Real-Time Trajectory Generation for Constrained Nonlinear Dynamical Systems Using Non-Uniform Rational B-spline Basis Functions

Thesis by
Melvin E. Flores

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

2007

(Defended November 12, 2007)

To my grandpa

Preface

This thesis was submitted at the California Institute of Technology on February 19, 2008, in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Control and Dynamical Systems. The thesis was defended on November 12, 2007, in Pasadena, CA, and was approved by the following thesis committee:

- Richard M. Murray (Chair), Control and Dynamical Systems and Mechanical Engineering,
  California Institute of Technology,
- Jerrold E. Marsden, Control and Dynamical Systems,
  California Institute of Technology,
- John Doyle, Control and Dynamical Systems, Electrical Engineering and Bioengineering
  California Institute of Technology,
- Mark B. Milam, GNC Engineer
  Northrop Grumman Space Technology,

Acknowledgements

I would like to express my most profound gratitude to my advisor, Richard Murray. His guidance during my tenure at CDS has been invaluable. I am thankful for his support and for giving me this opportunity that has allowed me to grow both personally and academically. In addition, I would like to thank Mark Milam for giving me the opportunity to work for Northrop Grumman Space Technology and for his direction and advise while stationed there; much of this work would not have been possible without our discussions and his insights. I extend my thanks to Joel Burdick for his advise and for indulging me while inquiring him about motion planning methodologies. I am also grateful to the members of my thesis committe — Richard Murray (chair), Jerrold Marsden, John Doyle, and Mark Milam. To my classmates — Xin Liu, Domitilla del Vecchio, Melvin Leok, Harish Bhat, Shreesh Mysore, Antonis Papachristodoulo, and Stephen Prajna, for sharing the journey of graduate school with its good and tough times. My thanks to the staff at Caltech, specially Gloria Bain. On a personal level, this work would not have been possible without the love and support of my wife. Mali, you are by far the best thing that has ever happened to my life. I love you dearly and our soon-to-come little baby girl, Sofia Melina. Lastly, I thank my families (Yes! I have been blessed with more than one) — Contreras, Sirolli, Aquino, and Flores for their caring and understanding through the years.

Real-Time Trajectory Generation
for Constrained Nonlinear Dynamical Systems Using
Non-Uniform Rational B-spline Basis Functions

by

Melvin E. Flores

In Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

Abstract

The thesis describes a new method for obtaining minimizers for optimal control problems whose minima serve as control policies for guiding nonlinear dynamical systems to achieve prescribed goals under imposed trajectory and actuator constraints. One of the major contributions of the present work resides in the approximation of such minimizers by piecewise polynomial functions expressed in terms of a linear combination of non-uniform rational B-spline (NURBS) basis functions and the judicious exploitation of the properties of the resulting NURBS curves to improve the computational effort often associated with solving optimal control problems for constrained dynamical systems.

In particular, by exploiting the two structures combined in a NURBS curve, NURBS basis functions and an associated union of overlapping polytopes constructed from the coefficients of the linear combination, we are able to separate an optimal control problem into two subproblems — guidance and obstacle-avoidance, making the original problem tractable. This is accomplished by laying out the union of overlapping polytopes in such a way that they delineate a section of space that avoids all obstacles and then manipulating the NURBS basis functions to obtain trajectories that are guaranteed to remain bounded by this section of space without explicitly including the conjunction of disjunctions naturally induced from obstacles into the guidance problem.

In addition, we show how one can construct systematically a feasible region that corresponds to a NURBS parameterization starting from an ordered union of pairwise adjacently overlapping nