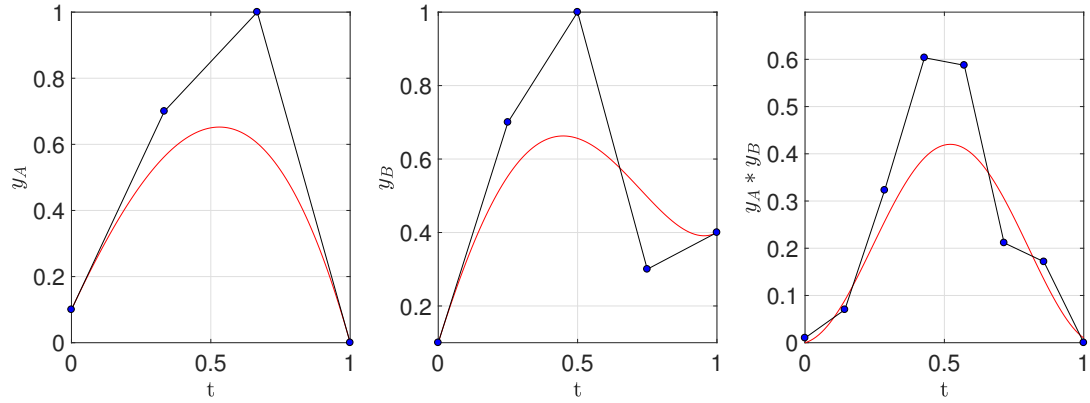


Figure 2.24: Multiplication of two Bézier curves.

factors. The smoothness of a trajectory is measured by the number of its continuous derivatives. We here give some definitions on the trajectory continuity when it is represented by a parametric curve [10].

Parametric continuity A parametric curve $y(t)$ is n-th degree continuous in parameter t, if its n-th derivative $\frac{d^n y(t)}{dt^n}$ is continuous. It is then also called C^n continuous.

The various order of parametric continuity of a curve can be denoted as follows:

- C^0curve *i.e.* the curve is continuous.
- C^1curve *i.e.* first derivative of the curve is continuous. For instance, the velocity is continuous.
- C^2 curve *i.e.* first and second derivatives of the curve are continuous. (The acceleration is continuous)

- C^3curve *i.e.* first, second and third derivatives of the curve are continuous. (the jerk is continuous)
- C^ncurve *i.e.* first through nth derivatives of the curve are continuous.

Example 2.5 *Lets take a linear curve for the joint position of a robot, as:*

$$p(t) = p^i + \frac{p^f - p^i}{T_{tt}} t$$

where p^i is the initial position, p^f is the final position and T_{tt} is the time interval. We obtain for the velocity and the acceleration the following curves:

- *for the velocity:* $v(t) = \dot{p} = \frac{p^f - p^i}{T_{tt}}$
- *for the acceleration* $a(t) = \ddot{p} = \begin{cases} \infty, & t = 0, T_{tt} \\ 0, & 0 < t < T_{tt} \end{cases}$

In this example, we can observe infinite accelerations at endpoints and discontinuous velocity when two trajectory segments are connected.

3

Constraints on Linear Flat Systems with Delays

Contents

Abstract: We consider the control of differentially flat linear delay systems with constraints. The constraints can be given on the state and/or on the control. Linear delay systems are here envisioned as modules over a ring of differential and distributed delay operators. Due to the nice Bezout property that this ring enjoys, the controllability notions of freeness, projectivity and torsion freeness coincide. Thanks to the flatness (corresponding to freeness for linear systems) property, all constraints are reported on the flat output (the basis of the corresponding module). We then make use of polynomial B-splines as specialisations for the flat output; the constraints are finally expressed as inequalities in these B-splines control points.

3.1 Chapter Overview

This chapter addresses our previous result from Section 2 to the case of *linear delay systems*.

We propose a control design technique based on the R-freeness property for infinite linear systems (delay systems [51, 99, 101], partial differential equations [61, 100]) which is an extension of the differential flatness originally developed for finite dimensional systems. Other algebraic related approaches include [28, 112, 122]. According to this method, all the states and the control input of the linear delay system can be parametrized through a so-called R-flat output by using differentiations, delays and advances. In other words, the nominal input and the states can be expressed as a linear combination of the delayed and advanced reference trajectory and its derivatives.

As in the previous chapter, our goal is to write the *state/input constraints* in terms of the R-flat output and its derivatives. The idea is to translate the beauty and simplicity of the freeness property of a delay system (analogue of flatness for finite dimensional system) into constrained control via B-spline procedures for the reference trajectories.

Due to the linearity of the system we are able to present it through the context of the B-spline curves (no more need of the product operator). In some applications we need to keep fixed some part of the curve and to modify only a specific region; this can be achieved very naturally in the context of the B-spline curves. The Bezier curve can accurately represent a trajectory, but it has the drawback that any modification of the control points will modify the entire curve. The adoption of B-spline curves is motivated by their peculiar properties, which allow a natural remapping of the constraints from the input/state to the flat output, while leaving

sufficient flexibility to express a rich class of reference trajectories. We establish explicit relations between the control points of the B-spline describing the reference trajectory and those of the B-spline expressing the control input and states with delays. This way, the constraints on input and states is directly translated in a set of inequalities for the control points of the reference trajectory.

Once the constrained open-loop trajectories are generated offline, and in order to guarantee the stability and a certain robustness of the approach, we need a *feedback control*. There are many different linear and nonlinear feedback controls that can be used to ensure convergence to zero of the tracking error. We obtain a stable trajectory tracking with prescribed tracking error dynamics if distributed delays are admitted in the feedback law. This is a model based prediction.

Outline

The outline of this chapter is as follows.

- In section 3.2, we recall the definition of π-freeness for delay linear systems and in the section 3.3, we give a stabilizing feedback law for a class of delay systems that makes use of predictor forms elaborated with distributed delays.
- In section 3.4, we give an overview of the B-splines curves and its properties.
- In section 3.5, we detail the procedure in establishing reference trajectories for constrained open-loop control.
- In section 3.6, we illustrate an example on car-following with human memory effects.

The results from this chapter have been published in [13].

3.2 R-freeness for delay linear systems

We shall use a module theoretic approach framework developed, among others, in [50, 99]. The adopted framework emphasizes on equations (rather than solutions) in order to study a given system. When dealing with linear equations, a system is associated with a module over a ring, this notion playing for differential equations the role played by vector spaces for linear algebraic equations. The basic definitions can be found in [45, 99].

3.2.1 Algebraic setting and preliminaries

We shall consider linear delay systems as modules over the polynomial ring $\mathbb{R}[\frac{d}{dt}, \delta_1, \ldots, \delta_r]$ where the δ_i's play the role of localized delay operators. This ring is isomorphic to the ring $\mathbb{R}[s, e^{-h_1 s}, \ldots, e^{-h_r s}]$ (the variable s plays the role of $\frac{d}{dt}$, the h_i's being the amplitudes of the corresponding delays). In order to involve distributed delays, we use an extended ring: $\mathfrak{S}_r = \mathbb{R}(s)[e^{-h_1 s}, \ldots, e^{-h_r s}] \cap \mathfrak{E}$, where $\mathfrak{E}$ denote the ring of entire functions. This ring is a Bézout domain *i.e.* any finitely generated ideal in this domain is principal. A typical element of $\mathfrak{S}_r$ is $(1 - e^{-h_i s})/s$ (it is an entire function, since $1 - e^{-h_i s} = h_i s - h_i^2 s^2 + h_i^3 s^3 + \cdots$ is zero when $s = 0$), which corresponds to a distributed delay operator in the time domain. Another slightly larger ring is $\mathbb{R}[s, s^{-1}, e^{-h_1 s}, \ldots, e^{-h_r s}]$ which contains the integration (through application of the s^{-1} operator).

Definition 3.1 *Given a ring R (commutative, with unity and no zero divisors, such as one of the above), an R-*system *is a module over R.*

We shall consider three controllability notions, corresponding to algebraic properties of the corresponding module.

Definition 3.2 *An R-system Λ is called R-*torsion free *(resp.* projective, free*) controllable if the corresponding module is torsion free (resp. projective, free).*

Let us recall that, on a Bezout ring (as well as on a principal ideal domain such as $\mathbb{R}[\frac{d}{dt}]$), the three notions coincide.

In the next sections, by using this R-freeness formalism, we obtain all the system open-loop trajectories z_r (the states and the inputs) as functions of the R-flat output y_r, a finite number of its derivatives, time delays, and advances.

In the case of a R-free delay system, we embed *constraints* $K_l, K_h \in \mathbb{R}$ on a system open-loop trajectory by imposing:

$$\begin{aligned} &K_l \leqslant z_r \leqslant K_h \text{ with} \\ &z_r = R(y_r, \dot{y}_r, \ldots, \delta_i^{\pm j} y_r^{(q)}, \theta y_r^{(q)}, \ldots, \delta_i^{\pm j} y_r^{(\gamma)}, \theta y_r^{(\gamma)}), \end{aligned}$$

where $\delta_i^{\pm j} y_r(t) = y_r(t \mp j\tau_i)$, are delays and advances respectively, and $(\theta y_r)(t) = \int_{t-h}^{t} e^{(t-\tau)} y_r(\tau) d\tau$ represents a distributed delay.

3.3 Stabilization of the system

Here by the means of distributed delays in the feedback law, we avoid pure predictions (torsion-free controllable) *i.e.* the delay is compensated by the controller [102]. The control law achieves asymptotic tracking compensating the effects of the input delay. With this, we want to overcome the delay in the closed-loop which may be a source of poor system performance and instability.

Let us first consider one of the simplest system, *i.e.* a linear system with commensurate delay in the input

$$\dot{y}(t) = u(t-h) \tag{3.1}$$

for which the open-loop control yields

$$u_r(t) = \dot{y}_r(t+h).$$

For the closed-loop control, setting

$$u(t) = \dot{y}_r(t+h) - K_p e(t+h), \quad e(t) = y(t) - y_r(t) \tag{3.2}$$

$$e(t+h) = \int_t^{t+h} \dot{e}(\tau)d\tau + e(t)$$

we obtain

$$\begin{aligned} u(t) &= \dot{y}_r(t+h) - K_p \int_t^{t+h} \dot{e}(\tau)d\tau - K_p e(t) \\ &= K_p\Big(- \int_t^{t+h} \dot{y}(\tau)d\tau + \int_t^{t+h} \dot{y}_r(\tau)d\tau - e(t)\Big) + \dot{y}_r(t+h) \end{aligned}$$

Finally, we obtain a closed-loop control

$$u(t) = K_p\Big(- \int_{t-h}^{t} u(\tau)d\tau + \int_t^{t+h} \dot{y}_r(\tau)d\tau - e(t)\Big) + \dot{y}_r(t+h)$$

which involves only distributed delays of finite support but no pure predictions.

3.4 B-splines preliminaries

Using a B-spline curve as reference trajectory is a simple way to reduce the problem of infinite unspecified function $f(t)$ into a finite dimensional one determined by control points c_j associated to a basis functions $B_{j,d}$.

3.4.1 B-splines

The B-spline $B_{j,d}$ depends on the knots $t_j, \ldots, t_{j+1+d}$. This means that if the knot vector is given by $\boldsymbol{t} = (t_j)_{j=1}^{n+d+1}$ for some positive integer n, we can form n B-splines $\{B_{j,d}\}_{j=1}^{n}$ of degree d associated with this knot vector. A B-spline curve (or a linear combination of B-splines) is a combination of B-splines of the form

$$f = \sum_{j=1}^{n} c_j B_{j,d} \tag{3.3}$$

where $\boldsymbol{c} = (c_j)_{j=1}^{n}$ are n real numbers. We formalise this in a definition.

Definition 3.3 *(B-spline curves). Let*

$$\boldsymbol{t} = (t_j)_{j=1}^{m=n+d+1} = [\underbrace{0, \ldots, 0}_{d+1}, t_{d+1}, \ldots, t_{m-d-1}, \underbrace{1, \ldots, 1}_{d+1}]$$

be a non-decreasing sequence of real numbers, i.e. *, a knot vector for a total of n B-splines. The linear space of all linear combinations of these B-splines is the spline space $\mathbb{S}_{d,\boldsymbol{t}}$ defined by*

$$\mathbb{S}_{d,\boldsymbol{t}} = \left\{ \sum_{j=1}^{n} c_j B_{j,d} \mid c_j \in \mathbb{R} \text{ for } 1 \leqslant j \leqslant n \right\}$$

An element $f = \sum_{j=1}^{n} c_j B_{j,d}$ of $\mathbb{S}_{d,\boldsymbol{t}}$ is called a B-spline curve or a spline function, of degree d with knots $\boldsymbol{t}$, and $(c_j)_{j=1}^{n}$ are called the control points of the B-spline curve (see Fig.3.1).

Definition 3.4 *(Cox-DeBoor recursion formula) Let d be a nonnegative integer and let $\boldsymbol{t} = (t_j)$, the knot vector or knot sequence, be a non-decreasing sequence of real numbers of at least $d+2$. The jth B-spline of degree d (order k) with knots $\boldsymbol{t}$ is defined by:*

$$B_{j,k,\boldsymbol{t}}(x) = \frac{x - t_j}{t_{j+d} - t_j} B_{j,d-1,\boldsymbol{t}}(x) + \frac{t_{j+k} - x}{t_{j+1+d} - t_{j+1}} B_{j+1,d-1,\boldsymbol{t}}(x)$$

for all real numbers x, with

$$B_{j,0,\boldsymbol{t}}(x) = \begin{cases} 1, & \text{if } t_j \leqslant x < t_{j+1} \\ 0, & \text{otherwise} \end{cases}$$

Remark 3.1

- *Choosing the knot vector in this way guarantees the start and end tangent property.*

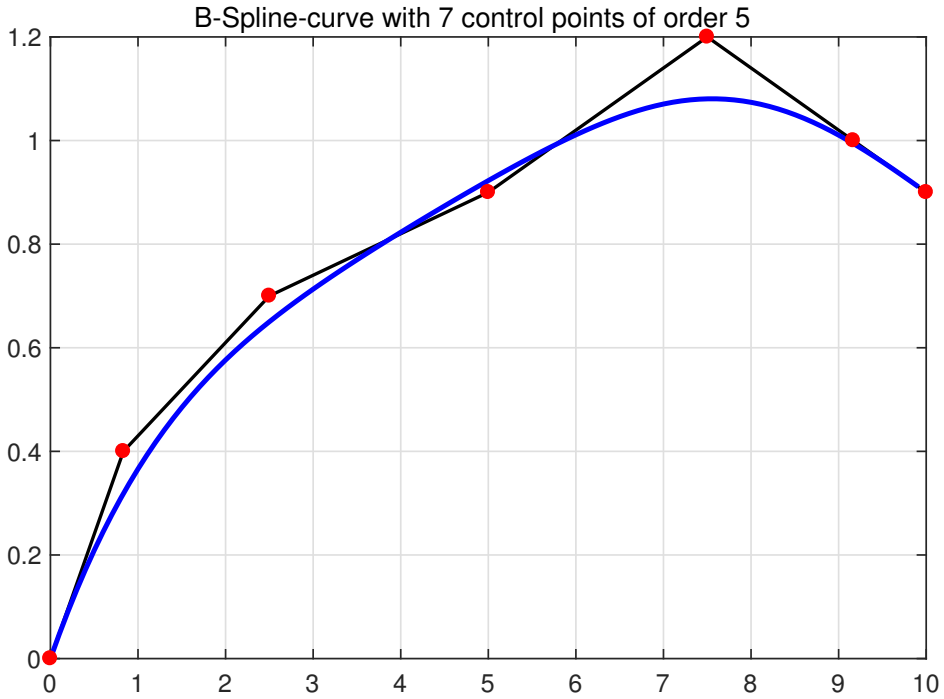


Figure 3.1: The B-spline curve and its control polygon.

- *Each control point movement only has local effects.*

Definition 3.5 *(Control polygon for B-spline curve (see [83])). Let* $f = \sum_{j+1}^{n} c_j B_{j,d}$ *be a spline in* $\mathbb{S}_{d,\boldsymbol{t}}$*. the control points of* f *are the points with coordinates* (t_j^*, c_j) *for* $j = 1, \ldots, n$ *where*

$$t_j^* = \frac{t_{j+1} + \cdots + t_{j+d}}{d} \tag{3.4}$$

are the knot averages of $\boldsymbol{t}$*. The control polygon of* f *is the piecewise linear function obtained by connecting neighbouring points by straight lines.*

3.4.2 B-spline properties

B-splines play a central role in the representation of B-spline curves. For that purpose, we report here the most important properties.

Lemma 3.1 *([83], page 40) Let* d *be a nonnegative polynomial degree and let* $\boldsymbol{t} = (t_j)$ *be a knot sequence. The B-splines on* $\boldsymbol{t}$ *have the following properties:*

1. *Local knots. The* j*th B-splines* $B_{j,d}$ *depends only on the knots* $t_j, t_{j+1}, \ldots, t_{j+d+1}$.

2. *Local support*

- *If x is outside the interval $[t_j, t_{j+d+1})$ then $B_{j,d}(x) = 0$. In particular, if $t_j = t_{j+d+1}$ then $B_{j,d}$ is identically zero.*
- *If x lies in the interval $[t_\mu, t_{\mu+1})$ then $B_{j,d}(x) = 0$ if $j < \mu - d$ or $j > \mu$.*

3. *Positivity. If $x \in (t_j, t_{j+d+1})$ then $B_{j,d}(x) > 0$. the closed interval $[t_j, t_{j+d+1}]$ is called the support of $B_{j,d}$.*

4. *Piecewise polynomial. The B-spline $B_{j,d}$ can be written*

$$B_{j,d}(x) = \sum_{k=j}^{j+d} B_{j,d}^k(x) B_{k,0}(x) \tag{3.5}$$

where each $B_{j,d}^k(x)$ is a polynomial of degree d.

5. *Special values. If $z = t_{j+1} = \ldots = t_{j+d} < t_{j+d+1}$ then $B_{j,d}(z) = 1$ and $B_{i,d}(z) = 0$ for $i \neq j$.*

6. *Smoothness. If the number z occurs m times among $t_j, t_{j+1}, \ldots, t_{j+d+1}$ then the derivatives of $B_{j,d}$ of order $0, 1, \ldots, d - m$ are all continuous at z.*

3.5 Constrained Trajectory Generation Procedure

In this section we present the design of the desired B-spline trajectory $y_r(t)$. The initial equilibrium point is its first control point $y_{initial} = c_0$ and the final equilibrium point is its last control point $y_{final} = c_n$.

For the sake of completeness, we state a few necessary B-spline ingredients (derivative, integral, and degree elevation) that are crucial for our procedure.

3.5.1 Derivative property of the B-spline curve

Theorem 3.1 *(see [83]) The derivative of the jth B-spline of degree d on $\boldsymbol{t}$ is given by*

$$DB_{j,d}(x) = d\left(\frac{B_{j,d-1}(x)}{t_{j+d} - t_j} - \frac{B_{j+1,d-1}(x)}{t_{j+1+d} - t_{j+1}}\right) \tag{3.6}$$

for $d \geqslant 1$ and for any real number x.

According to the previous theorem, if the flat output or the reference trajectory $\boldsymbol{y_r}$ is a B-spline curve, its derivative is still a B-spline curve and we can explicitly compute its control points.

Let $y^{(\nu)}(x)$ denote the ν^{th} derivative of $y(x)$. If x is fixed, we can obtain $y^{(\nu)}(x)$ by computing the νth derivatives of the basis functions:

$$y^{(\nu)}(x) = \sum_{j=1}^{n} c_j B_{j,d}^{(\nu)}(x) \tag{3.7}$$

Letting $c_j^{(0)} = c_j$, we write

$$y(x) = y^{(0)}(x) = \sum_{j=1}^{n} c_j^{(0)} B_{j,d}(x) \tag{3.8}$$

Then,

$$y^{(\nu)}(x) = \sum_{j=1}^{n-\nu} c_j^{(\nu)} B_{j,d-\nu}(x) \tag{3.9}$$

with derivative control points such that

$$c_j^{(\nu)} = \begin{cases} c_j, & \nu = 0 \\ \dfrac{d-\nu+1}{t_{j+d+1} - t_{j+\nu}} \left(c_{j+1}^{(\nu-1)} - c_j^{(\nu-1)} \right), & \nu > 0 \end{cases} \tag{3.10}$$

and a vector knot

$$\boldsymbol{t}^{(\nu)} = \underbrace{0, \ldots, 0}_{d-\nu+1}, t_{d+1}, \ldots, t_{m-d-1}, \underbrace{1, \ldots, 1}_{d-\nu+1}$$

Remark 3.2 *We should take a B-spline curve of degree $d > \nu$, where ν is the derivation order of the flat output, to avoid introducing discontinuities.*

3.5.2 Integral property of B-spline curve

Similar to the derivation operation, an integral of a B-spline is a B-spline and we are able to find the control points of the integral of the B-spline curve in terms of the control points of the initial B-spline curve.
The indefinite integral of a B-spline function $f(x)$ (see [34]):

$$f(x) = \sum_{j=1}^{n} c_j B_{j,d}(x) \tag{3.11}$$

on the knot vector $(t_j)_{j=1}^{n+d+1}$ is given by the B-spline function $g(x)$ where

$$g(x) = \int_{t_1}^{x} \sum_{j=1}^{n} c_j B_{j,k}(u) du = \sum_{j=1}^{n} \left(\frac{t_{j+k+1} - t_j}{k} \sum_{i=1}^{i} c_i \right) B_{j,k+1}(x), \quad t_k \leqslant x \leqslant t_{n+1}. \tag{3.12}$$

Hence the integral of a B-spline is presented as:

$$g(x) = \int_{t_1}^{x} f(u)du = \sum_{j=1}^{n+1} e_j B_{j,k+1}(x), \tag{3.13}$$

where

$$e_1 = 0, \quad e_{j+1} = \frac{1}{d+1} \sum_{i=1}^{j} c_i \left(t_{i+d+1} - t_i\right) \qquad 1 \leqslant j \leqslant n. \tag{3.14}$$

The knot vector for $g(x)$ matches that of the original curve except for the extra knot at both ends due to the increased degree. For a definite integral of a B-spline we have:

$$\int_{x_1}^{x_2} f(u)du = \int_{t_1}^{x_2} f(u)du - \int_{t_1}^{x_1} f(u)du = g(x_2) - g(x_1),$$
$$t_k \leqslant x_1, \quad x_2 \leqslant t_{n+1}.$$

Remark 3.3 *Notice that the input constraints in the presence of commensurable and/or distributed known delay in the state $x(t-h)$ or $\int_{t-h}^{t} x(\tau)d\tau$ can be given in straightforward algebraic manner.*

Thanks to the integral property, we can easily deal with distributed delay in the state. For instance, consider a system in the form:

$$\dot{y} = \int_{t-h}^{t} y(\tau)d\tau + y(t-h) + u \tag{3.15}$$

where y is the flat output. An open loop control allowing the tracking of y_r by y is:

$$u_r = \dot{y}_r(t+h) - \int_{t-h}^{t} y_r(\tau)d\tau - y_r(t+h). \tag{3.16}$$

3.5.3 Degree elevation and knot insertion

To accomplish an addition and/or subtraction of two B-spline curves $f(x)$ and $g(x)$ with different degrees d_f and d_g respectively s.t. $d_f < d_g$, first, we need to increase $(d_g - d_f)$ times the degree of $f(x)$. A good visual algorithm of the degree elevation and knot insertion of the B-spline reference trajectory can be found in [113].

3.5.4 Reference trajectory design procedure

The simple-minded idea on the reference trajectory design is based on the following steps:

- Assign a B-spline reference trajectory to each flat output.
- Find the analytical B-spline expressions of the states and the inputs.

- Express the input/state constraints as inequalities in terms of the B-spline control points and find the suitable region for each control point of the B-spline reference trajectory by using the B-spline properties (see the previous sections 3.5.1, 3.5.2, 3.5.3).

We find the relationship between the input control points U_i and the control points of the reference trajectory c_j such that $U_i = \Phi_i(c_1, \ldots, c_n)$. Our aim is to constraint $K_l \leqslant |U_i| \leqslant K_h$ by choosing suitable regions for the reference trajectory.

Remark 3.4 *To solve the system of inequalities, we use symbolic computation of the Cylindrical Algebraic Decomposition (CAD) algorithm which is the best currently known algorithms for solving many classes of problems related to systems of real polynomial equations and inequalities [130]. By using Cylindrical Algebraic Decomposition, we compute the regions in which one chooses the values for c_j's of the reference trajectory.*

3.6 Example: Car-following model

We investigate a car-following model including human drivers memory effects from [124]. For the sake of clarity, we consider a simplified example. The dynamics of two vehicles, when the first vehicle follows the second vehicle is represented by the following equations:

$$\ddot{y}_1 = \alpha \int_h^{h+\delta} f(\tau) H(t-\tau) d\tau - K_p(y_2 - y_1) \tag{3.17a}$$

$$\ddot{y}_2 = u \tag{3.17b}$$

where y_1 and y_2 are the positions of the first and the second vehicle respectively, $H(t) = y_2(t) - y_1(t)$ is the headway perturbation between the vehicles, u is the motor torque, taken as control input and the constants: $\alpha > 0$ is the measure of the driver's aggressiveness per unit vehicle mass and K_p is the human regulation parameter. The delayed action/decision of human drivers is represented using distributed delays. As distribution function $f(t)$, as stated in [124], we take the uniform distribution, which is a good fit for modelling the short-term memory of drivers:

$$f(\tau) = \begin{cases} \dfrac{1}{\delta}, & h \leqslant \tau \leqslant h + \delta \\ 0, & \text{otherwise,} \end{cases} \tag{3.18}$$

where h is the memory dead-time and δ is the memory window. For the sake of simplicity, we take $\delta = (k-1)h$, $k \in \mathbb{N}$. The model (3.17a)-(3.17b) is $R[s, s^{-1}, e^{-hs}]$-free, with basis (or flat output) given by H, *i.e.* all system variables can be differentially parametrized by H.

With the notation $\hat{y}_1$, $\hat{y}_2$ and $\hat{u}$ for the Laplace transform, (3.17a)-(3.17b) is given by:

$$s^2\hat{y}_1 = \alpha e^{-hs}\frac{1-e^{-\delta s}}{\delta s}(\hat{y}_2 - \hat{y}_1) - K_p(\hat{y}_2 - \hat{y}_1), \tag{3.19}$$

$$s^2\hat{y}_2 = \hat{u} \tag{3.20}$$

Let

$$a(s) = s^2 - K_p + \alpha e^{-hs}\frac{1-e^{-\delta s}}{\delta s}.$$

We get a differential parametrization of the system as

$$\begin{aligned}\hat{y}_1 &= \left(\frac{a(s)}{s^2} - 1\right)\hat{H},\\ \hat{y}_2 &= \frac{a(s)}{s^2}\hat{H},\\ \hat{u} &= a(s)\hat{H}.\end{aligned}$$

From where, we obtain the time domain expression for the open loop control:

$$u_r(t) = \ddot{H}_r(t) - K_p H_r(t) + \frac{\alpha}{\delta}\int_{t-\delta}^{t} H_r(\tau - h)d\tau \tag{3.21}$$

$$u_r(t) = \underbrace{\ddot{H}_r(t)}_{\text{first term}} - K_p \underbrace{H_r(t)}_{\text{second term}} + \frac{\alpha}{(k-1)h}\underbrace{\int_{t-kh}^{t-h} H_r(\tau)d\tau}_{\text{third term}} \tag{3.22}$$

We take as symbolic reference trajectory H_r a B-spline curve with degree $d = 4$, knot vector

$$T = \{0, 0, 0, 0, 0, 10/3, 20/3, 10, 10, 10, 10, 10\}$$

and control points vector $\boldsymbol{A} = (a_j)_{j=1}^{7}$ as

$$H_r = \sum_{j=1}^{7} a_j B_{j,d}. \tag{3.23}$$

The *constraints* we consider are the following:

1. Distance constraint: $H_{\min} \leqslant H \leqslant H_{\max}$

2. Actuator limit: $U_{\min} \leqslant u \leqslant U_{\max}$.

The first constraint will be respected by choosing the control points for the reference trajectory H_r such that $H_{\min} \leqslant a_j \leqslant H_{\max}$.

For the second constraint, using the properties of the B-spline curve, we can find the control points of the open-loop control u_r in terms of the a_j's by following these steps:

1. First, we find the control points $a_j^{(2)}$ for the second derivative $\ddot{H}_{xr}$ by using the formula (3.10):
$$\ddot{H}_{xr} = \sum_{j=1}^{5} a_j^{(2)} B_{j,1}(t)$$

2. We obtain the third term $\int_{t-kh}^{t-h} H(\tau)d\tau$ by
$$\int_{t-kh}^{t-h} H(\tau)d\tau = \sum_{j=1}^{8} e_j B_{j,5}(t)$$
which is a B-spline curve of degree 5 and where the control points e_i are calculated by the integral operation (3.14).

3. We elevate the degree of the first term and the second term up to 5 and then, we add additional knots in order to end up with the same number of control points in the three terms. After, we can find the sum of these terms. We end up with u_r as a B-spline curve of degree 8 with control points U_i:
$$u_r(t) = \sum_{i=1}^{14} U_i B_{i,5}(t)$$

We want all the input control points to respect the actuator limits $U_{\min} \leqslant u \leqslant U_{\max}$. The latter form a system of inequalities that can be used as a prior study to the sensibility of the control inputs with respect to the flat outputs. To solve this system, we use the Mathematica function *CylindricalDecomposition* for the symbolic computation of the Cylindrical Algebraic Decomposition. We compute the regions in which to choose the values for a_i's of the reference trajectory. For the sake of clarity, instead of keeping $U_{\min}, U_{\max}$ symbolically, we give a value for the constraints $U_{\min} = 0.2$ and $U_{\max} = 10$. The initial and final trajectory points are defined as $H_r(t_0) = a_1$ and $H_r(t_f) = a_7$ respectively. The condition under which the reference trajectory H_r will respect the input constraint is

$$\begin{aligned} &a_2 \in \mathbb{R} \\ &a_i < \frac{1}{20}(1 - 20a_{i-2} + 40a_{i-1}), \text{for} \quad 3 \leqslant i \leqslant 6. \end{aligned}$$

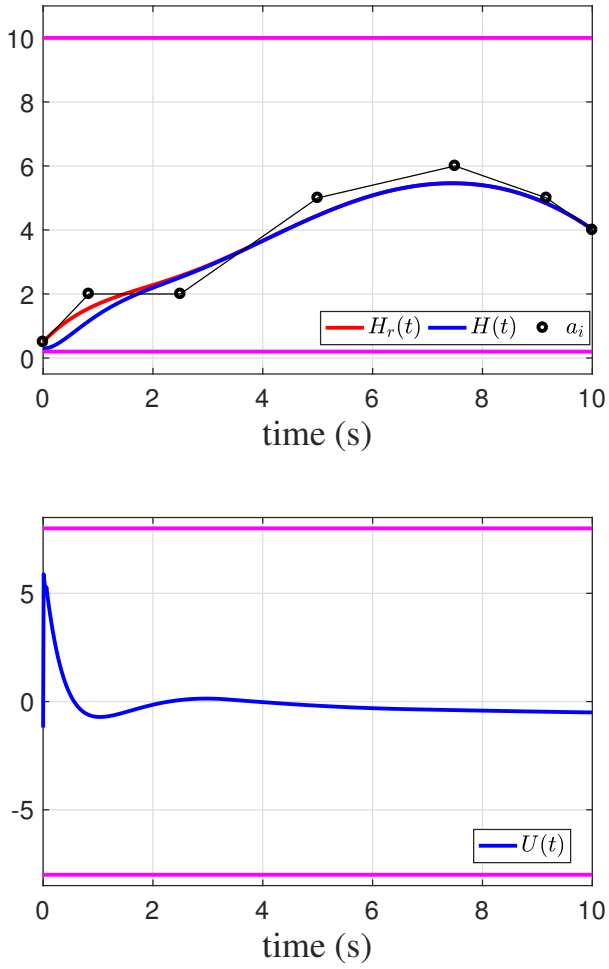


Figure 3.2: Closed-loop performance.

The reference 4th degree B-spline trajectory is specified with the control points $a_1 = 0.5$; $a_2 = 2$; $a_3 = 2$; $a_4 = 5$; $a_5 = 6$; $a_6 = 5$; $a_7 = 4$ chosen in the constrained region.

Figure 3.2 depicts the performance of the closed-loop control.

3.7 Closing remarks

This work seeks to find an explicit constraint on the control input and/or the state of a linear delay system. It thus provides a useful tool that can be implemented on various applications using the B-spline curves and flat system theory. By expression of the flat outputs in the form of B-spline curves, the input controls depend on

the control points and the degree of the B-spline curves (flat outputs). In our future works, we shall develop our approach further for systems represented by partial differential equations.

4

Cascaded Model-Free Control of Quadrotors

Contents

Abstract: In the subject of quadrotor controller design, usually modelling and identification are tedious and time-consuming tasks. In this chapter, we propose a controller design that avoids the quadrotor's system identification procedures while staying robust with respect to endogenous and exogenous disturbances. To reach our goal, based on the cascaded structure of a quadrotor, we divide the system into positional and attitude subsystems each controlled by an independent model-free controller. We validate our control approach in three realistic scenarios: in presence of unknown measurement noise, unknown time-varying wind disturbances and mass variation. We provide simulations on a realistic nonlinear quadrotor model following an aggressive position-yaw trajectory.

4.1 Chapter overview

In the last decade the use of unmanned aerial vehicles (UAVs) has increased significantly. From package delivery services, military uses and disaster management to photography and entertainment, the use-cases are numerous. Among the UAVs, quadrotors have been subject to intense research and development. One of the main challenges is modeling quadrotor dynamics and parameters estimation. This process can be tedious, time consuming and prone to modeling errors. A considerable amount of the literature has been devoted to model-based controls (Backstepping and sliding-mode techniques [19], Inner-Outer Loop Control by applying PID controller for the inner-loop and nested saturation controller for the outer-loop [21], Generalized Proportional Integral (GPI) observer based controller [125], Model Predictive Control [2]) among other methods. In these model-based approaches, the modeling accuracy can directly impact the controller success and performance. Moreover, many quadrotor applications are outdoor and may be faced with environmental uncertainties. Among the meteorological uncertainties, the wind effect has one of the highest impacts on the quadrotor performance while being very hard to predict. Therefore, in the model definition, the wind is usually considered as a predefined constant. As suggested in the papers [18, 92, 115], in practice, using a PID have shown more advantages than more advanced quadrotor controllers because of its simplicity and robustness. We chose to use the model-free control (MFC) since it has been already shown successful in practice (see [48], [33] and the references therein, and [1, 94, 114]) that the MFC have better performances and advantages over a simple PID. These facts motivate us to present a *cascaded-model-free approach* for a quadrotor while considering more realistic situations like time-varying wind disturbances, measurement noise and mass value variation. We make our contribution in the following framework:

- We use a minimalist structure upon which our control scheme rests (see Equations (4.12), (4.14), (4.19a)-(4.19c)). This structure is independent of any mass, inertia, gyroscopic or aerodynamic effects; the only information we use is that each positional and rotational dynamics is of second order and that the thrust produced by the propellers is suitably projected onto the inertial frame.

- Because of the underactuated nature of the quadrotor, we consider an inner-outer structure of the quadrotor that allows us to divide the control into two stages. At the first stage, we show that it is possible to apply a model-free control in the outer loop and from there, we derive the desired thrust, and the desired roll and pitch angles. We then apply again the model-free control in the inner loop for the second stage.

- No precise information of the quadrotor physical parameters is required. For instance, it will be inconvenient and in some cases imprecise to measure the mass value [137] of the quadrotor together with its payload during each flight. For this purpose, we test the system robustness when the mass varies up to almost a factor of 2 during the flight.

- For the simulations, we have tested our approach with a realistic quadrotor model that contains gyroscopic and aerodynamic effects.

Recent studies have shown the interest in dividing the control problem into two parts: one based on the model and the other based on the model-free control to cope with model uncertainties and/or external perturbations. In [138], model-free sliding-mode control (based on a linearized model) is studied, in [142] a model-free backstepping control (based on a linearized model) and an LQR control have been experimentally tested on a real system. In these studies, the control law is still partially linked to the system model knowledge.
In several works, the yaw angle is assumed to be zero due to high nonlinear coupling in position and attitude dynamics. In [139], an event-driven model-free control is compared with several other controls while taking zero reference for the yaw angle. This limits the maneuverability of quadrotors. In our case we show that the quadrotor can follow aggressive maneuvers while tracking the yaw angle ψ. This is a huge advantage with clear benefits on time and energy. The position-yaw tracking as mentioned in [121] can be useful in different use-cases *e.g.* Aerial Screwdriver task (the quadrotor should turn a screw), the Aerial Grasp (multiple quadrotors rotate their ψ-angle to grasp an object), etc.

Outline

This chapter has the following outline:

- In Section 4.2, we briefly describe the quadrotor dynamics focusing on its structure.
- Then, in Section 4.3, we review the Model-Free Control and we present our cascaded Model-Free approach.
- In Section 4.4, we discuss the practical stability analysis of the proposed approach.
- Finally, we validate our approach in Section 4.8 by testing it on three scenarios when the quadrotor is faced with unknown measurement noise, with unknown time-varying wind disturbances and mass value variation. The simulation results confirm its robustness while tracking two different types of aggressive position trajectory in the three chosen scenarios.

A part from this chapter has been published in [12].

4.2 Quadrotor model

Th quadrotor system has six degrees of freedom, position motion in three directions and rotational motion around three axes, but it has only four actual inputs. Hence it is an underactuated system and all of its motions are dependent of its attitude. The schematic configuration of a quadrotor we adopted in this study is shown in Figure 4.1 that includes the corresponding forces, angles and angular speeds.

A nonlinear model of quadrotor based on the Newton-Euler formalism (see the paper [82] for further information on the quadrotor model) is given by the following equations:

- Position dynamics

$$m\ddot{x} = (\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi)u_1 - A_x\dot{x}, \tag{4.1a}$$

$$m\ddot{y} = (-\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi)u_1 - A_y\dot{y}, \tag{4.1b}$$

$$m\ddot{z} = -mg + \cos\theta\cos\phi u_1 - A_z\dot{z}, \tag{4.1c}$$

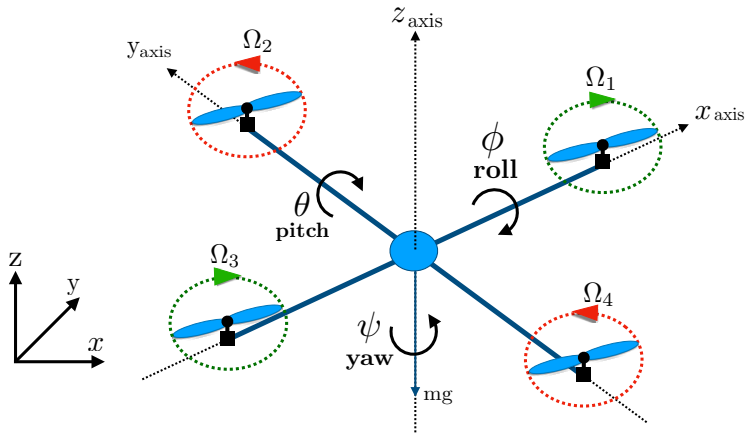


Figure 4.1: The quadrotor system.

- Attitude dynamics

$$I_x\ddot{\phi} = u_2 + \dot{\theta}\dot{\psi}(I_y - I_z) + J_r\dot{\theta}\Omega_r, \tag{4.2a}$$
$$I_y\ddot{\theta} = u_3 + \dot{\phi}\dot{\psi}(I_z - I_x) - J_r\dot{\phi}\Omega_r, \tag{4.2b}$$
$$I_z\ddot{\psi} = u_4 + \dot{\theta}\dot{\phi}(I_x - I_y), \tag{4.2c}$$

where x, y and z are the position coordinates of the quadrotor's center of gravity, and θ, ϕ and ψ are the pitch, roll and yaw rotation angles respectively. The constant m is the mass, g is the gravitation acceleration, I_x, I_y, I_z are the moments of inertia, and J_r is the moment of inertia of the rotors. The controllers are: u_1 the *total thrust* generated by the four propellers applied in the z direction; u_2, u_3 and u_4 *the torques* in the θ, ϕ and ψ directions respectively. The constants A_x, A_y and A_z are the drag force coefficients for velocities in the corresponding directions of the inertial frame. The *position dynamic model* (4.1a)-(4.1c) contains the gravity force mg, the thrust force in the z direction and the drag force. The *attitude dynamic model* (4.2a)-(4.2c) describing the roll, pitch and yaw rotations contains three terms which are the actuators action, the gyroscopic effect resulting from the rigid body rotation, and finally the gyroscopic effect resulting from the propeller rotation coupled with the body rotation.

The control inputs u_1, u_2, u_3, u_4, and the speed Ω_r are defined as:

$$\begin{aligned} u_1 &= b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2), \\ u_2 &= b(\Omega_4^2 - \Omega_2^2), \\ u_3 &= b(\Omega_3^2 - \Omega_1^2), \\ u_4 &= d(\Omega_1^2 + \Omega_3^2 - \Omega_2^2 - \Omega_4^2), \\ \Omega_r &= \Omega_1 - \Omega_2 + \Omega_3 - \Omega_4 \end{aligned} \tag{4.3}$$

Symbols and values	Variables
$m = 0.53\text{kg}$	Quadrotor mass
$I_x, I_y = 6.228 \times 10^{-3}\text{kg} \cdot \text{m}^2$	Inertia parameters
$I_z = 1.121 \times 10^{-2}\text{kg} \cdot \text{m}^2$	Inertia parameter
$J_r = 6.01 \times 10^{-5}\text{kg} \cdot \text{m}^2$	Rotor inertia moment
$b = 3.13 \times 10^{-5}\text{N} \cdot \text{s}^2$	Thrust coefficient
$d = 7.5 \times 10^{-7}\text{Nm} \cdot \text{s}^2$	Drag coefficient
$A_x, A_y, A_z = 0.25\text{kg} / \text{s}$	Drag force coefficients

Table 4.1: The parameters and their corresponding values for the quadrotor simulation.

where Ω_i is the angular speed of the ith rotor, and b and d are the thrust and the drag coefficients respectively.

This model will be used for the simulation. Table 4.1 lists the parameter values used for the simulation model in Section 4.8.

Remark 4.1 *The aerodynamic effects are difficult to model. Some have significant impact only in high velocities. Note that several other aerodynamic effects could be included in the model,* e.g. *the dependence of thrust on angle of attack, blade flapping and airflow disruptions that have been studied in [71] and [72].*

4.3 Control design

In this Section, we first summarize some of the main ideas on Model-Free Control (MFC) introduced in [48]. Then, we construct the cascaded-model-free controller for the quadrotor using only a minimal nominal dynamics.

4.3.1 Preliminaries for Model-Free Control

The unknown differential equation describing the input/output behavior of a finite-dimensional system with a single control variable u and a single output variable y

$$\begin{aligned}&\mathbf{E}(y, \dot{y}, \ldots, y^{(\iota)}, u, \dot{u}, \ldots, u^{(\kappa)}) = 0,\\ &\mathbf{E} : \mathbb{R}^{\iota+1} \times \mathbb{R}^{\kappa+1} \to \mathbb{R}, \mathbf{E} \in \mathbf{C}^{\infty}(\mathbb{R}^{\iota+1} \times \mathbb{R}^{\kappa+1})\end{aligned}$$

can be described as:

$$y^{(\nu)} = \mathbf{E}(t, y, \dot{y}, \ldots, y^{(\nu-1)}, y^{(\nu+1)}, \ldots, y^{(\iota)}u, \dot{u}, \ldots, u^{(\kappa)}), \tag{4.4}$$

where $0 < \nu \leqslant \iota, \dfrac{\partial E}{\partial y^{(\nu)}} \neq 0$. For simplicity, the Equation (4.4) can be represented in short time interval by an *ultra-local model* as:

$$y^{(\nu)} = F + \alpha u \tag{4.5}$$

where

- $\alpha \in \mathbb{R}$ is a non-physical constant parameter chosen such that αu and $y^{(\nu)}$ will be of the same order of magnitude. In the recent study [37], α is considered as a time-varying parameter in order to overcome the presence of unknown delays.
- the time-varying function $F(t)$ is approximated by a piecewise constant function. It represents the rest of the unmodelled dynamics in the input-output behaviour of the system and the unknown endogenous and exogenous disturbances. Therefore, it adapts to the changes of the system at each actuation step.

Obtaining a good estimate of $\hat{F}$ can be achieved by considering it as ν^{th} iterated integral on a short time interval $[t-T, t]$ (see [52]). For instance, when $\nu = 2$, we can rewrite the equivalent of the equation (4.5) in the Laplace domain as:

$$s^2Y(s) - sy(0) - \dot{y}(0) = \frac{F}{s} + \alpha U(s) \tag{4.6}$$

We get rid of the initial conditions $y(0)$ and $\dot{y}(0)$ by differentiating twice the equation (4.6) w.r.t. to the Laplace variable s.

$$2Y(s) + 4s\frac{dY}{ds} + s^2\frac{d^2Y}{ds^2} = \frac{2F}{s^3} + \alpha\frac{d^2U}{ds^2} \tag{4.7}$$

To greatly attenuate the noise, we multiply the both sides of equation (4.7) by s^{-3}

$$\frac{2}{s^3}Y(s) + 4\frac{1}{s^2}\frac{dY}{ds} + \frac{1}{s}\frac{d^2Y}{ds^2} = \frac{2F}{s^6} + \frac{\alpha}{s^3}\frac{d^2U}{ds^2} \tag{4.8}$$

In the time domain, we get

$$\hat{F} = \frac{5!}{2T^5}\int_{t-T}^{t}\Big((M^2 - 4\sigma M + \sigma^2)y(\sigma) - \frac{\alpha}{2}M^2\sigma^2u(\sigma)\Big)d\sigma, \tag{4.9}$$

where $M = T - \sigma$. The choice of the window T results in a trade-off. The larger is T, the smaller is the effect of noise and the larger is the error due to truncation. For a precise mathematical foundation on the treatment of the noise through iterated time integrals, see [46], which is based on non-standard analysis, and also [90] which

rests on Jacobi polynomials. The closed-loop control applied to the *ultra-local model* (4.5) is defined as the so-called *intelligent controller*

$$u = -\frac{\hat{F} - y_d^{(2)} + \mathfrak{C}(e)}{\alpha} \tag{4.10}$$

where y_d is the output reference trajectory, $e = y - y_d$ is the tracking error, and $\mathfrak{C}(e) = K_p e + K_d \dot{e}$ is a PD controller. Combining (4.5) and (4.10), it yields the following closed-loop error dynamics

$$\ddot{e} + K_d \dot{e} + K_p e = e_F = F - \hat{F}. \tag{4.11}$$

If the estimate $\hat{F}$ is good, the error $e_F \simeq 0$ is small and choosing the gains such that $K_p > 0, K_d > 0$ guarantees a good tracking of y_d.

4.3.2 Cascaded-model-free approach for the quadrotor

Problem statement: We want to ensure that the quadrotor tracks the desired time-varying position trajectory (x_d, y_d, z_d) and yaw angle ψ_d without precise information of the physical parameters and forces (for instance, the mass, the inertias and the aerodynamic forces) and despite external disturbances. We propose a cascaded model-free setting of the quadrotor based on a minimal dynamics (see Equations (4.12), (4.14), (4.19a)-(4.19c)).

Inspired by the quadrotor dynamics structure, we divide the control in two stages (see Figure 4.2): the outer loop with slow dynamics which controls the position, and the inner loop with fast dynamics which controls the attitude. From the quadrotor structure, we observe that the position dynamics depends on the attitude dynamics.

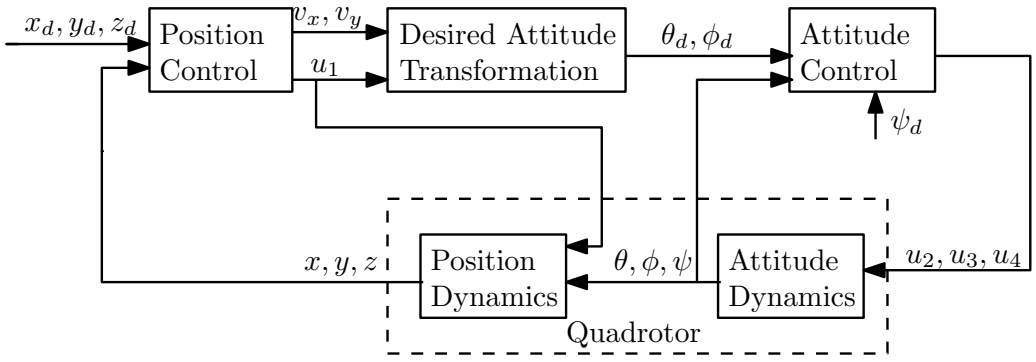


Figure 4.2: Cascaded control overview (The subscript d denotes desired trajectory).

4.3.3 Outer-loop Position control

The model-free setting of the vertical z-dynamics (4.1c) is given by

$$\ddot{z} = F_z + \alpha_z u_1 \tag{4.12}$$

where F_z represents the neglected dynamics and the external disturbances, and α_z is a constant parameter. To estimate the time-varying F_z over a time interval of length T, we use Equation (4.9) yielding

$$\hat{F}_z = \frac{5!}{2T^5}\int_{t-T}^{t}\Big((M^2 - 4\sigma M + \sigma^2)z(\sigma) - \frac{\alpha}{2}M^2\sigma^2 u_1(\sigma)\Big)d\sigma, \quad M = T - \sigma.$$

We then set the control law for the thrust u_1 as:

$$u_1 = \frac{1}{\alpha_z}(-\hat{F}_z + \ddot{z}_d - K_{pz}e_z - K_{dz}\dot{e}_z) \tag{4.13}$$

where $e_z = z - z_d$ and $\dot{e}_z = \dot{z} - \dot{z}_d$ are the tracking errors of the vertical position and velocity respectively. To operate safely the quadrotor requires a positive thrust $u_1 > 0$.

The model-free setting of the xy-dynamics (4.1a)-(4.1b) is given by

$$\begin{pmatrix}\ddot{x}\\ \ddot{y}\end{pmatrix} = \begin{pmatrix}F_x\\ F_y\end{pmatrix} + \alpha_{xy}R_\psi \begin{pmatrix}\sin\phi\\ \sin\theta\cos\phi\end{pmatrix} u_1. \tag{4.14}$$

where $\alpha_{xy} = \begin{pmatrix}\alpha_x & 0\\ 0 & \alpha_y\end{pmatrix}$ are constants, and F_x and F_y are the neglected dynamics and disturbances for x and y respectively. The rotation matrix $R_\psi \in SO(2)$ is given by

$$R_\psi = \begin{pmatrix}\sin\psi & \cos\psi\\ -\cos\psi & \sin\psi\end{pmatrix}, R_\psi^{-1} = R_\psi^T, \quad \|R_\psi\| = 1.$$

Here, $\|\cdot\|$ denotes the Euclidean norm.

Setting a new virtual input $(v_x, v_y)^T$ defined as:

$$\begin{pmatrix}\ddot{x}\\ \ddot{y}\end{pmatrix} = \begin{pmatrix}v_x\\ v_y\end{pmatrix} = R_\psi \begin{pmatrix}\sin\phi_d\\ \sin\theta_d\cos\phi_d\end{pmatrix} u_1, \tag{4.15}$$

we obtain

$$v_x = \frac{1}{\alpha_x}(-\hat{F}_x + \ddot{x}_d - K_{px}e_x - K_{dx}\dot{e}_x), \tag{4.16a}$$

$$v_y = \frac{1}{\alpha_y}(-\hat{F}_y + \ddot{y}_d - K_{py}e_y - K_{dy}\dot{e}_y). \tag{4.16b}$$

where $e_x = x - x_d$ and $e_y = y - y_d$ are the tracking errors for x and y respectively. From there, we can deduce the reference trajectories θ_d and ϕ_d for the attitude

dynamics by expressing the virtual input measures v_x and v_y in the body frame where $\psi = 0$ (no yaw dependence) by defining $\bar{v}_x$ and $\bar{v}_y$ as:

$$\begin{pmatrix} \sin\phi_d \\ \sin\theta_d \cos\phi_d \end{pmatrix} = \frac{1}{u_1} R_\psi^{-1} \begin{pmatrix} v_x \\ v_y \end{pmatrix} \triangleq \begin{pmatrix} \bar{v}_x \\ \bar{v}_y \end{pmatrix} \tag{4.17}$$

Hence we consider the following desired angles ϕ_d and θ_d:

$$\phi_d = \arcsin(\bar{v}_x), \tag{4.18a}$$

$$\theta_d = \arcsin\left(\frac{\bar{v}_y}{\cos\phi_d}\right). \tag{4.18b}$$

4.3.4 Inner-loop Attitude control

The attitude dynamics (4.2a)-(4.2c) is given by the following:

$$\ddot{\phi} = F_\phi + \alpha_\phi u_2, \tag{4.19a}$$

$$\ddot{\theta} = F_\theta + \alpha_\theta u_3, \tag{4.19b}$$

$$\ddot{\psi} = F_\psi + \alpha_\psi u_4 \tag{4.19c}$$

The attitude control performance is crucial since it is directly related to the actuators efficiency. After deducing the desired attitude ϕ_d, θ_d from the slow outer loop (4.18a)-(4.18b), the fast inner loop is stabilized by an MFC controller as

$$u_2 = \frac{1}{\alpha_\phi}(-\hat{F}_\phi + \ddot{\phi}_d - K_{p\phi} e_\phi - K_{d\phi}\dot{e}_\phi), \tag{4.20}$$

$$u_3 = \frac{1}{\alpha_\theta}(-\hat{F}_\theta + \ddot{\theta}_d - K_{p\theta} e_\theta - K_{d\theta}\dot{e}_\theta), \tag{4.21}$$

where $e_\phi = \phi - \phi_d, e_\theta = \theta - \theta_d$, $\dot{e}_\phi = \dot{\phi} - \dot{\phi}_d, \dot{e}_\theta = \dot{\theta} - \dot{\theta}_d$ are the tracking errors of the angular positions and estimated angular velocities respectively. Similarly, for the yaw angle, we apply the Model-Free control as

$$u_4 = \frac{1}{\alpha_\psi}(-\hat{F}_\psi + \ddot{\psi}_d - K_{p\psi} e_\psi - K_{d\psi}\dot{e}_\psi). \tag{4.22}$$

4.4 Practical stability

In this section, the closed-loop practical stability of our scheme is studied. First, we define the position error dynamics and the attitude error dynamics separately, and we recognize a *cascade structure* with an interconnection term which is dependent on the attitude subsystem. Without considering the interconnection term, the control laws from the previous section stabilize asymptotically each of the error

dynamics subsystems. By means of cascaded systems results and the obtained bound of the interconnection term, the complete system is *practically stable* when the convergence errors are bounded.

We rewrite the quasi-model-free quadrotor setting in a more compact form as

$$\ddot{p} = F_{xy} + \alpha_{xy} R_\psi \gamma(\phi, \theta) u_1, \tag{4.23a}$$

$$\ddot{z} = F_z + \alpha_z u_1, \tag{4.23b}$$

$$\ddot{r} = F_r + \alpha_r u_{23}, \tag{4.23c}$$

$$\ddot{\psi} = F_\psi + \alpha_\psi u_4, \tag{4.23d}$$

where $p = (x, y)^T$, $F_{xy} = (F_x, F_y)^T$, $\gamma(\phi, \theta) = (\sin\phi, \sin\theta\cos\phi)^T$, $r = (\phi, \theta)^T$, $F_r = (F_\phi, F_\theta)^T$, $\alpha_r = (\alpha_\phi, \alpha_\theta)^T$and $u_{23} = (u_2, u_3)^T$.

4.4.1 The system error dynamics

We define the *position tracking error* as $X_p = (X_{1p}, X_{2p})$ where $X_{1p} = (p - p_d, z - z_d)^T$, $X_{2p} = (\dot{p} - \dot{p}_d, \dot{z} - \dot{z}_d)^T \in \mathbb{R}^3$ and the *attitude tracking error* as $X_r = (X_{1r}, X_{2r})$ where $X_{1r} = (r - r_d, \psi - \psi_d)^T$, $X_{2r} = (\dot{r} - \dot{r}_d, \dot{\psi} - \dot{\psi}_d)^T \in \mathbb{R}^3$.

By replacing the rotation angles $(\phi, \theta)^T$ with $(\phi_d + e_\phi, \theta_d + e_\theta)^T$ and by means of the trigonometric relations, we compute the following expressions of $\gamma(\phi, \theta)$:

$$\begin{aligned} \gamma(\phi, \theta) &= \begin{pmatrix} \sin(\phi_d + e_\phi) \\ \sin(\theta_d + e_\theta)\cos(\phi_d + e_\phi) \end{pmatrix} \\ &= \gamma(\phi_d, \theta_d) + \begin{pmatrix} \sin(e_\phi/2)\cos(\phi_d + e_\phi/2) \\ -\sin(\theta_d)a_y + \cos(\phi_d)a_x - a_x a_y \end{pmatrix} \\ &= \gamma(\phi_d, \theta_d) + \Delta(e_\phi, e_\theta, r_d) \end{aligned} \tag{4.24}$$

where

$$\begin{aligned} a_x &= \sin(e_\theta/2)\cos(\theta_d + e_\theta/2) \\ a_y &= \sin(e_\phi/2)\sin(\phi_d + e_\phi/2). \end{aligned}$$

We can thus rewrite (4.23a) as:

$$\begin{aligned} \ddot{p} &= F_{xy} + \alpha_{xy} R_\psi u_1 (\gamma(\phi_d, \theta_d) + \Delta(e_\phi, e_\theta, r_d)) \\ &= F_{xy} + \alpha_{xy} v + \alpha_{xy} R_\psi u_1 \Delta(e_\phi, e_\theta, r_d) \end{aligned} \tag{4.25}$$

where $R_\psi u_1 \Delta(e_\phi, e_\theta, r_d)$ is the interconnection term.

We can define the *system error dynamics* as:

$$\dot{X}_p = \begin{pmatrix} X_{2p} \\ \dot{X}_{2p} \end{pmatrix}, \tag{4.26}$$

$$\dot{X}_r = \begin{pmatrix} X_{2r} \\ \dot{X}_{2r} \end{pmatrix}, \tag{4.27}$$

where

$$\begin{aligned} \dot{X}_{2p} = \begin{pmatrix} \ddot{p} - \ddot{p}_d \\ \ddot{z} - \ddot{z}_d \end{pmatrix} &= \begin{pmatrix} F_{xy} + \alpha_{xy} R_\psi u_1 - \ddot{p}_d \\ F_z + \alpha_z u_1 - \ddot{z}_d \end{pmatrix} \\ &= \begin{pmatrix} u_{xy} \\ u_1 \end{pmatrix} + \begin{pmatrix} \alpha_{xy}(R_\psi u_1 \Delta(e_\phi, e_\theta, r_d) - v) \\ 0 \end{pmatrix}. \end{aligned} \tag{4.28}$$

4.5 Stability results

Let $u_p = (u_{xy}, u_1)$, $F_p = (F_{xy}, F_z)$. Let $d = (\ddot{p}_d, \psi_d)$ be the desired trajectories. We define the error dynamics system as a cascaded system

$$\dot{X}_p = AX_p + B \begin{pmatrix} u_{xy} \\ u_1 \end{pmatrix} + b(X_p, X_r, d) \tag{4.29}$$

$$\dot{X}_r = AX_r + B \begin{pmatrix} u_r \\ u_\psi \end{pmatrix} \tag{4.30}$$

where $A = \begin{pmatrix} 0_{3\times3} & I_{3\times3} \\ 0_{3\times3} & 0_{3\times3} \end{pmatrix}$, $B = \begin{pmatrix} 0_{3\times3} \\ I_{3\times3} \end{pmatrix}$ and the interconnection term $b(X_p, X_r, d) = R_\psi u_1 \Delta(e_\phi, e_\theta, r_d)$.

Assumption 4.1 *(Subsystem Globally Asymptotically Stable (GAS)) The equilibrium $X_p = 0$ of $\dot{X}_p = f(X_p, u_p(X_p, \hat{F}_p)$ is GAS and the equilibrium $X_r = 0$ of $\dot{X}_r = g(X_r, u_r(X_r, \hat{F}_r)$ is GAS.*

Theorem 4.1 *(Attractivity) Given the Assumption 4.1, every solution $(X_p(t), X_r(t))$ either converges to $(X_p, X_r) = (0, 0)$ or is unbounded.*

Assumption 4.2 *(Boundedness) The trajectories $X_p(t, X_{p0})$ are bounded for all initial conditions X_{p0}.*

Assumption 4.3 *(Interconnection growth restriction) For two class K functions $\gamma_1(\cdot)$ and $\gamma_2(\cdot)$ differentiable at $\|X_r\| = 0$ it holds that*

$$\|b(X_p, X_r, d)\| \leqslant \gamma_1(\|X_r\|)\, \|X_p\| + \gamma_2(\|X_r\|) \tag{4.31}$$

i.e. *the function $\|b(X_p, X_r, d)\|$ has at most linear growth in X_p.*

Theorem 4.2 *(GAS) If Assumptions 4.1-4.3 hold, the zero equilibrium* $(X_p, X_r) = (0, 0)$ *of the cascade is GAS.*

Assumption 4.4 *(Local Exponential stability): The Jacobian linearization* (A, B) *of* $\dot{X}_r = g(X_r, u_r(X_r))$ *at* $X_r = 0$ *is stable.*

Assumption 4.5 *(Lyapunov function growth restriction): There exists a positive semidefinite radially unbounded function* $V(X_p)$ *and two positive constants* c_2 *and* c_3 *such that for* $\|X_p\| \geq c_2$

$$\begin{cases} \dfrac{\partial V}{\partial X_p} f(X_p, u_p(X_p, \ddot{p}_d), \ddot{p}_d) \leqslant 0 \\ \left\| \dfrac{\partial V}{\partial X_p} \right\| \|X_p\| \leqslant c_3 V(X_p) \end{cases} \tag{4.32}$$

Assumption 4.6 *Let* $e_{F_*} = F_* - \hat{F}_* \in \mathcal{L}_\infty$, $* = p, z, r, \psi$ *be bounded such that*

$$\|e_{F_*}\|_\infty \leqslant B_{F_*} \tag{4.33}$$

where B_{F_*} *are a known constants.*

Theorem 4.3 *(Boundedness) If Assumptions 4.1 and 4.3-4.5, 4.6 hold, the solutions of the cascaded error system* (4.29)-(4.30) *converge towards a small vicinity of the origin for all the initial conditions.*

The proof of the Theorems 4.1, 4.2 can be found in [39]. Under these Assumptions, the boundedness of the states is proven in [123] (Theorem 4.7). In the next section, we verify that the Assumptions hold for our approach.

4.6 Stability proof

The stability of the connected system (4.1a)-(4.2c) will be ensured if we choose stabilizing feedbacks as in Section 4.3 and prove that all the error trajectories (X_p, X_r) and the interconnection trajectories are bounded.

4.6.1 Position error subsystem

By replacing the control (4.13)-(4.16) in (4.26), the closed-loop position subsystem without the interconnection term Δ is given by:

$$\dot{X}_{2p} + \Gamma_p X_{1p} + \Gamma_d X_{2p} = \begin{pmatrix} e_{F_p} \\ e_{F_z} \end{pmatrix} \tag{4.34}$$

where $\Gamma_p = \begin{pmatrix} K_{px} & 0 & 0 \\ 0 & K_{py} & 0 \\ 0 & 0 & K_{pz} \end{pmatrix}$ and $\Gamma_d = \begin{pmatrix} K_{dx} & 0 & 0 \\ 0 & K_{dy} & 0 \\ 0 & 0 & K_{dz} \end{pmatrix}$.
$A_p = \begin{pmatrix} 0_{3\times3} & I_{3\times3} \\ -\Gamma_p & -\Gamma_d \end{pmatrix}$. We rewrite the position dynamics as

$$\dot{X}_p = A_p X_p + \begin{pmatrix} e_{F_p} \\ e_{F_z} \end{pmatrix} \tag{4.35}$$

where A_p is a Hurwitz matrix.

4.6.2 Attitude error subsystem

By replacing the controls (4.20)-(4.22) into (4.27), the closed-loop attitude subsystem is given by:

$$\dot{X}_{2r} + \Lambda_p X_{1r} + \Lambda_d X_{2r} = \begin{pmatrix} e_{F_r} \\ e_{F_\psi} \end{pmatrix} \tag{4.36}$$

where $\Lambda_p = \begin{pmatrix} K_{p\theta} & 0 & 0 \\ 0 & K_{p\phi} & 0 \\ 0 & 0 & K_{p\psi} \end{pmatrix}$ and $\Lambda_d = \begin{pmatrix} K_{d\theta} & 0 & 0 \\ 0 & K_{d\phi} & 0 \\ 0 & 0 & K_{d\psi} \end{pmatrix}$.
$A_r = \begin{pmatrix} 0_{3\times3} & I_{3\times3} \\ -\Lambda_p & -\Lambda_d \end{pmatrix}$. We rewrite the attitude dynamics as:

$$\dot{X}_r = A_r X_r + \begin{pmatrix} e_{F_r} \\ e_{F_\psi} \end{pmatrix} \tag{4.37}$$

where A_r is a Hurwitz matrix.

4.6.3 Verification of the assumptions

Assumptions 4.1 and 4.4 hold by choosing the matrices A_p and A_r as Hurwitz matrices. Since A_p is a Hurwitz matrix, for the nominal system $\dot{X}_p = A_p X_p$ there exist two matrices $P = P^T > 0, Q > 0 \in \mathbb{R}^{6\times6}$ such that the Lyapunov function $V(X_p) = X_p^T P X_p$ is positive definite and

$$\frac{\partial V}{\partial X_p} f = -X_p^T Q X_p \leqslant 0 \tag{4.38}$$

This satisfies the Assumption 4.5.

The Assumption 4.3 requires that the interconnection term can be upper bounded by a function with at most linear growth at X_p. Let $X_z = (e_z, \dot{e}_z)$. For $\|u_1\|$, we have:

$$\begin{aligned}\|u_1\| &= \frac{1}{\alpha_z}\left\|(-\hat{F}_z + \ddot{z}_d - K_{pz}e_z - K_{dz}\dot{e}_z)\right\| \\ &\leqslant \frac{1}{\alpha_z}(\left\|\hat{F}_z\right\| + \|\ddot{z}_d\|_\infty + K_{pz}\|e_z\| + K_{dz}\|\dot{e}_z)\|) \\ &\leqslant \frac{1}{\alpha_z}(B_{F_z} + \|\ddot{z}_d\|_\infty) + \frac{1}{\alpha_z}(K_{pz}\|e_z\| + K_{dz}\|\dot{e}_z)\|)) \\ &\leqslant m_1 + m_2\|X_z\|\end{aligned} \tag{4.39}$$

where $m_1 = \frac{1}{\alpha_z}(B_{F_z} + \|\ddot{z}_d\|_\infty)$ and $m_2 = \frac{1}{\alpha_z}\max(K_{pz}, K_{dz})\sqrt{2}$ are positive constants. We choose a reference trajectory such that $\|\ddot{z}_d\|_\infty$ is bounded. For sake of completeness, the bound of $\Delta(e_\phi, e_\theta, r_d)$ is given in the Appendix 4.A, p. 109, where we show that

$$\Delta(e_\phi, e_\theta, r_d) \leqslant m_3\|X_r\|. \tag{4.40}$$

Hence, the Assumption 4.3 holds such that:

$$\begin{aligned}\|b(X_p, X_r, d)\| &= \|R_\psi u_1 \Delta(e_\phi, e_\theta, r_d)\| \\ &\leqslant \|u_1\|\,\|\Delta(e_\phi, e_\theta, r_d)\| \\ &\leqslant (m_1 + m_2\|X_z\|)m_3\|X_r\|.\end{aligned} \tag{4.41}$$

If a small time interval h with respect to system dynamics is selected, as well as a small filtering time constant T, the bound on e_{F_i} will be small (Appendix 4.B, p. 110). The tracking error e will remain in a bounded ball and the system is *practical stable* (see Assumption 4.6).

4.7 Aggressive trajectory tracking

Usually, for simplicity, the quadrotor tracks a straight line or circular orbit. In this Section, we present the two types of agile trajectories that the quadrotor will follow in three different scenarios:

- unknown measurement noise,
- unknown time-varying wind disturbances, and
- mass value variation.

4.7.1 The Lissajous trajectory

We consider here an aggressive motion: a Lissajous octave curve in the x_d, y_d plane and an hyperbolic tangent for z_d and ψ_d.

$$\begin{aligned} x_d(t) &= A\sin(at+\delta) \\ y_d(t) &= B\sin(bt) \end{aligned} \tag{4.42}$$

where we take $A = 1, B = 1.1, a = 1, b = 2$ and $\delta = 1.7321$. The ratio $\frac{a}{b}$ determines the form of the curve. We specialize the state z_d to a sigmoid between two quasi constant altitudes, a situation frequently needed in practice.

$$z_d(t) = \frac{H_f - H_i}{2}\left(1 + \tanh(\gamma(t - t_m))\right) + H_i \tag{4.43}$$

where H_i is the initial altitude and H_f is the final altitude of the quadrotor; γ is the slope parameter of the sigmoid and t_m is the time when the quadrotor is taking off.

Remark 4.2 *By taking reference trajectories as in Section 4.8, we have explicitly the bounds of their derivatives. For z and ψ:The easy numerical implementation of the derivatives of $z(t)$ is due to the nice recursion. Let $R = \tanh(\gamma(t - t_m))$ and $C = \frac{H_f - H_i}{2}$. The first two derivatives of $z_d(t)$ are:*

$$\begin{aligned} \dot{z}_d &= \gamma C(1 - R^2) \\ \ddot{z}_d &= -2\gamma^2 CR(1 - R^2). \end{aligned}$$

The maximum values for its derivatives depend only on γ and C, and their values can be determined. We obtain their bounds as:

$$H_i \leqslant z_d \leqslant H_f, \quad 0 \leqslant \dot{z}_d \leqslant \gamma C, \quad \|\ddot{z}_d\| \leqslant \frac{4\sqrt{3}}{9}\gamma^2 C = L_z.$$

For the second derivatives of the Lissajous curves x and y, we have the following bounds:

$$\begin{aligned} \|\ddot{x}_d\| &\leqslant Aa^2, \\ \|\ddot{y}_d\| &\leqslant Bb^2. \end{aligned}$$

We take the same type of reference trajectory for the angle ψ. The aim here is to track the positions x_d, y_d and z_d, and the yaw angle ψ_d. For the control part, we consider the following values for the alphas: $\alpha_x = 1$, $\alpha_y = 1$, $\alpha_z = 2$ and $\alpha_\psi = 90$. The controller runs at 20Hz, and the sensor data is updated at 100Hz. As initial positions and initial angles, we take $(x_0, y_0, z_0) = (0.5, 0, 0)[m]$

and $(\phi_0, \theta_0, \psi_0) = (0, 0, 0)[rad]$ respectively. The initial velocities and initial angular velocities are zero. Table 4.1 lists the parameter values used in the simulation model.

In the nominal case (without endogenous or exogenous disturbances), we observe satisfactory results when the position and the angles change significantly. Figure 4.3 depicts the trajectory tracking of the four outputs. Figure 4.4 depicts the control inputs and Figure 4.5 depicts the estimated unknowns F_i for $i = x, y, z, \psi$. The 3D tracking is plotted in Figure 4.6.

4.7.2 The B-spline trajectory

We choose the B-spline curves (the definition of a B-Spline curve can be found in the Section 3.4) because they are always contained in the convex hull of their control polygon: the basis functions are positive and sum up to one (partition of unity), and have a local support [83]. By increasing the degree of the B-spline curve and/or by inserting extra knots, the distance between the control polygon and the B-spline curve can be reduced.
We consider here an aggressive motion $[x_d(t), y_d(t), z_d(t), \psi_d(t)] : [t_0 \quad t_f] \rightarrow \mathbb{R}^3 \times$

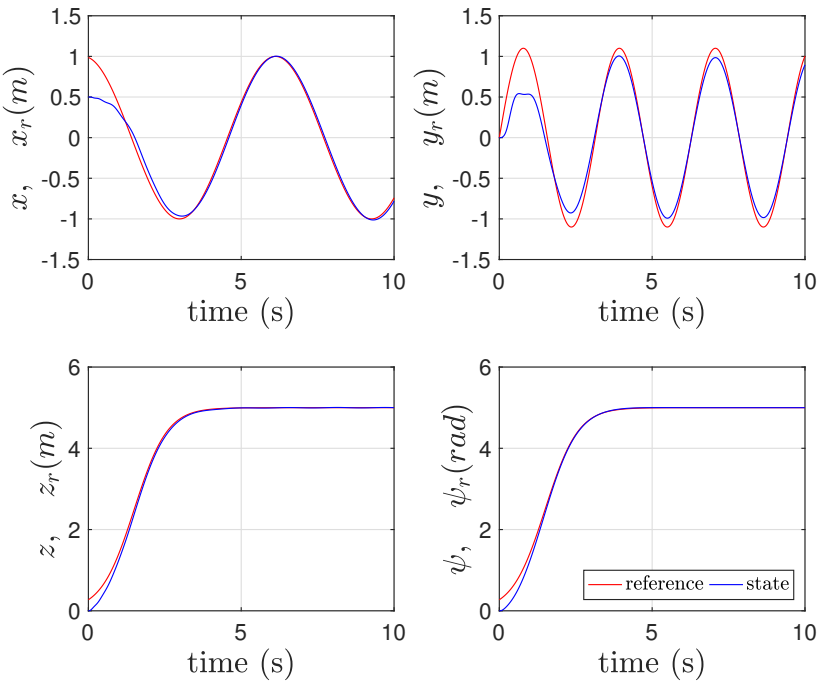


Figure 4.3: Lissajous trajectory case: Reference trajectories and model-free tracking trajectories of the four outputs x, y, z and ψ

$SO(2)$ defined as a B-spline curve:

$$\begin{aligned} x_d(t) &= \sum_{j=1}^{10} a_{xj} B_{j,d}, \\ y_d(t) &= \sum_{j=1}^{10} a_{yj} B_{j,d}, \\ z_d(t) &= \sum_{j=1}^{10} a_{zj} B_{j,d}, \end{aligned} \tag{4.44}$$

with degree $d = 4$, knot vector $\boldsymbol{t} = \{0, 0, 0, 0, 0, 2.5, 5, 7.5, 10, 12.5, 15, 15, 15, 15, 15\}$, and control point vectors $\boldsymbol{A_x} = (a_{xj})_{xj=1}^{10}$, $\boldsymbol{A_y} = (a_{yj})_{yj=1}^{10}$ and $\boldsymbol{A_z} = (a_{zj})_{zj=1}^{10}$ for $x_d(t)$, $y_d(t)$ and $z_d(t)$ respectively.

We specialize the reference angle ψ_d to a sigmoid between two constant angles a situation frequently needed in practice.

$$\psi_d(t) = \frac{\Psi_f - \Psi_i}{2} \left(1 + \tanh(\gamma(t - t_m))\right) + \Psi_i \tag{4.45}$$

where $\Psi_i = 0$rad is the initial ψ-angle and $\Psi_f = 1.5$rad is the final ψ-angle of the quadrotor; $\gamma = 0.4$ is the slope parameter of the sigmoid and $t_m = 7$.

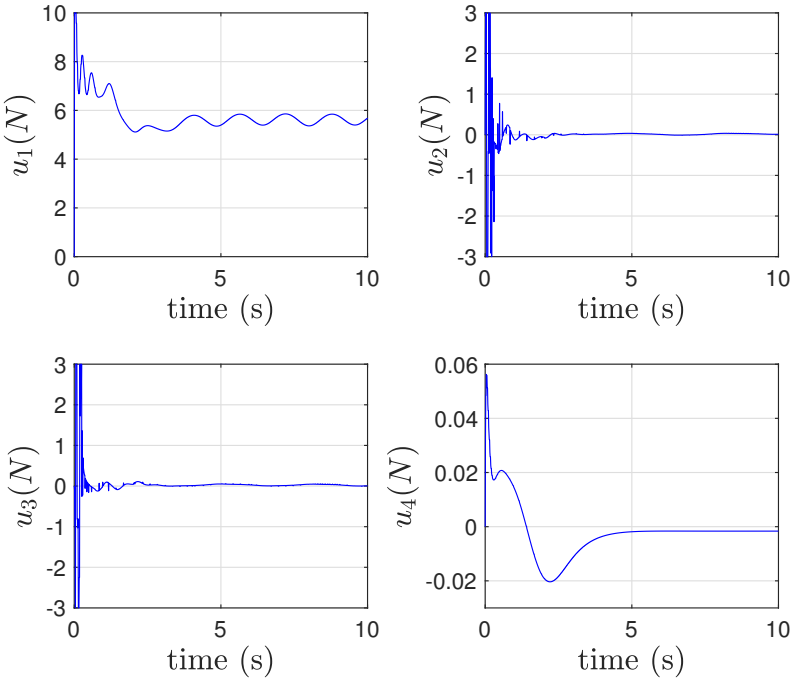


Figure 4.4: Lissajous trajectory case: Control inputs

4.8 Simulation results

To demonstrate the proposed control we consider a realistic quadrotor simulator where we use the quadrotor model defined by the equations (4.1a)-(4.1c), (4.2a)-(4.2c) and (4.3) presented in Section 4.2. Similar to the usual case in practice, the controller runs at 20Hz, and the sensor data is updated at 100Hz. The aim here is to track the positions x_d, y_d and z_d, and the yaw angle ψ_d. As initial positions and initial angles, we take $(x_0, y_0, z_0) = (-1.8, 1, 0)[m]$ and $(\phi_0, \theta_0, \psi_0) = (0, 0, 0)[rad]$ respectively. The initial linear velocities and initial angular velocities are zero. The control point vectors for the reference trajectories are the following:

$$\begin{aligned} \boldsymbol{A_x} &= \{-2, -1, 0, 2, 3, 3.5, 3.5, 5, 6.5, 7.5\}, \\ \boldsymbol{A_y} &= \{1.2, 2.5, 3.3, 1.8, 1.5, 2.5, 4, 4, 4, 4\}, \\ \boldsymbol{A_z} &= \{0, 0.2, 0.5, 2.5, 3.5, 3.9, 4.7, 6.5, 5.7, 3.5\}. \end{aligned} \tag{4.46}$$

Our trajectory is parametrized to avoid known static obstacles. In the *nominal case* (without disturbances), we observe in Figure 4.7 (trajectory tracking), Figure 4.8 (control inputs) and Figure 4.9 (unknown dynamics estimation) satisfactory results when the position and the yaw angle change significantly. The 3D tracking is plotted in Figure 4.10. The control gains that we consider are given in Table

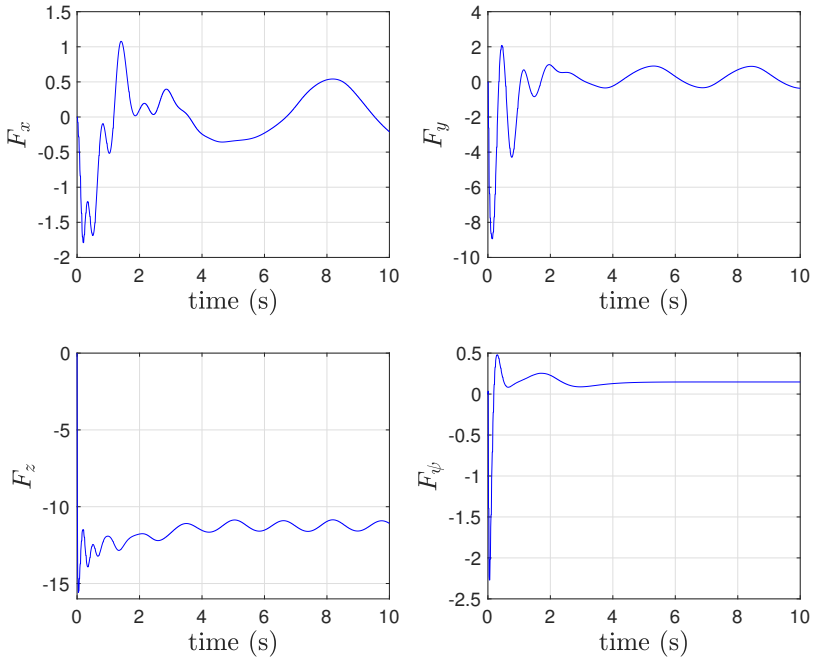


Figure 4.5: Lissajous trajectory case: Estimation of the unknowns F_i for $i = x, y, z, \psi$

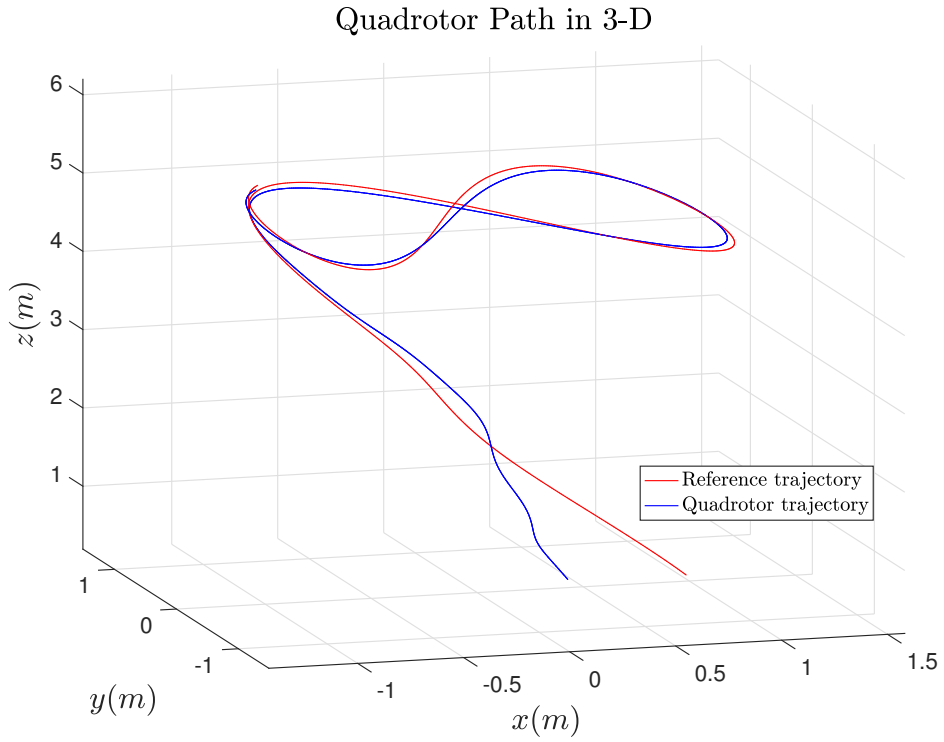


Figure 4.6: 3-D Position tracking of Lissajous path

P	V	P	V	P	V
K_{px}	3	K_{dx}	2	α_x	1
K_{py}	3	K_{dy}	2	α_y	1
K_{pz}	10	K_{dz}	15	α_z	2
$K_{p\phi}$	2	$K_{d\phi}$	0.5	α_ϕ	1
$K_{p\theta}$	1	$K_{d\theta}$	0.5	α_θ	1
$K_{p\psi}$	3	$K_{d\psi}$	2.5	α_ψ	9

Table 4.2: The control gains (P: Parameter, V:Value).

4.2. The values for α_i are chosen from a large possible range each dependant on the corresponding state dynamics.

4.8.1 Scenario 1: Unknown measurement noise

For this scenario, we include the presence of the measurement noise $b(t) \sim \mathcal{N}(0, \sigma^2)$ as an additive white Gaussian noise with zero mean and standard deviation $\sigma = 0.15$ in the four measured outputs x, y, z and ψ. The noisy measured outputs are set as:

$$\begin{aligned} \hat{x} &= x + b, \quad \hat{y} = y + b, \\ \hat{z} &= z + b, \quad \hat{\psi} = \psi + b. \end{aligned} \tag{4.47}$$

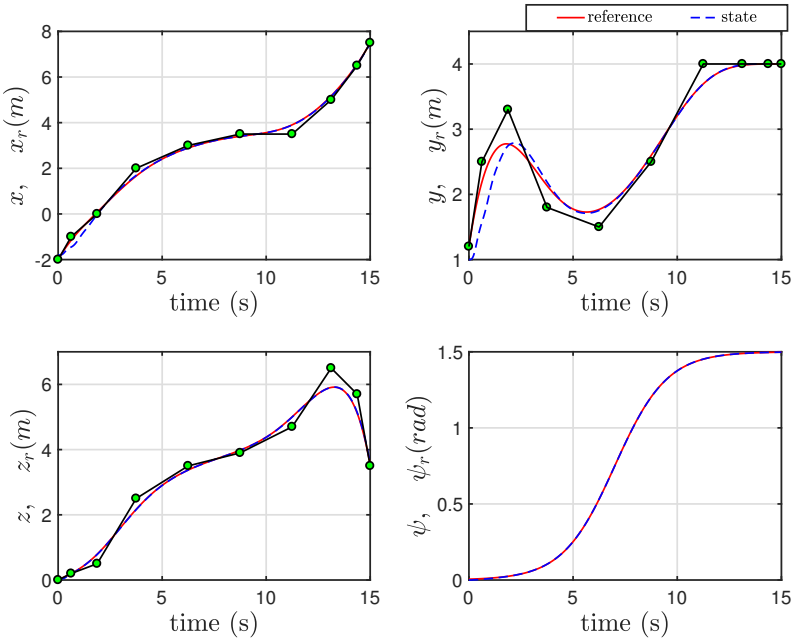


Figure 4.7: B-Spline trajectory case: Reference trajectories and model-free tracking trajectories of the four outputs x, y, z and ψ in the nominal case.

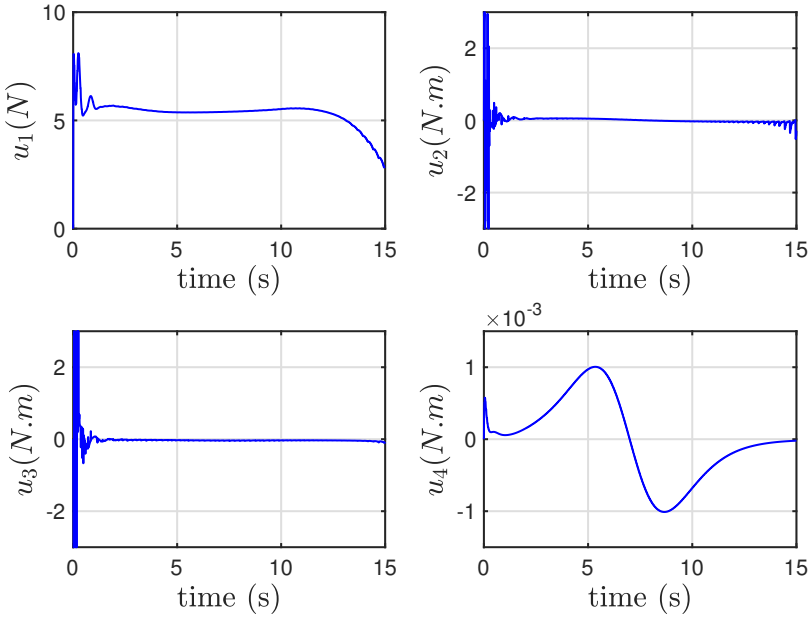


Figure 4.8: B-Spline trajectory case: Control inputs in the nominal case.

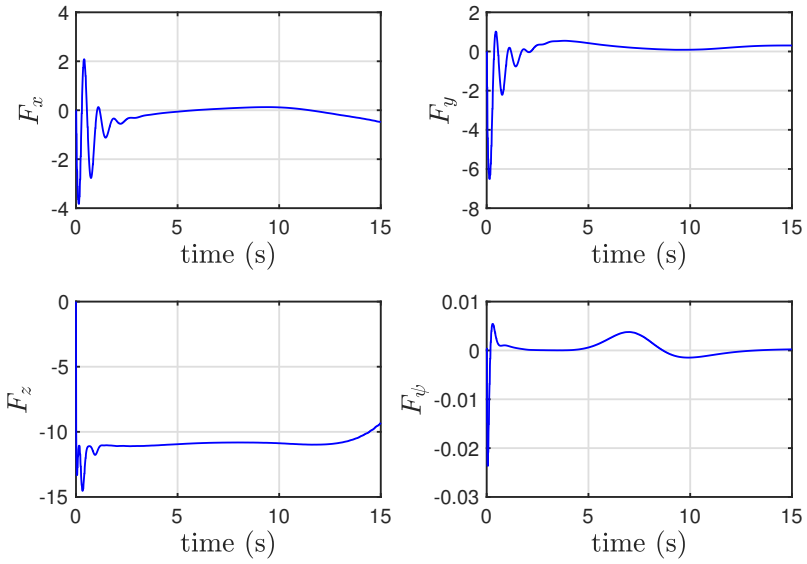


Figure 4.9: B-Spline trajectory case: Estimation of the unknowns F_i for $i = x, y, z, \psi$ in the nominal case.

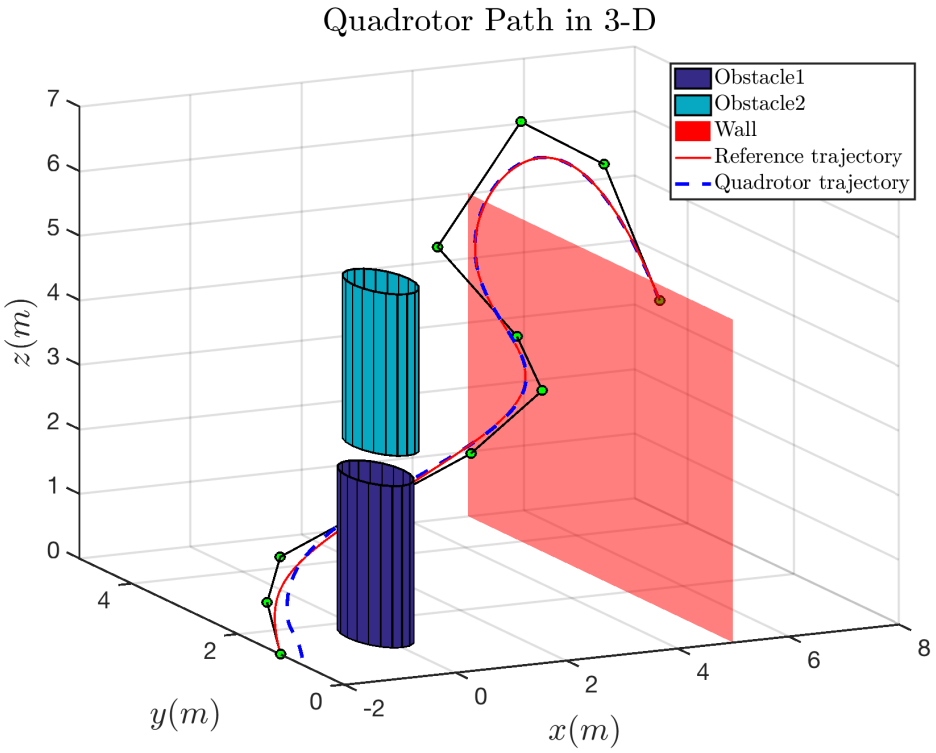


Figure 4.10: 3-D Position tracking of the B-spline path in the nominal case. Objects in the simulation are given just as potential obstacles, but no obstacle avoidance algorithm is yet used.

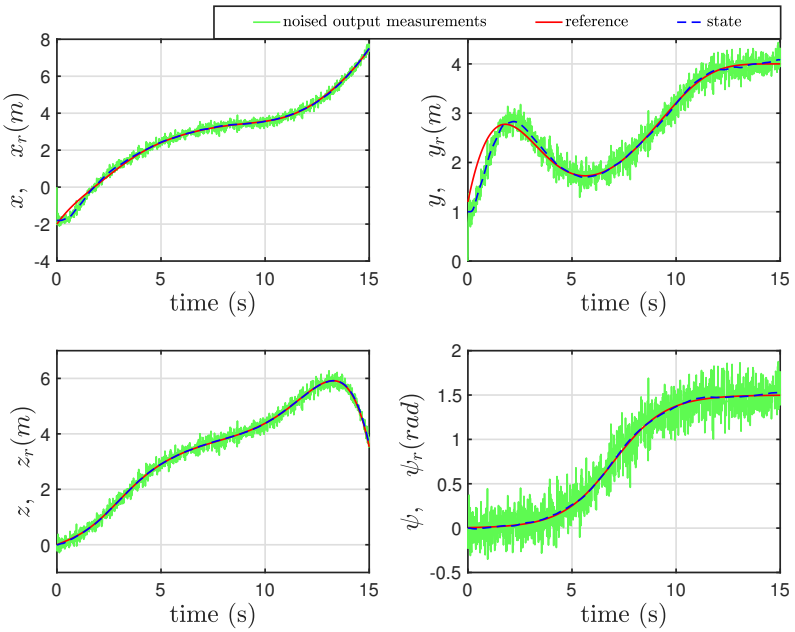


Figure 4.11: B-Spline trajectory case: Tracking trajectories in presence of noise (noisy measurements and real state position).

B-spline trajectory case

Figure 4.11 shows the noisy sensor measurements (green line) and the tracking trajectory of the quadrotor (blue dashed line) that follows the desired trajectory (red line). From the result in Figure 4.11 (trajectory tracking) and Figure 4.12 (control inputs), it is apparent that the control approach is robust to the sensor disturbances without previous knowledge of the noise.

Lissajous trajectory case

Figure 4.13 shows the trajectory tracking for the $x - y$ Lissajous case in presence of measurement noise. Figure 4.14 and Figure 4.15 depict the control inputs and 3D position tracking respectively. We observe the same satisfactory results as in the B-spline trajectory case.

4.8.2 Scenario 2: Unknown time-varying wind disturbance

In this scenario, we investigate the quadrotor tracking in presence of wind disturbance that is not constant and is not assumed to be known. The high varying wind disturbances used in the simulations (displayed in Figure 4.16) are represented by a sum of sinusoidal waves:

$$w(t) = 1.5\mu_{31} + \mu_7 + 0.5\mu_2 + 0.015\mu_{11} + 0.15b(t) \tag{4.48}$$

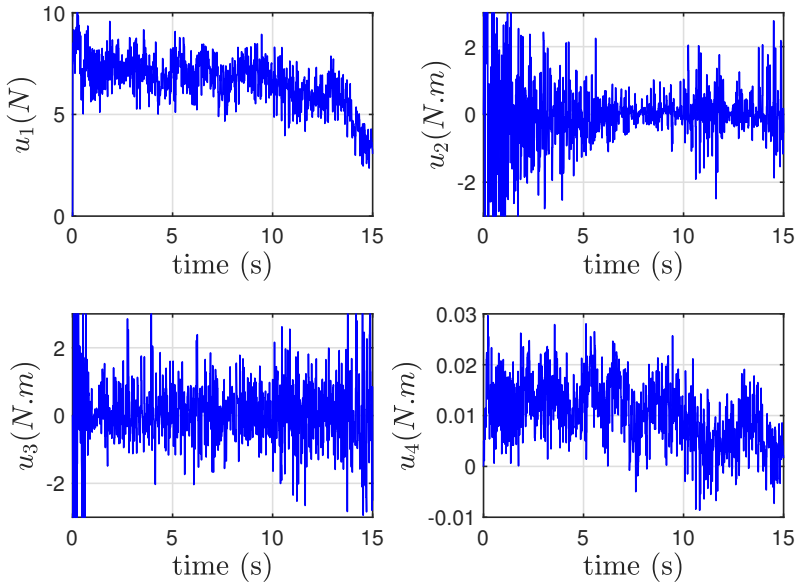


Figure 4.12: B-Spline trajectory case: Control inputs in presence of noise.

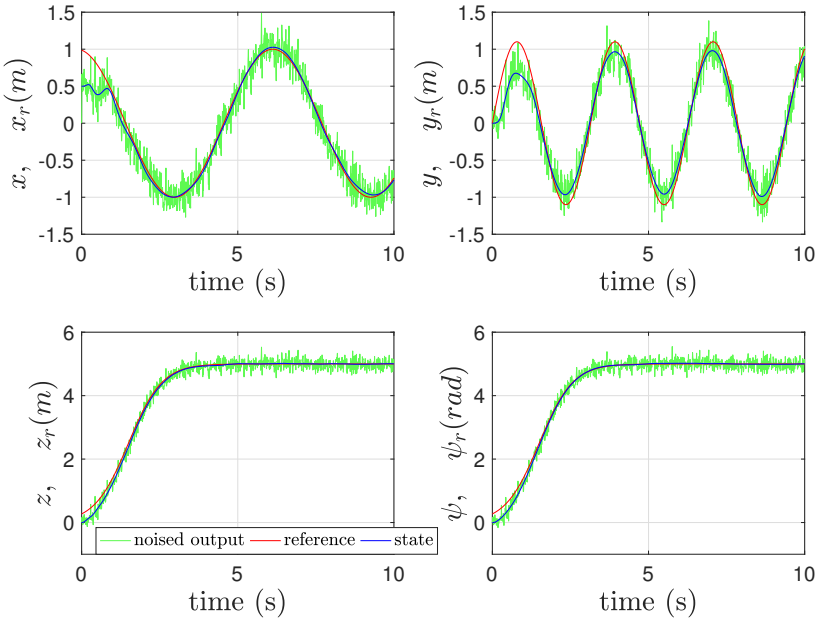


Figure 4.13: Lissajous trajectory case: Tracking trajectories in presence of noise.

where $\mu_p(t) = \sin\left(\frac{t\pi}{p}\right)$ and $b(t)$ is the random Gaussian noise. We add this disturbance in the simulation model as an additive force disturbance such that

$$m\ddot{x} = (\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi)u_1 - A_x\dot{x} + w(t), \tag{4.49a}$$

$$m\ddot{y} = (-\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi)u_1 - A_y\dot{y} + w(t), \tag{4.49b}$$

$$m\ddot{z} = -mg + (\cos\theta\cos\phi)u_1 - A_z\dot{z} + w(t). \tag{4.49c}$$

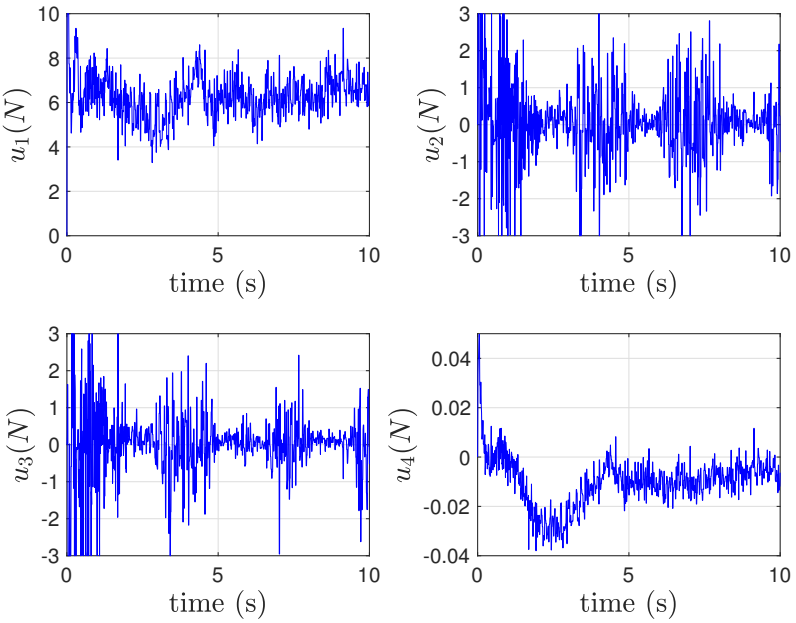


Figure 4.14: Lissajous trajectory case: Control inputs in presence of noise.

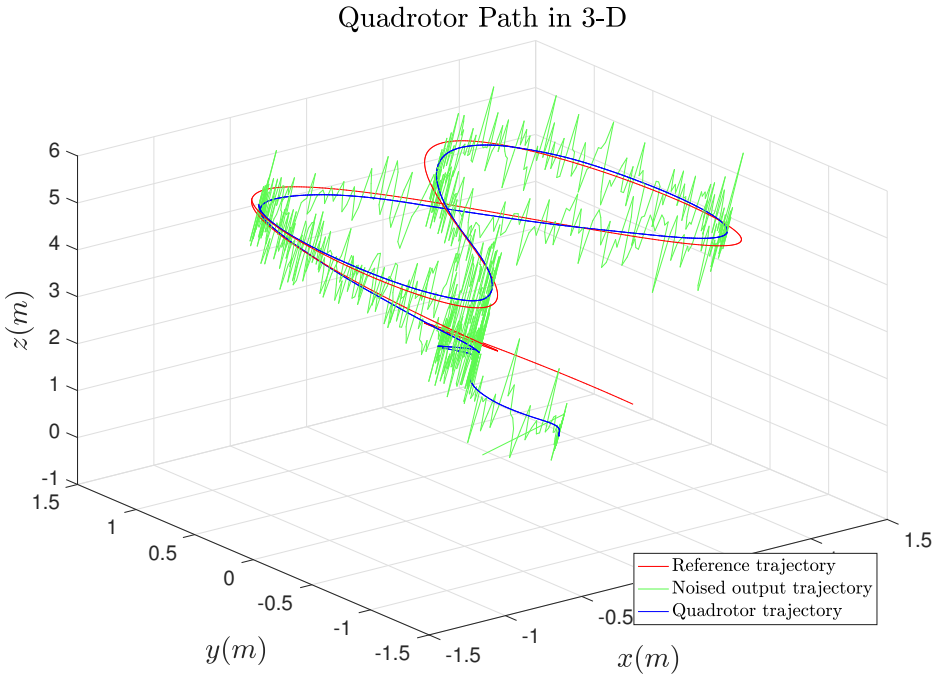


Figure 4.15: 3-D Position tracking of Lissajous path in presence of noise.

B-spline trajectory case

We observe the unknowns F_i for $i = x, y, z, \psi$ in the nominal case (see Figure 4.9) and in the presence of wind disturbance (see Figure 4.18). We show that the

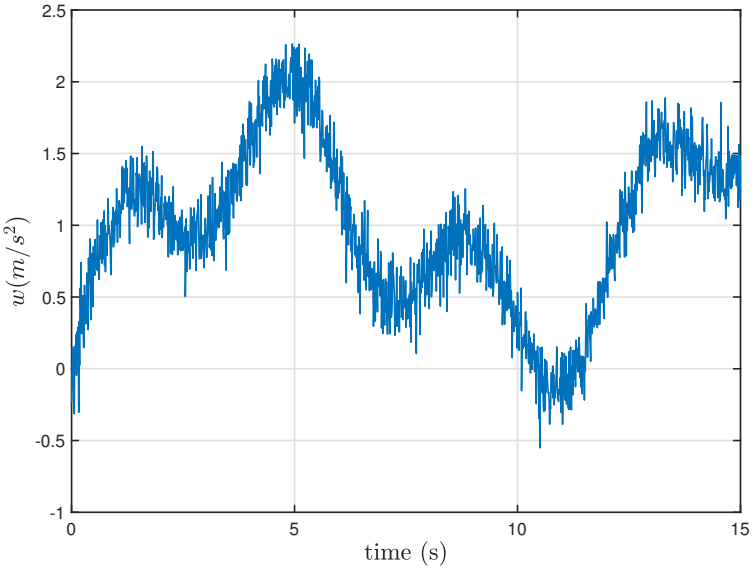


Figure 4.16: Wind disturbance.

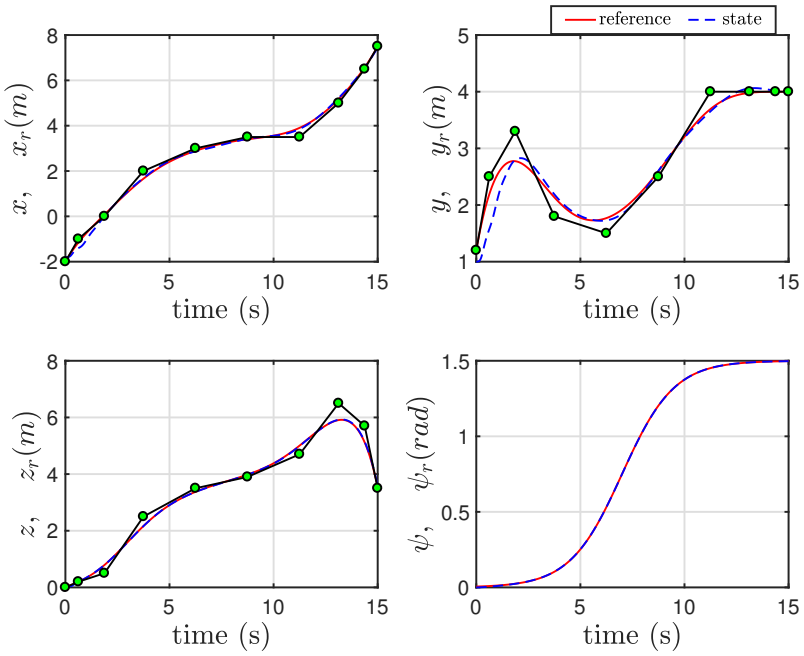


Figure 4.17: B-Spline trajectory case: Tracking outputs in presence of wind disturbance.

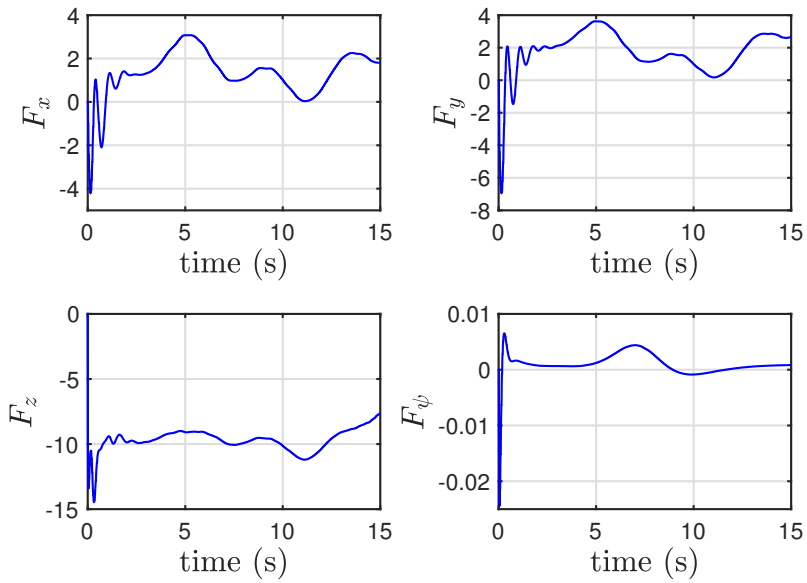


Figure 4.18: B-Spline trajectory case: Estimation of the unknowns F_i for $i = x, y, z, \psi$ in presence of wind disturbance.

precise estimation of the unknown F_i for $i = x, y, z, \psi$ leads us to a robust control performance. The trajectory tracked by the quadrotor in presence of time-varying wind disturbance depicted in Figure 4.17 is almost identical to the nominal case.

Lissajous trajectory case

Figure 4.19 shows the good performance when the unknown wind disturbance impacts the position tracking.

4.8.3 Scenario 3: Mass parameter variation

One of the useful features of the quadrotor is transporting a payload. Here we examine the situation when the quadrotor is carrying a payload by changing the mass value during the flight.

Lissajous trajectory case

The simulations show that the quadrotor successfully picks up the payload at $t = 3s$, transports (hovering) and at the end drops off the payload at $t = 8s$ while tracking an aggressive position trajectory and yaw angle (see Figure 4.22). We observe F_z in the nominal case (Figure 4.3) when the mass value doesn't change over time in Figure 4.5 and F_z when the mass varies in Figure 4.20. We notice that

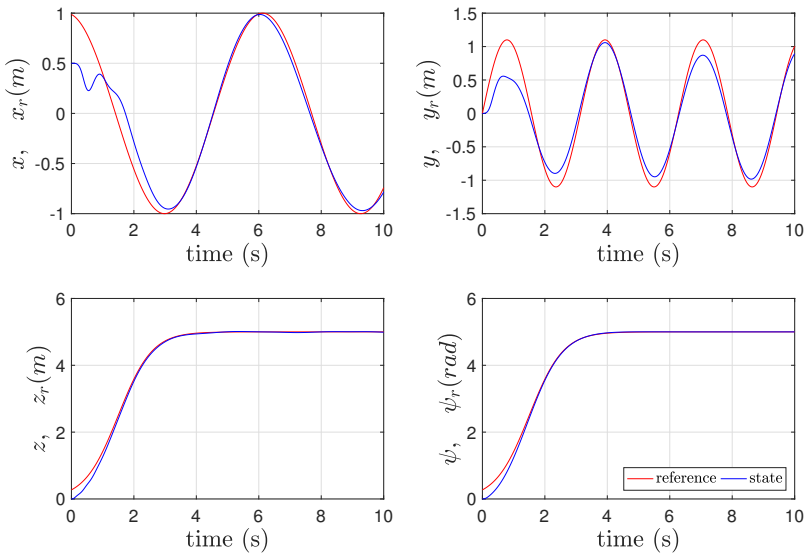


Figure 4.19: Lissajous trajectory case: Tracking outputs in presence of wind disturbance.

the mass variation is motion dependent and that the good estimate of F_z results in changes in the thrust control u_1 (shown in Figure 4.21). So, the quadrotor's tracking performance of an aggressive trajectory will be limited by its maximal thrust. By the latter, we show that the precise estimation of the unknown F_z leads us to a robust control performance.

In this particular example, we take the variation of the mass parameter but we would have the same outcome if we had changed several parameters such as the inertia parameters at the same time.

4.9 Closing remarks

This methodology offers a cascaded model-free control design of a quadrotor. It stays robust, despite unknown disturbances and parameter variation. We have shown that a complex trajectory tracking on realistic scenarios is feasible. We have analyzed the practical stability of the proposed closed-loop system (quadrotor and cascaded model-free controller). Considering the low complexity and the good performances of the proposed control, it sheds new lights on a possible commercial off-the-shelf solution.

The final step of any control theory lies in experiment. In terms of future works, the author hopes to apply the proposed control law in real experiments with quadrotors.

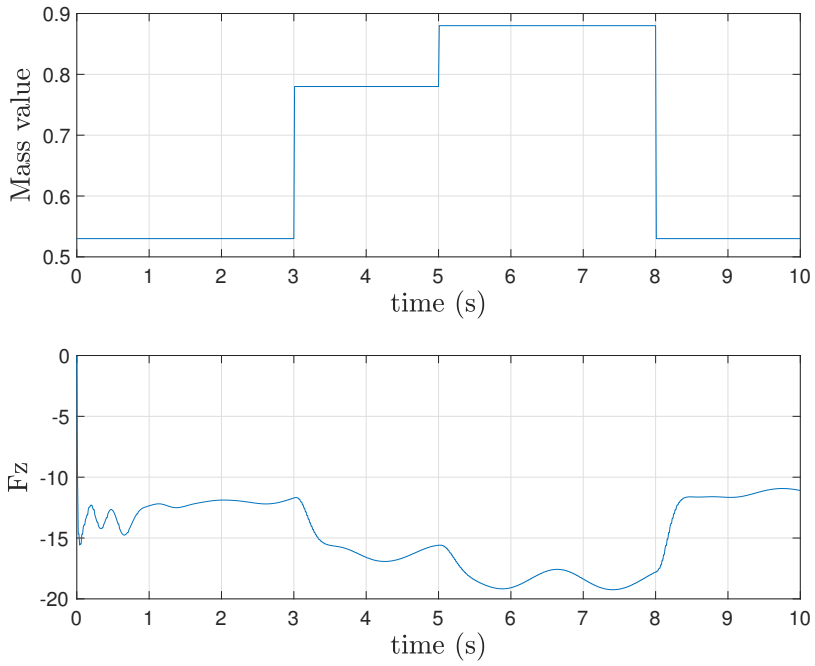


Figure 4.20: Lissajous trajectory case: Mass value variation over time and estimation of the unknown F_z.

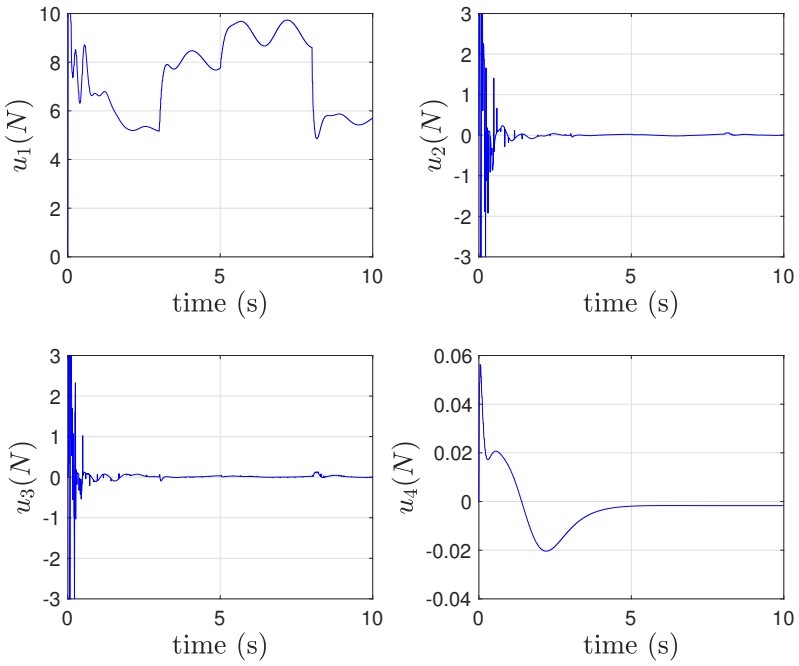


Figure 4.21: Lissajous trajectory case: The control inputs when the mass varies over time.

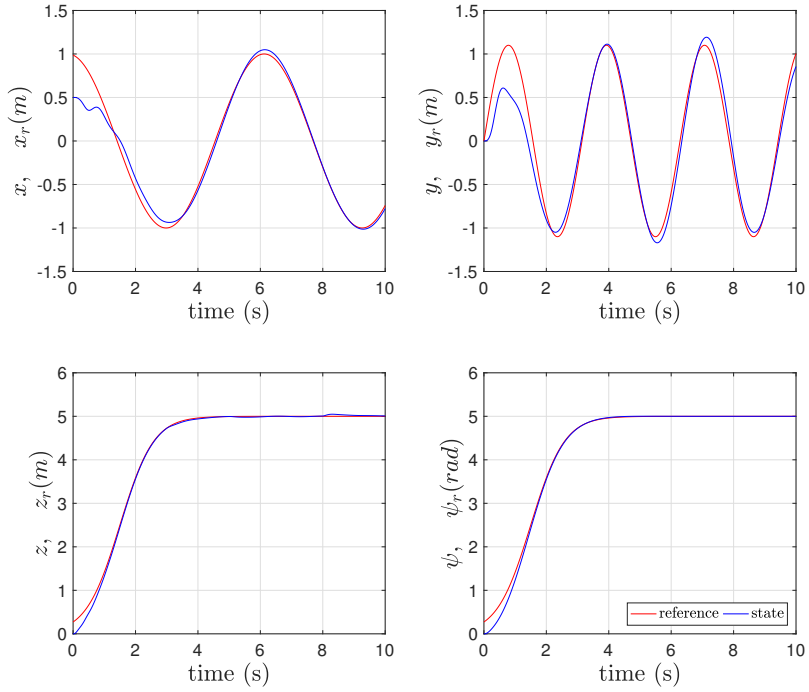


Figure 4.22: Lissajous trajectory case: The tracking outputs when the mass varies over time.

Appendix

4.A Boundedness of the interconnection term

For sake of completeness, we include the bound of the interconnection term defined as

$$\|R_\psi u_1 \Delta(e_\phi, e_\theta, r_d)\| \leqslant \|u_1\| \, \|\Delta(e_\phi, e_\theta, r_d)\| \, . \tag{4.50}$$

- For $\|u_1\|$, we have:

$$\begin{aligned} \|u_1\| &= \frac{1}{\alpha_z} \left\| (-\hat{F}_z + \ddot{z}_d - K_{pz} e_z - K_{dz} \dot{e}_z) \right\| \\ &\leqslant \frac{1}{\alpha_z} (\left\| \hat{F}_z \right\| + \|\ddot{z}_d\| + K_{pz} \|e_z\| + K_{dz} \|\dot{e}_z)\|) \\ &\leqslant \frac{1}{\alpha_z} (B_{F_z} + L_z) + \frac{1}{\alpha_z} (K_{pz} \|e_z\| + K_{dz} \|\dot{e}_z)\|)) \\ &\leqslant M + N \|e_z\| \end{aligned} \tag{4.51}$$

  where $M = \dfrac{1}{\alpha_z}(B_{F_z} + L_z)$ and $N = \dfrac{1}{\alpha_z} \max(\lambda_{pz}, \lambda_{dz})\sqrt{2}$ are constants.

- For $\|\Delta(e_\phi, e_\theta, r_d)\|$, we have:

$$\|\Delta(e_\phi, e_\theta, r_d)\| = \sqrt{h_x^2 + h_y^2}. \tag{4.52}$$

  Having

$$\begin{aligned} |h_x| &\leqslant |\sin(e_\phi/2)|, \\ |h_y| &\leqslant |\sin(e_\phi/2)\sin(e_\theta/2)| + |\sin(e_\theta/2)| + |\sin(e_\phi/2)|, \end{aligned} \tag{4.53}$$

  and then, by using following inequalities

$$\begin{aligned} &|\sin a| \leqslant |a|, \\ &|a||b| \leqslant \frac{1}{2}(|a| + |b|), \text{ for } |a| \leqslant 1 \text{ and } |b| \leqslant 1, \end{aligned} \tag{4.54}$$

  we obtain

$$\begin{aligned} |h_x| &\leqslant \frac{1}{2}|e_\phi|, \\ |h_y| &\leqslant \frac{3}{2}(|\sin(e_\theta/2)| + |\sin(e_\phi/2)|) \leqslant \frac{3}{4}(|e_\theta| + |e_\phi|). \end{aligned} \tag{4.55}$$

Then, we compute the squared expressions of (4.55) such that

$$\begin{aligned} h_x^2 &\leqslant \frac{1}{4} e_\phi^2 \\ h_y^2 &\leqslant \frac{9}{16}(e_\theta^2 + e_\phi^2 + 2|e_\theta||e_\phi|) \end{aligned} \tag{4.56}$$

and knowing that $2|e_\theta||e_\phi| \leqslant e_\theta^2 + e_\phi^2$, we obtain

$$\begin{aligned} h_x^2 &\leqslant \frac{1}{4} e_\phi^2, \\ h_y^2 &\leqslant \frac{9}{8}(e_\theta^2 + e_\phi^2). \end{aligned} \tag{4.57}$$

Finally, we get that

$$\|\Delta(e_\phi, e_\theta, r_d)\| = \sqrt{h_x^2 + h_y^2} \leqslant \sqrt{\frac{11}{8} e_\phi^2 + \frac{9}{8} e_\theta^2} \leqslant \sqrt{\frac{11}{8}(e_\phi^2 + e_\theta^2)} \leqslant K \|e_r\| \tag{4.58}$$

with $K = \sqrt{\frac{11}{8}}$.

The bound of the interconnection term is

$$\|R_\psi u_1 \Delta(e_\phi, e_\theta, r_d)\| \leqslant (M + N\|e_z\|) K \|e_r\|. \tag{4.59}$$

4.B Bound on the estimation error e_F

Here we follow the same reasoning as in [139].
The estimation error $e_F = F - \hat{F} \in \mathbb{R}$ is time-varying, and can be expressed as

$$e_F = y^{(\nu)} - \widehat{y^{(\nu)}} - \alpha(u - \hat{u}) \tag{4.60}$$

where $\hat{u}$ is the control input u in the previous time interval.
Let us suppose, without loss of generality, that $\widehat{y^{(\nu)}}$ is the result of a filtering process of $y^{(\nu)}$. For example, in the Laplace domain:

$$\mathcal{L}(\widehat{y^{(\nu)}}) = \left(\frac{1}{1+Ts}\right)^\nu \mathcal{L}(y^{(\nu)})$$

where $\mathcal{L}$ designates the Laplace transform. Since this is a filtering process, T is assumed to be small, i.e. $T \ll 1$. In Fourier transform terms, this yields, following the same steps as in [36]:

$$\begin{aligned} \mathcal{F}(y^{(\nu)}) - \mathcal{F}(\widehat{y^{(\nu)}}) &= \left(1 - \frac{1}{(1+2i\pi T\omega)^\nu}\right) \mathcal{F}(y^{(\nu)}) = \left(1 - \frac{1}{(1+2i\pi T\omega)^\nu}\right) \frac{1}{2i\pi\omega} \mathcal{F}(y^{(\nu+1)}) \\ &= \frac{(1+2i\pi T\omega)^\nu - 1}{(1+2i\pi T\omega)^\nu} \frac{1}{2i\pi\omega} \mathcal{F}(y^{(\nu+1)}) = \frac{\sum_{k=0}^{\nu} \binom{\nu}{k} (2i\pi T\omega)^k - 1}{2i\pi\omega(1+2i\pi T\omega)^\nu} \mathcal{F}(y^{(\nu+1)}) \\ &= \frac{\sum_{k=1}^{\nu} \binom{\nu}{k} (2i\pi T\omega)^k}{2i\pi\omega(1+2i\pi T\omega)^\nu} \mathcal{F}(y^{(\nu+1)}) = T \frac{\sum_{k=0}^{\nu} \binom{\nu}{k} (2i\pi T\omega)^{k-1}}{(1+2i\pi T\omega)^\nu} \mathcal{F}(y^{(\nu+1)}) \end{aligned}$$

Thus, the difference is, by Plancherel's theorem:

$$\left\|y^{(\nu)} - \widehat{y^{(\nu)}}\right\|_\infty = \left\|\mathcal{F}(y^{(\nu)}) - \mathcal{F}(\widehat{y^{(\nu)}})\right\|_\infty \leqslant T \left\|\frac{\sum_{k=0}^{\nu}\binom{\nu}{k}(2i\pi T\omega)^{k-1}}{(1+2i\pi T\omega)^\nu}\right\|_\infty \left\|\mathcal{F}(y^{(\nu+1)})\right\|_\infty$$

And the function

$$\varphi(\omega) = \left|\frac{\sum_{k=0}^{\nu}\binom{\nu}{k}(2i\pi T\omega)^{k-1}}{(1+2i\pi T\omega)^\nu}\right| = \left|\frac{(1+2i\pi T\omega)^\nu - 1)}{2i\pi\nu(1+2i\pi T\omega)^\nu}\right|$$

has a unique maximum in $\omega = 0$.

Hence, we have

$$\left\|\frac{\sum_{k=0}^{\nu}\binom{\nu}{k}(2i\pi T\omega)^{k-1}}{(1+2i\pi T\omega)^\nu}\right\|_\infty = \varphi(0) = \binom{\nu}{1} = \nu$$

And then we get

$$\left\|y^{(\nu)} - \widehat{y^{(\nu)}}\right\|_\infty \leqslant T\nu \left\|\mathcal{F}(y^{(\nu+1)})\right\|_\infty = T\nu \left\|y^{(\nu+1)}\right\|_\infty$$

Suppose for instance that y is of some Gevrey class of order β on a compact subset $K \subset \mathbb{R}$, i.e.

$$\exists m_K, \gamma_K \in \mathbb{R}^+, \forall k \in \mathbf{N}, \quad \sup_{t\in K}|y^{(k)}(t)| \leqslant \frac{m_K}{\gamma_K^k}(k!)^\beta.$$

Recall that functions of Gevrey order $\beta < 1$ are entire, while analytic for $\beta = 1$ and non-analytic if $\beta > 1$.

Then, we obtain the following estimate

$$\left\|y^{(\nu)} - \widehat{y^{(\nu)}}\right\|_\infty \leqslant \frac{T\nu m_K}{\gamma_K^\nu}\Big((\nu+1)!\Big)^\beta$$

Considering $u - \hat{u}$, and taking for instance the approximation

$$\hat{u}(t) = u(t-h)$$

where h is the variable sampling period, and supposing u to be Lipschtiz with constant L, one has

$$|u(t) - \hat{u}(t)| \leqslant Lh$$

Then, the error e_F admits the following estimate

$$e_F \leqslant \left\|y^{(\nu)}(t) - \widehat{y^{(\nu)}}(t)\right\|_\infty + \alpha\left\|\hat{u}(t) - \hat{u}(t)\right\|_\infty \leqslant \frac{T\nu m_K}{\gamma_K^\nu}\Big((\nu+1)!\Big)^\beta + \alpha Lh \quad (4.61)$$

If a small time interval h with respect to system dynamics is selected, as well as a small filtering time constant T, the bound on e_F will small. Then, according to equation (4.11), the tracking error e will remain in a bounded ball and the system is practical stable. Notice that in our case $\nu = 2$.

5

Model-Free Control Framework for Cloud Resource Elasticity

Contents

Abstract: In cloud computing management, the dynamic adaptation of computing resource allocations under time varying workload is an active domain of investigation. Several control strategies were already proposed. Here the Model-Free Control setting and the corresponding "intelligent" controllers, which are most successful in many concrete engineering situations, are employed for the "horizontal elasticity". When compared to the commercial "Auto-Scaling" algorithms, our easily implementable approach behaves better, even with sharp workload fluctuations. This is confirmed by experiments on Amazon Web Services (AWS).

5.1 Introduction

5.1.1 Data is driving the revolution

From the industry to healthcare, from connected objects to social networks, the areas may be diverse but together they tell a similar story: the amount of data and processing work in the world is growing fast. As a result, the traditional IT infrastructures are not flexible enough to handle this growth. An attractive solution to this problem is the "limitless" power of cloud computing.

Let us begin with an example: When a website or mobile application is accessed, an internet connection is created with a distant computer. In this setting, the distant computer provides a service (here a software) over the internet to its final user. This service provisioning model is commonly regarded as Software as a Service or SaaS. A SaaS provider relies on an existing IT infrastructure generally composed of the following resources: computing, storage, networking, power supply, cooling systems.

There exist multiple configurations of how, where and by whom the IT infrastructure is managed. Traditionally the necessary resources were installed and managed inside the SaaS providers company premises. Due to the complexity in maintaining a fully functional installation, companies rapidly externalized the hosting of their IT infrastructure to highly monitored and maintained Data Centers.

With the rapid expansion of internet services, more companies were creating IT infrastructures to serve their clients worldwide. However, for a software company at early stages there remains four difficulties in a conventional IT Infrastructure:

- The necessity to invest in costly IT hardware and software upfront.

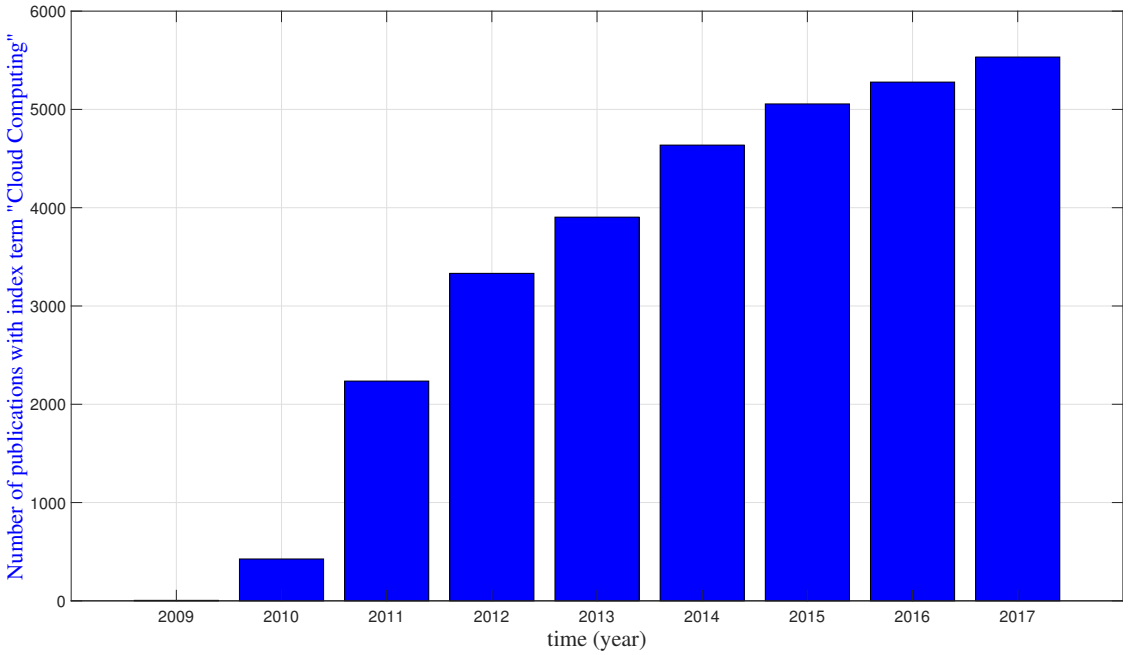


Figure 5.1: Number evolution of publications having "Cloud Computing" as index term in the IEEE Database (`http://ieeexplore.ieee.org`)

- To predict and pre-provision an IT installation that can sustain estimated peak loads of customer requests.
- Any modification or evolution to an infrastructure would take a considerable amount of time to physically provision the necessary hardware and install/configure the required software.
- Recovery from infrastructure failures would be costly and time consuming and would impact the business significantly.

5.1.2 Utility Computing and Cloud Computing

With the adoption of hardware virtualization in early 2000s and the popularization of internet usage, a long thought idea from 1960s around a service provisioning model called Utility Computing became a reality. The service provisioning was based on a pay-as-you-go model for the use of computing resources. Adding the capability to provision and use computing resources on-demand through internet gave birth to Cloud Computing. As shown in Figure 5.1, the scientific contributions to cloud computing research grew rapidly over the past 10 years.

Among early providers of Cloud Computing through a public offering, we can mention Amazon with the commercialization of Elastic Compute Cloud (EC2) in 2006, Google AppEngine in 2008 and Microsoft Azure in 2010.

In recent years, the cloud computing adoption among internet companies grew exponentially. We mention here some major use cases:

- *File Storage:* Some Cloud Storage providers are using public cloud infrastructure to host their services. For example, Apple iCloud relies on Google Cloud Platform and Amazon Web Services to host and manage its Storage service.
- *Video Streaming:* As an example, Netflix is extensively using Amazon Web Services Public Cloud for its video on demand platform.
- *Artificial Intelligence:* Training machine learning models is mostly done on high performance GPU instances in the cloud.
- *Connected Vehicles:* Car manufacturers are using cloud services to provide connected vehicle services to their customers. For example, Renault-Nissan is using Microsoft Azure and Amazon Web Services.
- *Mobile Applications:* A great majority of mobile applications are using cloud services to provide content (multimedia, user data, etc.) and dynamic functionalities.
- *Banking:* Many financial institutions are migrating their traditional infrastructures to the cloud. For example, Société Générale is extensively using Amazon Web Services and Microsoft Azure.

Before going further, let us introduce some commonly used terms in Table 5.1.

5.1.3 Problem statement

Cloud services are usually expected to deliver results in a limited (constrained) amount of time. Final users expect reactivity in their interaction with the service. Being able to satisfy computing requests in a given maximum amount of time requires knowledge of the task complexity and an estimation of a required computing power. To satisfy the time constrains, the Cloud computing power should adapt itself. Cloud Users need to optimize and adapt the allocated resources to the performance needs, in order to maintain Business performance requirements while reducing Cloud Platform costs. On the other hand, popular Cloud Providers like Amazon Web Services (AWS), Google Cloud Platforms or Microsoft Azure are all looking

Term	Definition
Cloud User or SaaS Provider	An entity who uses cloud infrastructure to create and offer internet services.
Final Users or SaaS User	An entity who uses the services offered by the Cloud User or SaaS Provider.
Cloud Platform	A combination of computer hardware and software that can provide computing and storage resources over the internet.
Cloud Provider	An entity which create, provide and maintain the Cloud Platform to the Cloud User.
Virtual Machine	is an emulation of a computer system which provides the necessary functionality to execute an operating system and its applications.
Computing power	Usually measured in number of vCPU (virtual Central Processing Unit) that is a share of a physical CPU attributed to a Virtual Machine.
Scalability	The ability for a cloud platform to adjust the computing power to a Cloud User's need by provisioning and deprovisioning resources. (Virtual Machines).
Auto-scaling	The automatic adjustments of the computing power based on measurable usage indicators like CPU usage.
User Traffic/Workload	The amount of Final User requests arriving on a SaaS provider internet service during a period of time.

Table 5.1: Commonly used terms in Cloud Computing.

for ways to provide the most tailored services by optimizing the resource allocation across clients. Having clear physical limits tied to the number of physical machines running in their datacenters, the Cloud Provider faces clear resource management challenges to optimize operating cost. Following this train of thought, the economic gains coming from an intelligent computing resource management system is totally obvious.

The survey papers [5, 6] give a clear introduction of the cloud computing services and resume the benefits and the obstacles that may be encountered by using them. One of the main features of cloud computing is *resource elasticity* (the obstacle 8 from [5, 6]). With this feature, the traditional capacity planning and resource anticipation procedures disappear. Cloud Users won't have to guess the maximum capacity needs and thus under-actuate their resources. Computing power elasticity/scaling happens in two ways: horizontally and vertically. *Vertical up or down scaling* is achieved by adding more computing capacity to an existing machine

(more CPU and RAM), while the horizontal scaling adds more machines (VM instances) to increase computing capacity as a whole. Vertical scaling takes more time to apply and usually requires system shutdown. On the contrary, *horizontal out or in scaling* can be done on a running cloud infrastructure without much technical side-effects. This makes horizontal scaling more interesting economically and technically.

Horizontal scaling can either be done manually or automatically. Current computing resource management systems use auto-scaling techniques in order to meet compute capacity demands. The management software takes as input standard system metrics like average RAM usage, average CPU usage, average Network throughput. The Cloud User can either define thresholds on each of these metrics in order for the management software to trigger the out or in scaling process (adding or removing compute nodes) or define rules such as peak traffic scheduling to scale based on previous recorded compute needs. For example, Amazon algorithms can trigger scaling actions based on common metrics like average CPU usage. For example, when scaling out is triggered, the amount of extra resources necessary is either predefined (step-scaling) or automatically computed (target tracking). Amazon tends to have a conservative approach and keeps the scaled-out cluster for an extended amount of time before scaling in. We believe there can be a clear improvement of this algorithm by more accurately tracking the performance objectives and adjusting the cluster size with a more agile dynamic (see Figure 5.2). Better performance tracking and clear financial gains are promised. Control Theory has the necessary tools to solve problems such as auto-scaling in a more efficient way than the threshold or rule-based auto-scaling (see [9, 77]).

In this chapter, we employ the Model-Free Control to tackle the "horizontal elasticity" of a Cloud Services system. For our application to control a running cloud service, we will use a production-ready 3-tier web service architecture under realistic workload (English Wikipedia web traffic) on Amazon Web Services infrastructure.

Outline

This chapter has the following outline:

- In Section 5.2, we briefly describe the related existing approaches.
- In Section 5.3, we review the MFC for systems of first order dynamics.
- In Section 5.4, we present our model-free framework for the "horizontal elasticity" of a Cloud services system.

- In Section 5.5, we describe the experimental setup and, then, we report the successful results from our real experiments.

The results from this chapter have been published in [14].

5.2 Existing approaches

Let us briefly summarize some of the publications (see *e.g.* [134]) that have already used control theory to improve *horizontal elasticity*, with results on public clouds.

In [64], a feedback-based control strategy using a Gain Scheduling policy to dynamically allocate computing resources to guarantee a pre-defined Service Level Objective despite the presence of large fluctuating loads and VM failures is introduced. In [65], the controller gain is programmed to best assign virtual machines by employing a fuzzy approach without relying on a model. The 30% benchmark is a sign of underutilization of resources and cannot be considered as a realistic goal. The behaviour remains relatively similar under different query loads: Wikipedia, FIFA98 and constant load. For example, under a constant load, their algorithm adds a considerable number of VMs without having a direct impact on the average CPU load. Since the workload trajectory is not presented, we cannot judge the performance of the results. The paper [9] explores the differences between an industrial public cloud, Amazon EC2, and a community-driven research cloud environment, SAVI, which is a custom OpenStack implementation. They deployed PID controllers in both type of clouds and made a comparison between their behaviors. The PID controllers are used under the following conditions: 1) they can auto-scale only one tier of the application. 2) The goal of the controller is to maintain utilization of the tier, which can be monitored, to certain levels. By extension (Equation 2 in the paper [9]), they also control response times of this tier. 3) The input of the system is the number of servers in the controlled cluster. They assume that the workloads are limited in intensity and that they do not cause the saturation of the uncontrolled tiers. The PID correctors deployed by [9] supervise a single floor. The performance for several cloud architectures is compared. Despite the fine VM granularity in the results, the deviation of the average CPU load from the 55% benchmark is very high for a production environment with a real traffic load. These CPU spikes can make the web application unstable and have a direct effect on the response time. In [109], the experiments are done on the Amazon EC2 cloud with Docker containers. They compare three different methods: Thresholds, PID and Model-based (a simple linear model). They do not assume a model of the application or a model of the cloud as it happens with complex

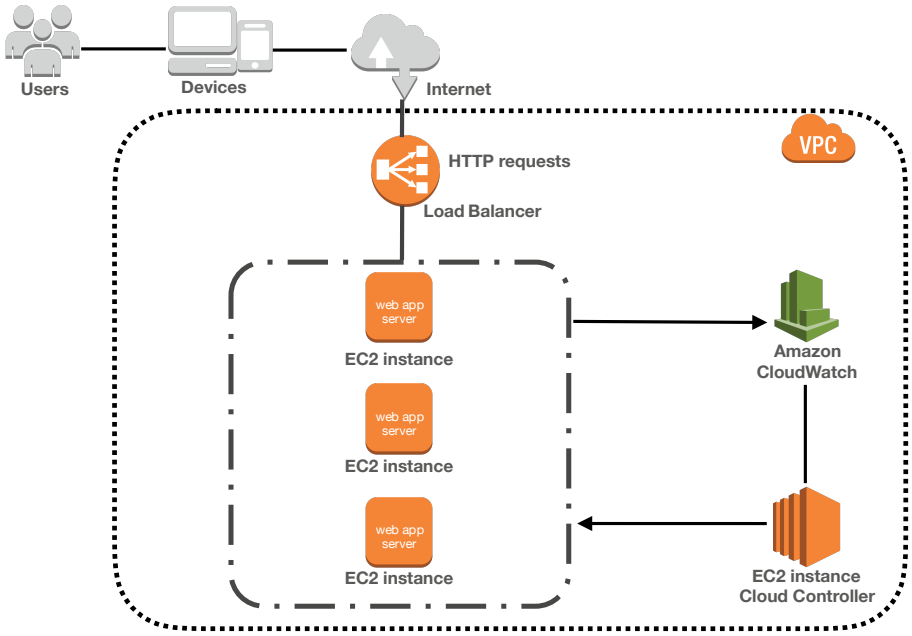


Figure 5.2: High-level architecture of the system

optimization algorithms. In [86], it is considered having different types of resources. Their autoscaling approach consists not only in changing the number of resources to be allocated, but also in considering the type of resource that need to be added or removed while respecting fixed budget constraints. In many company use-cases, good prediction of the workloads is of great importance to manage the resource provisioning. Recent contribution in this direction is given in [53] where the workload predictions are determined through efficient time-series forecasting techniques.

5.3 Model-Free Control

In the previous Chapter, in Section 4.3.1, we briefly reviewed the *Model-Free Control* (MFC) and its "inteligent" controller when the system is assumed to be of second order dynamics. The aim of this section is to review the *Model-Free Control* (MFC) for first order dynamics that is used in this chapter.

5.3.1 The ultra-local model

For a system dynamics of first order *i.e.* $\nu = 1$, we replace the unknown global description by the *ultra-local model*:

$$\dot{y} = F + \alpha u \tag{5.1}$$

where

- the control and output variables are u and y,
- the derivation order of y is 1 like in most concrete situations,
- $\alpha \in \mathbb{R}$ is chosen by the practitioner such that αu and $\dot{y}$ are of the same magnitude.

The following explanations on F may be useful:

- F is estimated via the measure of u and y,
- F subsumes not only the unknown system structure but also any perturbation.

5.3.2 Intelligent controllers

The loop is closed by an *intelligent proportional controller*, or *iP*,

$$u = -\frac{F - \dot{y}_d - K_P e}{\alpha} \tag{5.2}$$

where

- y_d is the reference trajectory,
- $e = y - y_d$ is the tracking error,
- K_P is the usual tuning gain.

Combining Equations (5.1) and (5.2) yields:

$$\dot{e} = K_P e$$

where F does not appear anymore. The tuning of K_P, in order to insure local stability, becomes therefore quite straightforward. This is a major benefit when compared to the tuning of classic PIDs.

Remark 5.1 *An appropriate choice of y_d is of utmost importance. Let us emphasize that this step is almost always knowledge-based.*

5.3.3 Estimation of F

The computations below stem from the estimation techniques (see [52]).

First approach

The term F in Equation (5.1) may be assumed to be "well" approximated by a piecewise constant function $\hat{F}$. Rewrite then Equation (5.1) in the operational domain (see, *e.g.*, [141]):

$$sY = \frac{\Phi}{s} + \alpha U + y(0)$$

where Φ is a constant. We get rid of the initial condition $y(0)$ by acting on both sides, through $\frac{d}{ds}$:

$$Y + s\frac{dY}{ds} = -\frac{\Phi}{s^2} + \alpha\frac{dU}{ds}$$

Noise attenuation is achieved by multiplying both sides on the left by s^{-2}. It yields in the time domain the real-time estimate, thanks to the equivalence between $\frac{d}{ds}$ and the multiplication by $-t$,

$$\hat{F}(t) = -\frac{6}{\tau^3}\int_{t-\tau}^{t} \left[(\tau - 2\sigma)y(\sigma) + \alpha\sigma(\tau - \sigma)u(\sigma)\right] d\sigma \tag{5.3}$$

Second approach

Close the loop with the iP (5.2):

$$\hat{F}(t) = \frac{1}{\tau}\left[\int_{t-\tau}^{t} (\dot{y}_d - \alpha u - K_P e)\, d\sigma\right] \tag{5.4}$$

Remark 5.2 *Note the following facts:*

- *integrals* (5.3) *and* (5.4) *are low pass filters,*
- $\tau > 0$ *might be quite small,*
- *the integrals may of course be replaced in practice by classic digital filters.*

Remark 5.3 *A hardware implementation of the above computations is easy [73].*

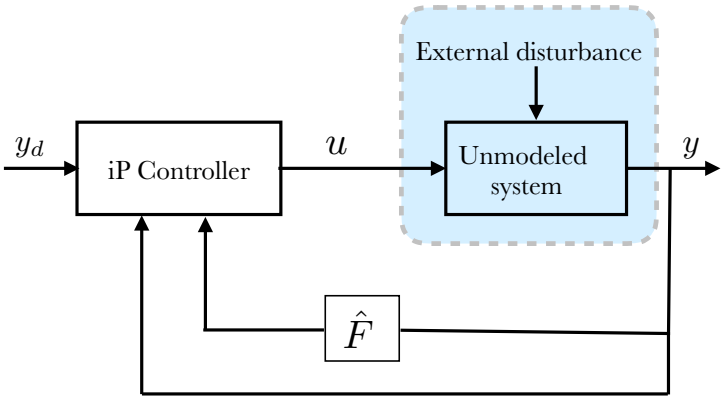


Figure 5.3: Model-Free Control scheme

5.4 Model-Free setting in the Cloud framework

In the case of a Cloud services system, we employ the ultra local model of first order (5.1) and the iP controller (5.2).

- The control output y represents the *central processing units* (CPU) usage, during the sampling interval $[t-h, t)$:
  - Let $\mathrm{CPU}_i^h(t)$ be the average load, over the time interval $[t-h, t)$, of the ith virtual machine processor, depending on the received requests.
  - Let $M_{\mathrm{act}}^h(t)$, $M_{\min} \leqslant M_{\mathrm{act}}^h(t) \leqslant M_{\max}$, be the number of actives virtual machines.

  Therefore
  $$y(t) = \sum_{i=1}^{M_{\mathrm{act}}^h(t)} \mathrm{CPU}_i^h(t) \tag{5.5}$$
- For the desired trajectory or set-point, y_d, we choose
  $$y_d(t) = \frac{M_{\mathrm{act}}^h(t)}{2} \tag{5.6}$$

  This choice is a trade-off. Thus, a set-point equal to $0.3 \times M_{\mathrm{act}}^h(t)$ (respectively $0.8 \times M_{\mathrm{act}}^h(t)$) would imply an under-exploitation (respectively over-exploitation). Note that any significant overexploitation leads to a significant delay in the execution of requests. Hence, a deterioration in the quality of service.
- The control input u represents the cluster cardinality, *i.e.* the number of active virtual machines.

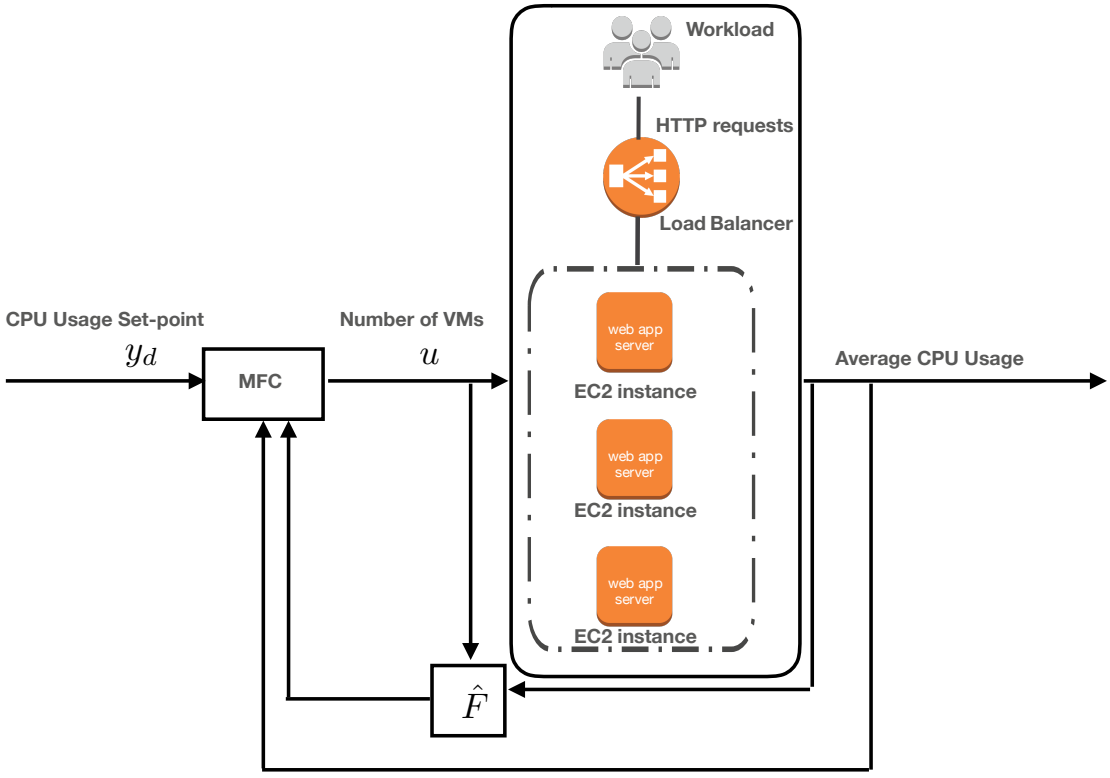


Figure 5.4: Control Design

Remark 5.4 *In the experiments, in section 5.5, the CPU values used in the computation of the setpoint and the output measurements are given in percentages. That's why, for example,* 1/2 *in the equation* (5.6) *corresponds to* 50%*.*

The control design is shown in Figure 5.4.

5.5 Experiments

For our experiments, we implemented the architecture in Figure 5.5.

The architecture components are the following:

1. *Workload Generator instances:* We create EC2 instances evenly distributed in 5 different geographical regions (North California, Mumbai, Sao Paolo, Paris, Sydney). For HTTP request generation, Apache JMeter (an advanced API testing/benchmarking tool) is used with the following plugins:

    - Throughput Shaping Timer (to define a time-varying RPS (request per second) trajectory with an open workload approach)

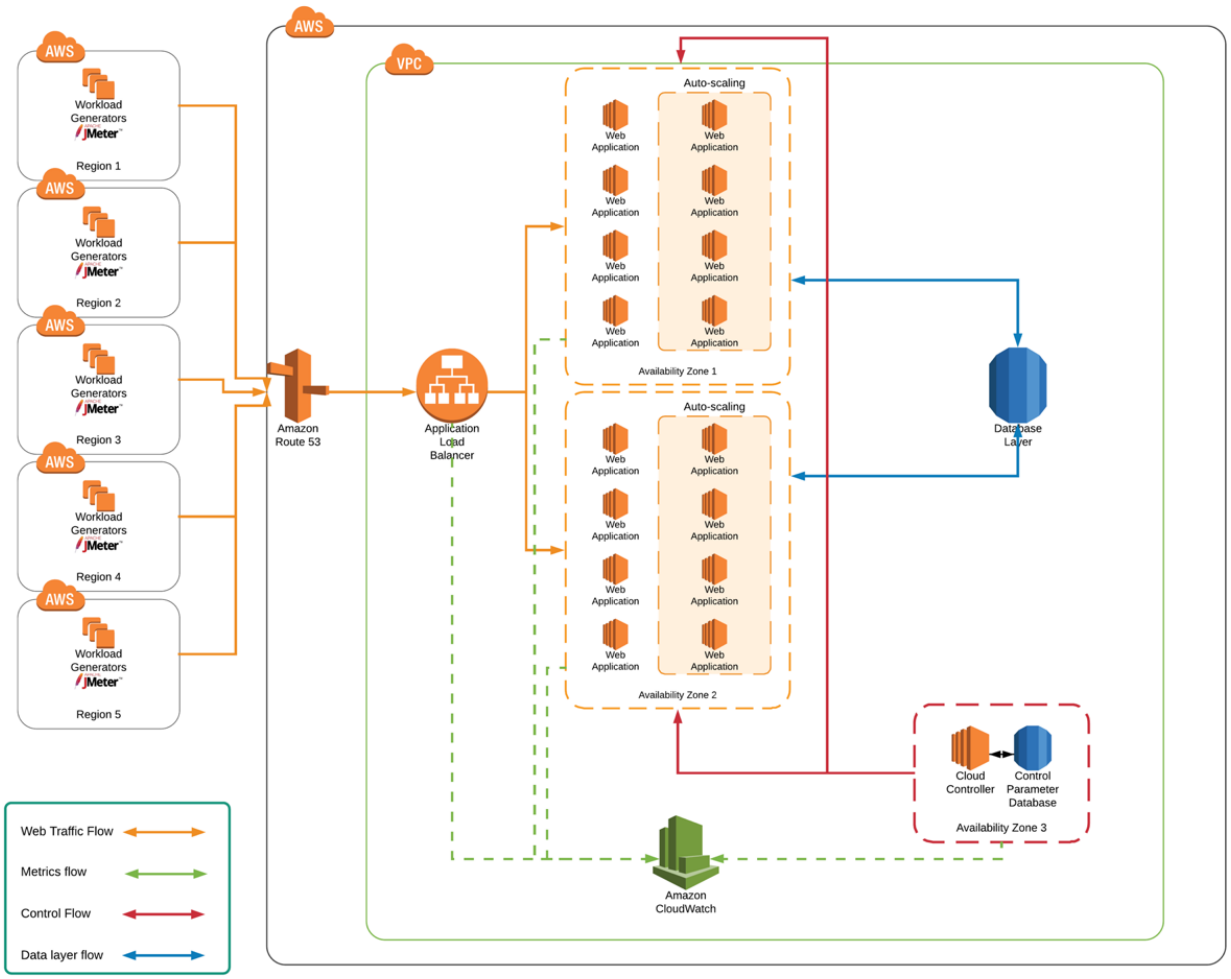


Figure 5.5: Experimental Framework

- Ultimate Thread Group (to initialize a sufficient amount of working threads).

2. *DNS and Load Balancing:* Amazon Route 53 is the DNS provider and AWS Application Elastic Load Balancer distributes the traffic between two availability zones.

3. *Web Application Worker instances:* evenly distributed cluster of EC2 instances in two different availability zones: eu-west-1b and eu-west-1c which are geographically isolated Amazon Data Centers.

4. *Metrics Collection:* AWS CloudWatch, a managed metrics collection and alerting service by Amazon, is used for the following metrics:

- Application Elastic Load Balancer: Average Target Response Time,
- EC2 worker instances with detailed monitoring: Average CPU usage per minute.

5. *AWS RDS instance:* It is used in the 3 tier web application experiments.

6. *Cloud Controller:* We create an EC2 instance for our controller with a MySQL database for controller configurations.

5.5.1 Experimental Setup

In this section, we present the setup within which we conducted our experimentations:

Web Service

We use Wikipedia English as our 3-tier web application example. The official data dump of Wikipedia from November 2017 was imported into a big AWS RDS MySQL instance. The database contains more than 15 million articles stored in an XML format. MediaWiki (Wikipedia's Official Web Application) is used as front and web service tier. It is a stateless PHP web application capable of scaling horizontally. In our experimentation, we don't serve the media files (photos, videos, etc.).

Instance Sizes

For the sake of good control performance, we use fine scaling granularity *i.e.* small size EC2 On-Demand instances: t2.medium (4GB RAM and 2 vCPU). Our workload generators are also t2.medium instances.

Web Service Response Time

Considering user's experience, we aim to keep the web service response time below 1 second.

Cluster Size

To have a consequent computing capacity capable of serving a high number of initial requests per second; we initialize our cluster with 10 instances. We set the maximum limit to 40 instances.

Load balancing limitations

AWS Application Elastic Load Balancer is a managed service capable of scaling automatically. For the scaling to be able to adjust to the extra traffic flow, Amazon recommends having maximum traffic ramp ups of +50% in a 5 minutes interval. We adjust our traffic generation script to meet this limitation.

Reference Setpoint

The reference setpoint is the average CPU usage of the web service worker nodes. We will try to track the 50% usage mark to keep the web service's performance optimal.

We introduce two different workloads:

1. A staircase traffic
2. A traffic with strong variations, created by condensing Wikipedia traffic of 120 hours in 2 hours.

In both cases, we send 1 million page-visit requests during a 2 hours experiments. In these experiments, we consider two different auto-scaling methods:

1. AWS auto-scaling with the Target Tracking algorithm
2. Model-Free Control

5.5.2 Experimental results

Static Cluster (without auto-scaling)

To begin we use the first workload on a static VM Cluster with a constant number of nodes (see Figures 5.10 and 5.11). We observe that at $M^h_{\text{act}}(t)$ =30, *i.e.* 30 VM, the cluster is oversized. The average CPU usage is lower than the reference (see sub-figure *Cluster Average CPU Usage*). We then change the VM count to 20, $M^h_{\text{act}}(t)$ =20 (see Figure 5.11). We observe that the CPU usage remains above the reference for three quarters of the experiment and saturates at the end. With the high CPU Usage, many request failures occur at this stage and the service becomes unavailable (failed requests are shown in orange in the sub-figure *Request Count*).

AWS Target Tracking

Figure 5.12 depicts the results using AWS Target Tracking algorithm (version of April 2018). The CPU Usage reference is set to 50 % and other parameters are set to their default values proposed by Amazon. This algorithm detects the first peak of traffic and increases the number of VM to the maximum allowed. When the traffic load is reduced the algorithm tends to keep the cluster at its high capacity for a long period of time before reducing the cluster size. The following facts should be highlighted:

- The user cannot change the sampling frequency and activation thresholds.
- Amazon does not detail the way they determine the number of VMs.

Model-Free Control

Figure 5.6 shows the results using the Model-Free Control algorithm. We observe the perfect adaptation of the cluster size $M^h_{\text{act}}(t)$ (sub-figure *VM count*) to the number of requests. The sub-figure *TargetResponseTime* describes a response time that is approximately constant, except when a peak load occurs for the first time. In figure 5.7, in sub-figure *Reference Tracking*, we observe a very good follow-up of the desired set-point. The blue curve represents the sum of the CPU usage measured on each VM (*cf.* (5.5)) and the green curve represents the desired setpoint (*cf.*(5.6)). The sub-figure *F estimated* depicts an excellent estimate of the workload shape.

To refine the confrontation with the model-free, in the second workload we introduce very brutal variations which are unusual in practice. According to the figures 5.8 and 5.9, the iP controller follows the load demands and best adapts the size of the cluster.

As mentioned before, Amazon Elastic Load Balancing *ELB*, of type *Application*[1] is designed to adapt to the requests arrival. However, there is a limitation: for a period of 5 minutes, traffic must not increase by more than 50 percent [2].

Comparisons

The table 5.2 compares the different methods discussed above by using the following *Key Performance Indicators* (*KPI*):

- Sum of the VM durations (in seconds) during each experimental test.

[1] See
`https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html`

[2] See
`https:///aws.amazon.com/articles/best-practices-in-evaluating-elastic-load-balancing`

Method	**Sum of the used VM durations (in seconds)**	**Average deviation of the CPU from the reference**
MFC (fig. 5.6)	127 920	8,53%
AWS Target Tracking (fig. 5.12)	187 080	21,73%
No elasticity, with 20 VM (fig. 5.11)	144 000	28,36%
No elasticity, with 30 VM (fig. 5.10)	216 000	21,78%

Table 5.2: Comparison of the different methods.

- Measurement deviation *i.e.* Average CPU Usage, with respect to set-point 50% (see the sub-figures: *Cluster Average CPU Usage*).

In the second column we observe that the autoscaling algorithm based on Model-Free Control significantly decreases the overall duration of VM instances used in a single experimentation (-31.6% compared to AWS Target Tracking). This measure directly determines the financial cost of a Cloud Infrastructure because the Computing Resources are billed in seconds of utilization. The third column shows how well the algorithm tracks the desired CPU set-point. We observe that the Model-Free Control auto-scaling algorithm shows less deviation from the target reference CPU usage than the AWS Target Tracking algorithm.

Remark 5.5 *Comparisons with other control laws, such as PID control, seem equally favourable to our approach. The lack of details in the publications, which we have been able to read, prevents us from saying more here.*

5.6 Closing remarks

In this Chapter, we presented our implementation of Model-Free Control to help overcome one of the main challenges of Cloud Computing, namely Auto-Scaling. The algorithm was applied on a popular Web Service (Wikipedia) using a real-life workload. Application possibilities are countless. For example, we can use our approach on a MapReduce compute cluster used for Big Data Analysis. The computing queries will represent Business Intelligence requests made by users on a dataset (see [15, 23]). Fault tolerance, a classical subject in control theory (see, for example, [17]), obviously appears in Cloud Computing (see, for example, [4]). Like any computer system, virtual machines are subject to failure. As the cardinality of VMs is the command, we know [48, 75] that the Model-Free Control easily overcomes

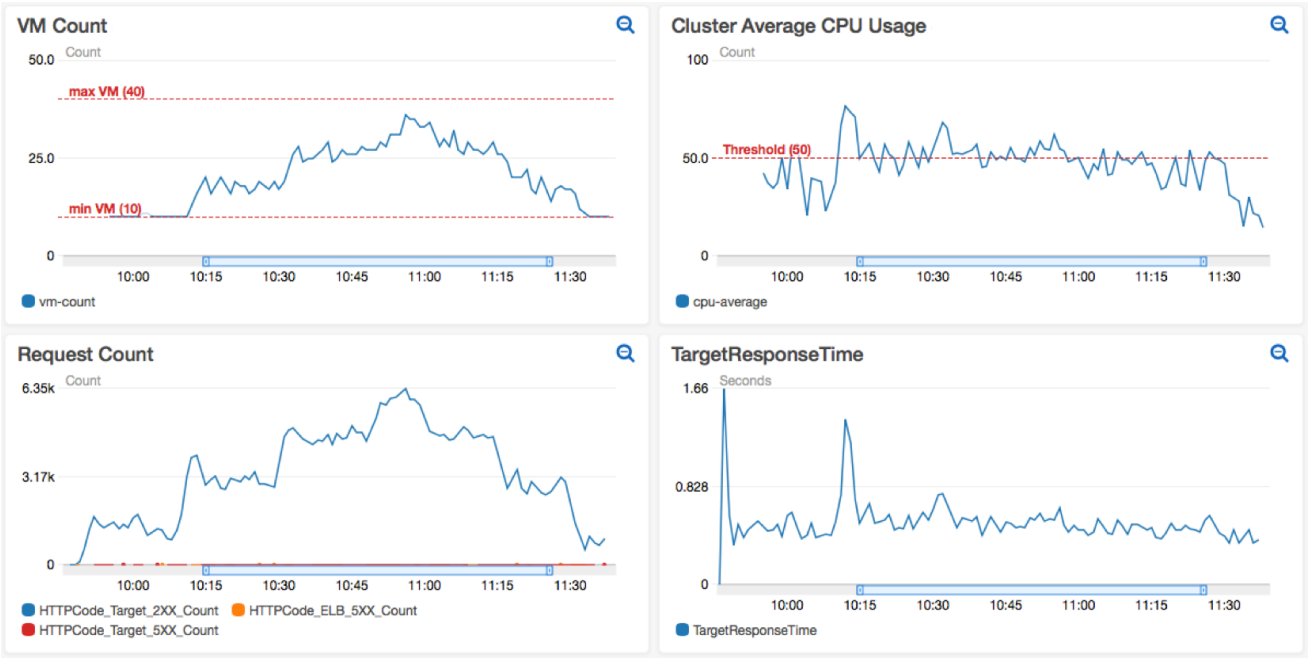


Figure 5.6: Experimental results when the AWS cluster is under **average** time-varing Wikipedia workload.

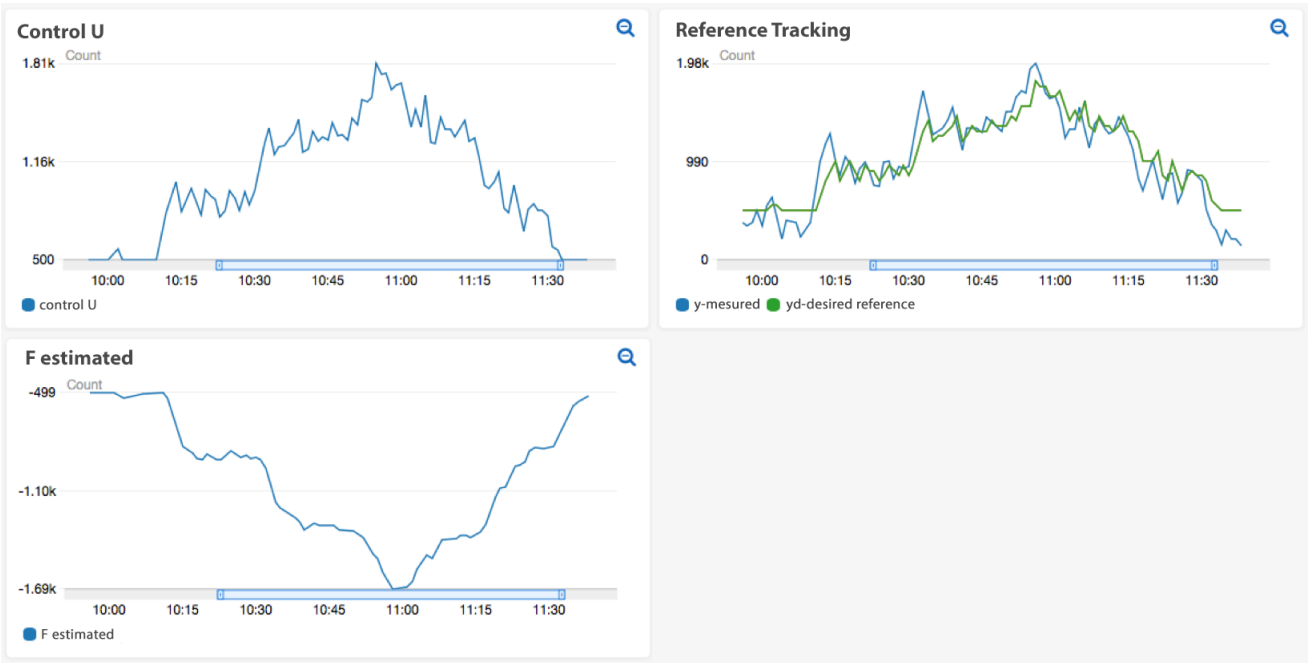


Figure 5.7: Experimental control design metrics when the AWS cluster is under **average** time-varing Wikipedia workload with $K_p = 0.8$, $\alpha = 1$ and $T = 1$.

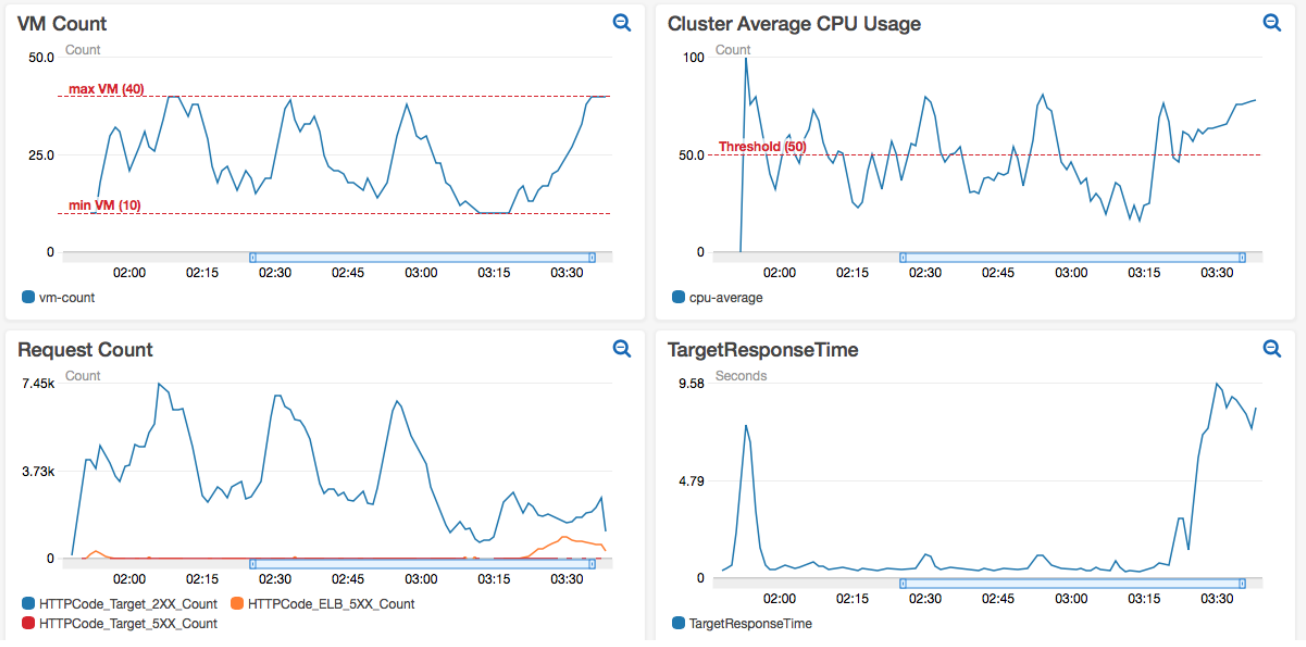


Figure 5.8: Experimental results when the AWS cluster is under **sharp** time-varing Wikipedia workload.

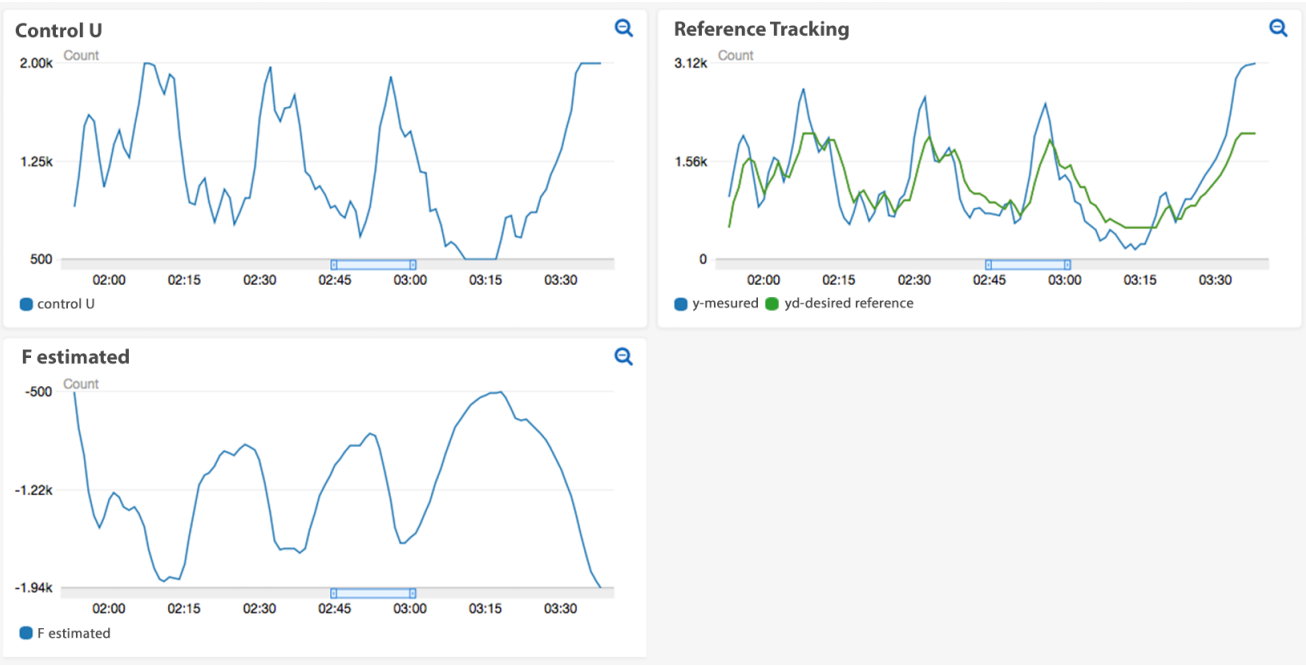


Figure 5.9: Experimental control design metrics when the AWS cluster is under **sharp** time-varing Wikipedia workload with $K_p = 0.8$, $\alpha = 1$ and $T = 1$.

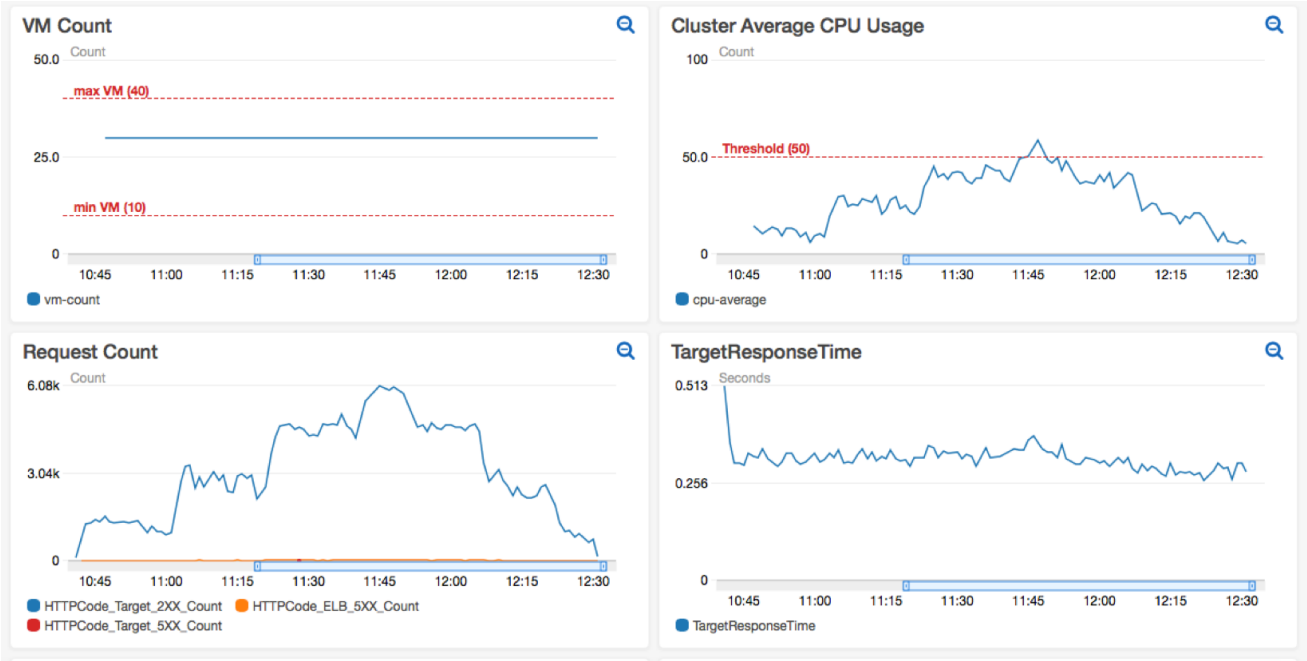


Figure 5.10: Experimental results when no elasticity method is used. The number of VMs is fixed to 30.

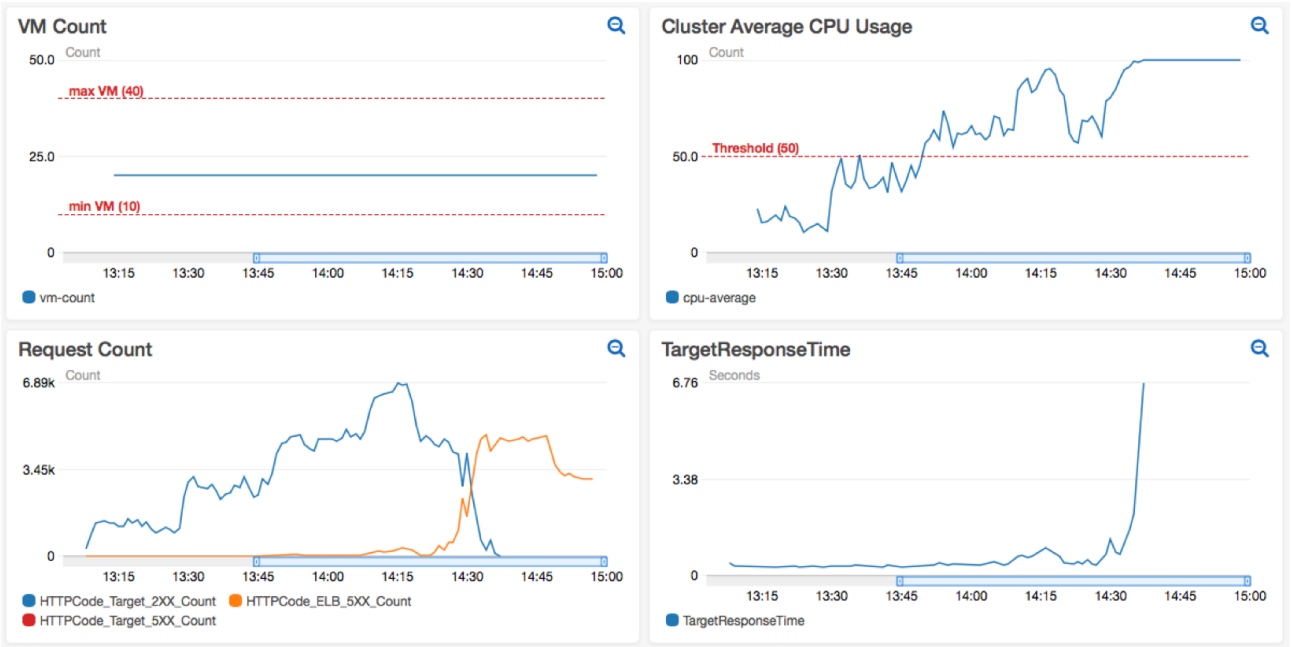


Figure 5.11: Experimental results when no elasticity method is used. The number of VMs is fixed to 20.

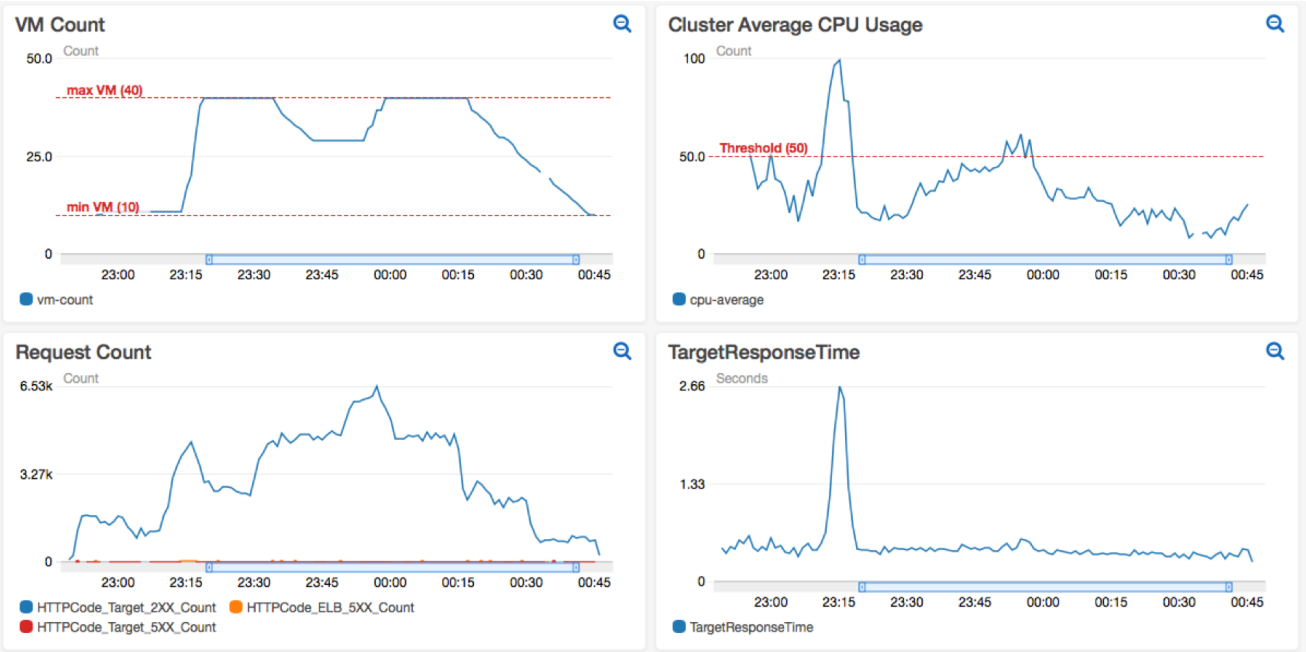


Figure 5.12: Experimental results with AWS Target Tracking Auto-scaling Algorithm.

these unpredictable events. Interestingly, our approach can be easily applied to auto-scale the use of containers (see, for example, [16, 110]). Popularized by Linux and Docker, containers are rapidly gaining ground in production environments. Note, for example, that [8] studies the elasticity by regulation from a simplified model. Many other topics, quite similar to this work, also deserve consideration. These works are the long-anticipated signal of Control Theory's prominent place in computer science, especially if one abandons the ambition, too often vain, of precise mathematical modeling, be it deterministic or probabilistic. May the mathematicians reassure themselves! Their role will not be diminished if they participate in the construction of new tools required, Model-Free Control is just one example among many others to develop if not to invent.

6 Conclusion and Future Works

We shall here state the obtained results and emphasize three aspects : at first some motivations, then the advocated method advantages and finally the limits and left open questions or further developments. In this thesis, we have presented two main parts. In the first part, we set the problem of finding the feasible reference trajectories for the control of differentially flat systems with constraints. In the second part, we proposed two applications of the Model-Free Control to the quadrotor system and the horizontal elasticity of Cloud Computing system.

6.1 Results summary and advantages

Motivation

- In Chapters 2, 3, we aim to fulfill the need for a systematic approach able to satisfy the system constraints directly in the control formulation.
- In Chapters 4, 5, we search for a feedback law able to control the system when the system model is poorly known or too complex to be modelled/identified. This controller should be robust to external disturbances, and this without any a priori knowledge on the latter.

Advantages

As far as Chapters 2 and 3 are concerned

- Finding a set of feasible reference trajectories (i.e. in the identified semi-algebraic set) automatically yields a fulfillment of the system's constraints.

Indeed, this constraints fulfillment is embedded in the computed semi-algebraic set whose structure is adapted to the differential flatness parametrization.

- The advocated scheme can deal with non-constant constraints on the inputs, states, and outputs, and, as well their derivatives. Moreover, the constraints can be any Bézier curve, which is able to approximate any nonlinear constraint.

- All the computations are made off-line, without involving sampling. This yields a very fast replanning procedure. Indeed, when one wishes to replan the trajectory, he can then pick another one from the semi-algebraic feasible trajectory set.

- When an exact symbolic solution (via the CAD) exists, the solution can be evaluated quickly for different numerical values.

As far as Chapters 4 and 5 are concerned:

- The proposed cascaded model-free approach for the quadrotor system is independent from the quadrotor physical parameters (mass, inertia, gyroscopic or aerodynamic effects). The quadrotor can follow aggressive maneuvers while tracking the yaw angle.

- Dealing with very complex computing systems remotely provided as a service, the proposed auto-scaling algorithm, enables Cloud Computing users to more efficiently manage costly resources by adjusting their computing power consumption to actual service demands. Our approach also allows Cloud Providers to better distribute their data centers computing power and therefore reduce electric power consumption.

6.2 Limits and further development tracks

As far as Chapters 2 and 3 are concerned:

- The proposed approach depends on the non-linear model complexity of the system. The flat output parametrization may be quite complex for some systems. A possible way of overcoming this limitation is to find a an approximative local linear model of the complex nonlinear model. The constrained trajectory method for such system can then be achieved in a linear setting.

- The use of Bézier curves has been investigated. It would be nice to have a scheme with piecewize Bézier curves (aka polynomial splines), the latter having local deformability wrt to the control points properties.
- The use of shape preserving splines (*e.g.* Akima, Hermite or Chebyshev splines) would also be worth investigating, through interpolating rather than approximating curve based schemes.

As far as Chapters 4 and 5 are concerned:

- A practical perspective on quadrotors is to test the very promising model free law on a real testbed.
- An application on popular Cloud Computing concepts like container systems would prove the efficiency of our approach on a wider scale.
- The model-free approach can further be applied on different levels of the Cloud Architecture, *e.g.,* the database layer or the load-balancing component.

6.3 Future work: Robust control of flat systems

Systems with constraints and uncertainties are considered for many real applications of different nature. In this Section, we will briefly describe the research line of the future works.

In real-time applications, usually, we have a system model, even though we know that the model is never a true description of the real process. This motivates us, in this final Chapter, to conclude with a perspective framework for nonlinear flat systems, where we use the known flat model while considering the uncertainties/disturbances. The stabilization law is using a flatness approach for the known part (presented in Chapter 2) and a model-free approach (presented in Chapter 4) for the unknown part.This kind of structure is the most common in real-time applications. This idea is not new and it has been already explored in the recent book [127] and in the recent papers [29], [97], [144], to name few. The advantage is that, in our proposal, we can also deal up with the system constraints. We would then have a very seducing alternative to the well spread MPC (Model Predictive Control) scheme. In the latter, there is no degree of freedom in the trajectory choice, since we are left with an optimization problem.

Consider the uncertain differentially flat system represented as:

$$y^{(k)} = f(y, \dot{y}, \ldots, y^{(\rho)}, u) + F \tag{6.1}$$

In a first time, the unknown part F will be considered as constant like in MFC. In a second time, for systems with slow dynamics, or for slow sensors, the unknown part F will be considered as an approximating or interpolating function, *e.g.* a spline. Since a (polynomial) spline is a piecewise polynomial function, the above approximation would be made on a finite time horizon.

References

[1] Hassane Abouaïssa, Michel Fliess, and Cédric Join. "On ramp metering: towards a better understanding of ALINEA via model-free control". In: *International Journal of Control* 90.5 (2017), pp. 1018–1026.

[2] Kostas Alexis, George Nikolakopoulos, and Anthony Tzes. "Model predictive quadrotor control : attitude , altitude". In: *IET Control Theory and Applications* 6.June 2011 (2012), pp. 1812–1827.

[3] Hirokazu Anai. "Effective quantifier elimination for industrial applications." In: *ISSAC*. 2014, pp. 18–19.

[4] Hamid Arabnejad et al. "A fuzzy load balancer for adaptive fault tolerance management in cloud platforms". In: *European Conference on Service-Oriented and Cloud Computing*. Springer, 2017, pp. 109–124.

[5] Michael Armbrust et al. "Above the clouds: A Berkeley view of cloud computing". In: *University of California, Berkeley, Tech. Rep. UCB* (2009), pp. 07–013. arXiv: 0521865719978052186571 5.

[6] Michael Armbrust et al. "A view of cloud computing". In: *Communications of the ACM* 53.4 (2010), p. 50. arXiv: 0521865719978052186571 5.

[7] Martin Bak. "Control of Systems with Constraints". PhD thesis. 2000, pp. 1–17.

[8] Luciano Baresi et al. "A discrete-time feedback controller for containerized cloud applications". In: *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2016, pp. 217–228.

[9] Cornel Barna et al. "Cloud adaptation with control theory in industrial clouds". In: *Proceedings - 2016 IEEE International Conference on Cloud Engineering Workshops, IC2EW 2016* (2016), pp. 231–238.

[10] Brian A. Barsky and Tony D. DeRose. "Geometric Continuity of Parametric Curves: Constructions of Geometrically Continuous Splines". In: *IEEE Computer Graphics and Applications* 10.1 (1990), pp. 60–68.

[11] Saugata Basu. "Algorithms in real algebraic geometry: a survey". In: *arXiv preprint arXiv:1409.1534* (2014).

[12] Maria Bekcheva, Cédric Join, and Hugues Mounier. "Cascaded Model-Free Control for trajectory tracking of quadrotors." In: *International Conference on Unmanned Aircraft Systems (ICUAS)*. 2018, pp. 1359–1368.

[13] Maria Bekcheva, Hugues Mounier, and Luca Greco. "Control of differentially flat linear delay systems with constraints. IFAC-PapersOnLine, 50(1), 13348-13353." In: *IFAC World Congress*. 2017, pp. 13348–13353.

[14] Maria Bekcheva et al. "Meilleure élasticité « nuagique » par commande sans modèle". In: *ISTE OpenScience Automatique* 2.1 (2018).

[15] Mihaly Berekmeri et al. “Feedback Autonomic Provisioning for Guaranteeing Performance in MapReduce Systems”. In: *IEEE Transactions on Cloud Computing* PP.99 (2016), p. 1.

[16] David Bernstein. “Containers and cloud: From lxc to docker to kubernetes”. In: *IEEE Cloud Computing* 1.3 (2014), pp. 81–84.

[17] Mogens Blanke et al. *Diagnosis and fault-tolerant control*. Vol. 2. Springer, 2006.

[18] Samir Bouabdallah, Andre Noth, and Roland Siegwart. “PID vs LQ control techniques applied to an indoor micro quadrotor”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vol. 3. 2004, pp. 2451–2456.

[19] Samir Bouabdallah and Roland Siegwart. “Backstepping and sliding-mode techniques applied to an indoor micro Quadrotor”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2005, pp. 2247–2252.

[20] Christopher W Brown. “An overview of QEPCAD B: a tool for real quantifier elimination and formula simplification”. In: *Journal of Japan Society for Symbolic and Algebraic Computation* 10.1 (2003), pp. 13–22.

[21] Ning Cao and Alan F. Lynch. “Inner-Outer Loop Control for Quadrotor UAVs with Input and State Constraints”. In: *IEEE Transactions on Control Systems Technology* 24.5 (2016), pp. 1797–1804.

[22] Garcia Carlos E., Prett David M., and Morari Manfred. “Model Predictive Control : Theory and Practice a Survey”. In: *Automatica* 25.3 (1989), pp. 335–338.

[23] Sophie Cerf et al. “Cost function based event triggered Model Predictive Controllers application to Big Data Cloud services”. In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. Las Vegas, 2016, pp. 1657–1662.

[24] Abbas Chamseddine et al. “Flatness-based trajectory planning/replanning for a quadrotor unmanned aerial vehicle”. In: *IEEE Transactions on Aerospace and Electronic Systems* 48.4 (2012), pp. 2832–2847.

[25] Abbas Chamseddine et al. “Trajectory Planning and Replanning Strategies Applied to a Quadrotor Unmanned Aerial Vehicle”. In: *Journal of Guidance, Control, and Dynamics* 35.5 (2012), pp. 1667–1671.

[26] Aneesh N Chand, Michihiro Kawanishi, and Tatsuo Narikiyo. “Non-linear model-free control of flapping wing flying robot using iPID”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 2930–2937.

[27] George E Collins. “Quantifier elimination for real closed fields by cylindrical algebraic decompostion”. In: *Automata Theory and Formal Languages 2nd GI Conference Kaiserslautern, May 20–23, 1975*. Springer, 1975, pp. 134–183.

[28] Giuseppe Conte and Anna Maria Perdon. “Modeling Time-delay Systems by means of Systems with Coefficients in a Ring”. In: *In Proceedings Workshop on Modeling and Control of Complex Systems*. 2005.

[29] John Cortés-romero et al. “Algebraic estimation and active disturbance rejection in the control of flat systems”. In: *Control Engineering Practice* 61.45 (2017), pp. 173–182.

[30] Michel Coste. *An introduction to semialgebraic geometry*. Université de Rennes, 2002.

[31] Fabrizio Dabbene, Didier Henrion, and Constantino M. Lagoa. "Simple approximations of semialgebraic sets and their applications to control". In: *Automatica* 78 (2017), pp. 110–118. arXiv: 1509.04200.

[32] Michel Dambrine, Jean-Pierre Richard, and Pierre Borne. "Feedback Control of Time-Delay Systems with Bounded Control and State". In: *Mathematical Problems in Engineering* 1.1 (1995), pp. 77–87.

[33] Brigitte D'Andréa-Novel et al. "A mathematical explanation via intelligent PID controllers of the strange ubiquity of PIDs". In: *18th Mediterranean Conference on Control and Automation (MED)*. 2010, pp. 395–400. arXiv: 1005.0440.

[34] Carl de Boor. *A Practical Guide to Splines*. Vol. 27. New York: Springer, 2001.

[35] Emmanuel Delaleau. "A proof of stability of model-free control". In: *2014 IEEE Conference on Norbert Wiener in the 21st Century (21CW)*. Boston, MA, 2014, pp. 1–7.

[36] Sette Diop. "Observers for sampled data nonlinear systems via numerical differentiation". In: *European Control Conference*. Kos, Greece, 2007, pp. 1179–1184.

[37] Maxime Doublet, Cédric Join, and Frédéric Hamelin. "Model-free control for unknown delayed systems". In: *Conference on Control and Fault-Tolerant Systems, SysTol*. 2016.

[38] Nadeem Faiz, Sunil Agrawal, and Richard M. Murray. "Differentially Flat Systems with Inequality Constraints: An Approach to Real-Time Feasible Trajectory Generation". In: *AIAA Journal of Guidance, Control, and Dynamics* 24.2 (2001), pp. 219–227.

[39] Guillermo P. Falconi et al. "Admissible thrust control laws for quadrotor position tracking". In: *2013 American Control Conference* July 2016 (201302), pp. 4844–4849.

[40] Rida T Farouki and V. T. Rajan. "Algorithms for polynomials in Bernstein form". In: *Computer Aided Geometric Design* 5.1 (1988), pp. 1–26.

[41] Matthias Fassler. "Quadrotor Control for Accurate Agile Flight". PhD thesis. University of Zurich, 2018.

[42] Timm Faulwasser, Veit Hagenmeyer, and Rolf Findeisen. "Optimal exact path-following for constrained differentially flat systems". In: *IFAC Proceedings Volumes (IFAC-PapersOnline)* 18.PART 1 (2011), pp. 9875–9880.

[43] Timm Faulwasser, Veit Hagenmeyer, and Rolf Findeisen. "Constrained reachability and trajectory generation for flat systems". In: *Automatica* 50.4 (2014), pp. 1151–1159.

[44] Michel Fliess. "Generalized controller canonical form for linear and nonlinear dynamics". In: *IEEE Transactions on Automatic Control* 35.9 (1990), pp. 994–1001.

[45] Michel Fliess. "Some basic structural properties of generalized linear systems". In: *Systems & Control Letters* 15 (1990), pp. 391–396.

[46] Michel Fliess. “Analyse non standard du bruit”. In: *Comptes Rendus Mathematique* 342.10 (2006), pp. 797–802. arXiv: `0603003 [cs.CE]`.

[47] Michel Fliess and Cédric Join. “Model-free control and intelligent PID controllers: Towards a possible trivialization of nonlinear control?” In: *IFAC Proceedings Volumes (IFAC-PapersOnline)* 15.PART 1 (2009), pp. 1531–1550.

[48] Michel Fliess and Cédric Join. “Model-free control”. In: *International Journal of Control* 86.12 (2013), pp. 2228–2252. arXiv: `1305.7085`.

[49] Michel Fliess and Cédric Join. “Deux améliorations concurrentes des PID Two competing improvements of PID controllers : A comparison”. In: *ISTE OpenScience Automatique* 2 (2018), pp. 1–23.

[50] Michel Fliess and Hugues Mounier. “Tracking control and π-freeness of infinite dimensional linear systems”. In: *Dynamical Systems, Control, Coding, Computer Vision*. Springer, 1999, pp. 45–68.

[51] Michel Fliess and Hugues Mounier. “On a class of linear delay systems often arising in practice”. In: *Kybernetika* 37.37 (2001), pp. 295–308.

[52] Michel Fliess and Hebertt Sira-Ramirez. “Closed-loop parametric identification for continuous-time linear systems via new algebraic techniques”. In: *Identification of Continuous-time Models from Sampled Data*. Springer London, 2008, pp. 362–391.

[53] Michel Fliess et al. “Easily implementable time series forecasting techniques for resource provisioning in cloud computing”. In: *6th International Conference on Control, Decision and Information Technologies (CoDIT)*. Paris.

[54] Michel Fliess et al. “On differentially flat nonlinear systems”. In: *Nonlinear Control Systems Design 1992*. Elsevier, 1993, pp. 159–163.

[55] Michel Fliess et al. “Flatness and defect of non-linear systems: introductory theory and examples”. In: *International Journal of Control* 61.6 (1995), pp. 1327–1361.

[56] Michel Fliess et al. “A lie-bäcklund approach to equivalence and flatness of nonlinear systems”. In: *IEEE Transactions on Automatic Control* 44.5 (1999), pp. 922–937.

[57] Melvin E. Flores and Mark B. Milam. “Trajectory generation for differentially flat systems via NURBS basis functions with obstacle avoidance”. In: *Proceedings of the American Control Conference* 2006 (2006), pp. 5769–5775.

[58] Alessandro Gasparetto and Vanni Zanotto. “A technique for time-jerk optimal planning of robot trajectories”. In: *Robotics and Computer-Integrated Manufacturing* 24.3 (2008), pp. 415–426.

[59] Pierre-Antoine. Gédouin et al. “Model-Free Control of Shape Memory Alloys Antagonistic Actuators”. In: *Preprints 17th IFAC World Congress*. Seoul, Korea, 2008.

[60] Pierre-Antoine Gédouin et al. “Experimental comparison of classical {PID} and model-free control: position control of a shape memory alloy active spring”. In: *Control Engineering Practice* 19.5 (2011), pp. 433–441.

[61] Nicole Gehring, Joachim Rudolph, and Frank Woittennek. *Controllability and prediction-free control of coupled transport processes viewed as linear systems with distributed delays*. Vol. 46. 26. IFAC, 2013, pp. 13–18.

[62] Knut Graichen and Michael Zeitz. “Feedforward control design for finite-time transition problems of non-linear MIMO systems under input constraints”. In: *International Journal of Control* 81.3 (2008), pp. 417–427.

[63] Knut Graichen and Michael Zeitz. “Feedforward Control Design for Finite-Time Transition Problems of Nonlinear Systems With Input and Output Constraints”. In: *IEEE Transactions on Automatic Control* 53.5 (2008), pp. 485–488.

[64] Domenico Grimaldi et al. “A feedback-control approach for resource management in public clouds”. In: *Global Communications Conference (GLOBECOM), IEEE*. 2015.

[65] Domenico Grimaldi et al. “A Fuzzy Approach based on Heterogeneous Metrics for Scaling Out Public Clouds”. In: *IEEE Transactions on Parallel and Distributed Systems* 9219.c (2017), pp. 1–1.

[66] Veit Hagenmeyer. “Robust nonlinear tracking control based on differential flatness”. In: *at-Automatisierungstechnik Methoden und Anwendungen der Steuerungs-, Regelungs-und Informationstechnik* 50.12/2002 (2002), p. 615.

[67] Veit Hagenmeyer and Emmanuel Delaleau. “Exact feedforward linearization based on differential flatness”. In: *International Journal of Control* 76.6 (2003), pp. 537–556.

[68] Veit Hagenmeyer and Emmanuel Delaleau. “Continuous-time non-linear flatness-based predictive control: an exact feedforward linearisation setting with an induction drive example”. In: *International Journal of Control* 81.10 (2008), pp. 1645–1663.

[69] Veit Hagenmeyer and Emmanuel Delaleau. “Robustness analysis with respect to exogenous pertubations for flatness-based exact feedforward linearization”. In: *IEEE Transactions on Automatic Control* 55.3 (2010), pp. 727–731.

[70] Jean-Claude Hennet and Sophie Tarbouriech. “Stability conditions of constrained delay systems via positive invariance”. In: *International Journal of Robust and Nonlinear Control* 8.3 (1998), pp. 265–278.

[71] Gabriel M. Hoffmann et al. “Quadrotor helicopter flight dynamics and control: Theory and experiment”. In: *American Institute of Aeronautics and Astronautics* (2007), pp. 1–20.

[72] Haomiao Huang et al. “Aerodyamics and Control of Automous Quadrotor Helicopters in Aggressive Maneuvering”. In: *ICRA2009* (2009), pp. 3277–3282.

[73] Cedric Join, Frédéric Chaxel, and Michel Fliess. “"Intelligent" controllers on cheap and small programmable devices”. In: *2nd International Conference on Control and Fault-Tolerant Systems (SysTol’13)*. Nice, 2013.

[74] Manuel Kauers. “How to use cylindrical algebraic decomposition”. In: *Seminaire Lothraringien* 65.2011 (2011), pp. 1–16.

[75] Frédéric Lafont et al. “A model-free control strategy for an experimental greenhouse with an application to fault accommodation”. In: *Computers and Electronics in Agriculture* 110 (2015), pp. 139–149.

[76] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[77] Harold C. Lim et al. “Automated control in cloud computing”. In: *Proceedings of the 1st workshop on Automated control for datacenters and clouds - ACDC ’09* (2009), p. 13.

[78] Stephen R Lindemann and Steven M Lavalle. “Computing Smooth Feedback Plans Over Cylindrical Algebraic Decompositions”. In: *Robotics:Science and Systems*. Philadelphia, USA, 2006.

[79] W. Van Loock, Goele Pipeleers, and Jan Swevers. “B-spline parameterized optimal motion trajectories for robotic systems with guaranteed constraint satisfaction”. In: *Mechanical Sciences* 6.2 (2015), pp. 163–171.

[80] Christophe Louembet, Franck Cazaurang, and Ali Zolghadri. “Motion planning for flat systems using positive B-splines: An LMI approach”. In: *Automatica* 46.8 (2010), pp. 1305–1309.

[81] David Lutterkort. “Envelopes of Nonlinear Geometry”. PhD thesis. Purdue University, 1999.

[82] Teppo Luukkonen. “Modelling and Ccontrol of Quadcopter”. In: *Independent research project in applied mathematics, Espoo.* (2011).

[83] Tom Lyche and Knut Morken. “Spline Methods”. In: (2002).

[84] Victor Magron, Didier Henrion, and Jean-Bernard Lasserre. “Semidefinite approximations of projections and polynomial images of semialgebraic sets”. In: *SIAM Journal on Optimization* 25.4 (2015), pp. 2143–2164.

[85] Philipp Mai, Cédric Join, and Johan Reger. “Flatness-based fault tolerant control of a nonlinear MIMO system using algebraic derivative estimation To cite this version :” in: *3rd IFAC Symposium on System, Structure and Control*. 2007.

[86] Ming Mao, Jie Li, and Marty Humphrey. “Cloud auto-scaling with deadline and budget constraints”. In: *Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on.* 2010.

[87] Reble Marcus et al. “Model predictive control of constrained non-linear time-delay systems”. In: *IMA Journal of Mathematical Control and Information* (2010).

[88] Philippe Martin, Pierre Rouchon, and Richard M. Murray. *Flat systems, equivalence and trajectory generation*. 3rd cycle. 2006, p. 81.

[89] David Q. Mayne et al. “Constrained model predictive control: Stability and optimality”. In: *Automatica* 36.6 (2000), pp. 789–814.

[90] Mamadou Mboup, Cédric Join, and Michel Fliess. “Numerical differentiation with annihilators in noisy environment”. In: *Numerical Algorithms- Springer Verlag* 4.50 (2009), pp. 1–27.

[91] Daniel Mellinger and Vijay Kumar. “Minimum snap trajectory generation and control for quadrotors”. In: *Proceedings - IEEE International Conference on Robotics and Automation* (2011), pp. 2520–2525.

[92] Daniel Mellinger, Nathan Michael, and Vijay Kumar. “Trajectory generation and control for precise aggressive maneuvers with quadrotors”. In: *Springer Tracts in Advanced Robotics* 79.2009 (2014), pp. 361–373.

[93] Lghani Menhour et al. “Multivariable decoupled longitudinal and lateral vehicle control: A model-free design”. In: *Proceedings of the IEEE Conference on Decision and Control* (2013), pp. 2834–2839.

[94] Lghani Menhour et al. “An Efficient Model-Free Setting for Longitudinal and Lateral Vehicle Control: Validation Through the Interconnected Pro-SiVIC/RTMaps Prototyping Platform”. In: *IEEE Transactions on Intelligent Transportation Systems* 19.2 (2017), pp.461–475. arXiv: 1705.03216.

[95] Mark B Milam. “Real-Time Optimal Trajectory Generation for Constrained Dynamical Systems”. PhD thesis. 2003.

[96] Taghreed MohammadRidha et al. “A Variable Reference Trajectory for Model-Free Glycemia Regulation”. In: *2015 Proceedings of the Conference on Control and its Applications* (2015), pp. 60–67.

[97] Rafael Morales, Hebertt Sira-ramirez, and J. A. Somolinos. “Robust control of underactuated wheeled mobile manipulators using GPI disturbance observers”. In: *Multibody System Dynamics* 32.4 (2014), pp. 511–533.

[98] Knut Mørken. “Some identities for products and degree raising of splines”. In: *Constructive Approximation* 7.1 (1991), pp. 195–208.

[99] Hugues Mounier. “Propriétés structurelles des systèmes linéaires à retards: aspects théoriques et pratiques”. PhD thesis. Université Paris–Sud, Orsay, 1995.

[100] Hugues Mounier and Luca Greco. “Modelling and structural properties of distributed parameter wind power systems”. In: *Proceedings of the 22nd International Symposium on Mathematical Theory of Networks and Systems (MTNS)*. 2016.

[101] Hugues Mounier, Pierre Rouchon, and Joachim Rudolph. “Some examples of linear systems with delays”. In: *Journal européen des systèmes automatisés* 31.6 (1997), pp. 911–925.

[102] Hugues Mounier and Joachim Rudolph. “Flatness-based control of nonlinear delay systems: A chemical reactor example”. In: *International Journal of Control* 71.5 (1998), pp. 871–890.

[103] Charifa Moussaoui, Rosa Abbou, and Jean Jacques Loiseau. “Controller Design for a Class of Delayed and Constrained Systems: Application to Supply Chains”. In: *Low-Complexity Controllers for Time-Delay Systems*. Ed. by Alexandre Seuret et al. Springer International Publishing, 2014, pp. 61–75.

[104] David Nairn, Jörg Peters, and David Lutterkort. “Sharp , quantitative bounds on the distance between a polynomial piece and its Bézier control polygon”. In: *Computer Aided Geometric Design* 16 (1999), pp. 613–631.

[105] Ibrahima N’Doye et al. “Intelligent Proportional-Integral-Derivative Control-Based Modulating Functions for Laser Beam Pointing and Stabilization”. In: *IEEE Transactions on Control Systems Technology* PP (2019), pp. 1–8.

[106] Ibrahima N’Doye et al. “Intelligent proportional-integral-derivative control-based modulating functions for laser beam pointing and stabilization”. In: *IEEE Transactions on Control Systems Technology* (2019).

[107] Sorin Olaru and Silviu-iulian Niculescu. “Predictive control for linear systems with delayed input subject to constraints”. In: *IFAC Proceedings Volumes*. 2008, pp. 11208–11213.

[108] Johannes Oldenburg and Wolfgang Marquardt. “Flatness and higher order differential model representations in dynamic optimization”. In: *Computers and Chemical Engineering* 26.3 (2002), pp. 385–400.

[109] Pradeep Padala et al. “Adaptive control of virtualized resources in utility computing environments”. In: *In Proceedings of the European Conference on Computer Systems* (2007), pp. 289–302.

[110] Claus Pahl et al. “Cloud container technologies: a state-of-the-art review”. In: *IEEE Transactions on Cloud Computing* (2017).

[111] Nicolas Petit, Mark B Milam, and Richard M Murray. “Inversion based constrained trajectory optimization”. In: *Proc. of the 5th IFAC Symposium on Nonlinear Control Systems* (2001), pp. 1–6.

[112] P. Picard, O. Sename, and J-F Lafay. “Weak controllability and controllability indices for linear neutral systems”. In: *Mathematics and Computers in Simulation* 45.3-4 (1998), pp. 223–233.

[113] Les Piegl and Wayne Tiller. “Software-engineering approach to degree elevation of B-spline curves”. In: *Computer-Aided Design* 26.1 (1994), pp. 17–28.

[114] Philip Polack et al. “Finite-Time Stabilization of Longitudinal Control for Autonomous Vehicles via a Model-Free Approach”. In: *IFAC World Congress*. 2017. arXiv: `1704.01383`.

[115] Paul Pounds, Robert Mahony, and Peter Corke. “Modelling and control of a large quadrotor robot”. In: *Control Engineering Practice* 18.7 (2010), pp. 691–699.

[116] Hartmut Prautzsch, Wolfgang Boehm, and Marco Paluszny. *Bézier and B-spline techniques*. Springer Science & Business Media, 2002.

[117] Mohammadreza Radmanesh and Manish Kumar. “Flight formation of UAVs in presence of moving obstacles using fast-dynamic mixed integer linear programming”. In: *Aerospace Science and Technology* 50.December (2016), pp. 149–160.

[118] Stefan Ratschan. “Applications of Quantified Constraint Solving over the Reals Bibliography”. In: *ArXiv* (2012), pp. 1–13. arXiv: `arXiv:1205.5571v1`.

[119] Jean-Pierre Richard. “Time-delay systems: an overview of some recent advances and open problems”. In: *Automatic Control, IEEE Transactions on* 39.10 (2003), pp. 1667–1694.

[120] Mike Roberts and Pat Hanrahan. “Generating Dynamically Feasible Trajectories for Quadrotor Cameras”. In: *Proc. of Siggraph ’16* (2016), 61:1–61:11.

[121] Anand Sanchez-Orta et al. “Position-Yaw Tracking of Quadrotors”. In: *Journal of Dynamic Systems, Measurement, and Control* 137.6 (2015), p. 061011.

[122] Olivier Sename, Rabah Rabah, and Jean-François Lafay. “Decoupling without prediction of linear systems with delays: A structural approach”. In: *Systems & Control Letters* 25 (1995), pp. 387–395.

[123] Rodolphe Sepulchre, Mrdjan Jankovic, and Petar Kokotovic. *Constructive Nonlinear Control*. Springer Science & Business Media, 1996, pp. 1–315. arXiv: 9809069v1 [arXiv:gr-qc].

[124] Rifat Sipahi and Silviu-Iulian Niculescu. "Stability of car following with human memory effects and automatic headway compensation." In: *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences* 368.1928 (2010), pp. 4563–83.

[125] Hebertt Sira-Ramirez. "On the linear control of the quad-rotor system". In: *Proceedings of the 2011 American Control Conference*. 2011, pp. 3178–3183.

[126] Hebertt Sira-Ramirez and Sunil K Agrawal. *Differentially flat systems*. CRC Press, 2004.

[127] Hebertt Sira-Ramirez et al. *Active Disturbance Rejection Control of Dynamic Systems*. First edit. 2017.

[128] Florin Stoican, Vlad-Mihai Ivanusca, and Ionela Prodan. "Obstacle avoidance via B-spline parameterizations of flat trajectories". In: 1 (2016), pp. 1002–1007. arXiv: 1603.04911.

[129] Florin Stoican, Ionela Prodan, and Dan Popescu. "Flat trajectory generation for way-points relaxations and obstacle avoidance". In: *2015 23rd Mediterranean Conference on Control and Automation, MED 2015 - Conference Proceedings* (2015), pp. 695–700.

[130] Adam W. Strzebonski. "Cylindrical Algebraic Decomposition using validated numerics". In: *Journal of Symbolic Computation* 41.9 (2006), pp. 1021–1038.

[131] Fajar Suryawan, José De Dona, and Maria Seron. "Splines and polynomial tools for flatness-based constrained motion planning". In: *International Journal of Systems Science* 43.8 (2012), pp. 1396–1411.

[132] Alfred Tarski. "A decision method for elementary algebra and geometry". In: *Quantifier elimination and cylindrical algebraic decomposition*. Springer, 1998, pp. 24–84.

[133] Didier Theilliol et al. "Actuator fault-tolerant control design based on reconfigurable reference input". In: *International Journal of Applied Mathematics and Computer Science, De Gruyter* 18.4 (2008), pp. 553–560.

[134] Amjad Ullah et al. "A control theoretical view of cloud elasticity: taxonomy, survey and challenges". In: *Cluster Computing* 21.4 (2018), pp. 1735–1764.

[135] Jorge Villagra et al. "Robust grey-box closed-loop stop-and-go control To cite this version : HAL Id : inria-00319591 Robust grey-box closed-loop stop-and-go control". In: (2008).

[136] Johannes von Löwis and Joachim Rudolph. "Real-time trajectory generation for flat systems with constraints". In: *Nonlinear and Adaptive Control*. 2002, pp. 385–394.

[137] Chen Wang et al. "Trajectory Tracking Control for Quadrotor Robot Subject to Payload Variation and Wind Gust Disturbance". In: *Journal of Intelligent and Robotic Systems: Theory and Applications* 83.2 (2016), pp. 315–333.

[138] Haoping Wang et al. “Model-free-based terminal SMC of quadrotor attitude and position”. In: *IEEE Transactions on Aerospace and Electronic Systems* 52.5 (2016), pp. 2519–2528.

[139] Jing Wang et al. “Event-driven model-free control in motion control with comparisons”. In: *IMA Journal of Mathematical Control and Information* 34.4 (2016), pp. 1255–1275.

[140] David J. Wilson et al. “Cylindrical Algebraic Sub-Decompositions”. In: *Mathematics in Computer Science* 8.2 (2014), pp. 263–288.

[141] Kosaku Yosida. *Operational Calculus: A Theory of Hyperfunctions*. 1984.

[142] Younes Al Younes et al. “Robust Model-Free Control Applied to a Quadrotor UAV”. In: *Journal of Intelligent and Robotic Systems: Theory and Applications* 84.1-4 (2016), pp. 37–52.

[143] Jing Yu, Zhihao Cai, and Yingxun Wang. “Minimum jerk trajectory generation of a quadrotor based on the differential flatness”. In: *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference*. IEEE, 2014, pp. 832–837.

[144] Eric William Zurita-Bustamante, Alberto Luviano-Juárez, and Hebertt Sira-Ramirez. “On the Robust Flat-Filtering Control of MIMO nonlinear systems : The PMSM Experimental Case Study”. In: *American Control Conference*. 2018, pp. 6755–6760.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Commande de systèmes plats avec contraintes et Applications de la Commande sans Modèle aux quadrotors et au Cloud Computing

Mots clés : Platitude différentielle, Commande sans modèle, Commande des systèmes avec contraintes, Quadrotors, Cloud Computing.

Résumé : La première partie de la thèse est consacrée à la commande avec contraintes de systèmes différentiellement plats. Deux types de systèmes sont étudiés : les systèmes non linéaires de dimension finie et les systèmes linéaires à retards. Nous présentons une approche unifiée pour intégrer les contraintes d'entrée/état/sortie dans la planification des trajectoires. Pour cela, nous spécialisons les sorties plates (ou les trajectoires de référence) sous forme de courbes de Bézier. En utilisant la propriété de platitude, les entrées/états du système peuvent être exprimés sous la forme d'une combinaison de sorties plates (courbes de Bézier) et de leurs dérivées. Par conséquent, nous obtenons explicitement les expressions des points de contrôle des courbes de Bézier d'entrées/états comme une combinaison des points de contrôle des sorties plates. En appliquant les contraintes souhaitées à ces derniers points de contrôle, nous trouvons les régions faisables pour les points de contrôle de Bézier de sortie, c'est-à-dire un ensemble de trajectoires de référence faisables. Ce cadre permet d'éviter le recours, en général fort coûteux d'un point de vue informatique, aux schémas d'optimisation.

Pour résoudre les incertitudes liées à l'imprécision de l'identification et modélisation des modèles et les perturbations, nous utilisons la commande sans modèle (Model Free Control-MFC) et dans la deuxième partie de la thèse, nous présentons deux applications démontrant l'efficacité de notre approche :

1. Nous proposons une conception de contrôleur qui évite les procédures d'identification du système du quadrotor tout en restant robuste par rapport aux perturbations endogènes (la performance de contrôle est indépendante de tout changement de masse, inertie, effets gyroscopiques ou aérodynamiques) et aux perturbations exogènes (vent, bruit de mesure). Pour atteindre notre objectif en se basant sur la structure en cascade d'un quadrotor, nous divisons le système en deux sous-systèmes de position et d'attitude contrôlés chacun indépendamment par la commande sans modèle de deuxième ordre dynamique. Nous validons notre approche de contrôle avec trois scénarios réalistes : en présence d'un bruit inconnu, en présence d'un vent variant dans le temps et en présence des variations inconnues de masse, tout en suivant des manœuvres agressives.
2. Nous utilisons la commande sans modèle et les correcteurs « intelligents » associés, pour contrôler (maintenir) l'élasticité horizontale d'un système de Cloud Computing. Comparée aux algorithmes commerciaux d'Auto-Scaling, notre approche facilement implémentable se comporte mieux, même avec de fluctuations aigües de charge. Ceci est confirmé par des expériences sur le cloud public Amazon Web Services (AWS).

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Title : Flatness-based Constrained Control and Model-Free Control Applications to Quadrotors and Cloud Computing

Keywords : Differential Flatness, Model-Free Control, Constrained Control, Quadrotors, Cloud Computing.

Abstract : The first part of the thesis is devoted to the control of differentially flat systems with constraints. Two types of systems are studied: non-linear finite dimensional systems and linear time-delay systems. We present an approach to embed the input/state/output constraints in a unified manner into the trajectory design for differentially flat systems. To that purpose, we specialize the flat outputs (or the reference trajectories) as Bézier curves. Using the flatness property, the system's inputs/states can be expressed as a combination of Bézier curved flat outputs and their derivatives. Consequently, we explicitly obtain the expressions of the control points of the inputs/states Bézier curves as a combination of the control points of the flat outputs. By applying desired constraints to the latter control points, we find the feasible regions for the output Bézier control points i.e. a set of feasible reference trajectories. This framework avoids the use of generally high computing cost optimization schemes.

To resolve the uncertainties arising from imprecise model identification and the unknown pertubations, we employ the Model-Free Control (MFC) and in the second part of the thesis we present two applications demonstrating the effectiveness of our approach:

1. We propose a controller design that avoids the quadrotor's system identification procedures while staying robust with respect to the endogenous (the control performance is independent of any mass change, inertia, gyroscopic or aerodynamic effects) and exogenous disturbances (wind, measurement noise). To reach our goal, based on the cascaded structure of a quadrotor, we divide the system into positional and attitude subsystems each controlled by an independent Model-Free controller of second order dynamics. We validate our control approach in three realistic scenarios: in presence of unknown measurement noise, with unknown time-varying wind disturbances and mass variation while tracking aggressive manoeuvres.

2. We employ the Model-Free Control to control (maintain) the "horizontal elasticity" of a Cloud Computing system. When compared to the commercial "Auto-Scaling" algorithms, our easily implementable approach behaves better, even with sharp workload fluctuations. This is confirmed by experiments on the Amazon Web Services (AWS) public cloud.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France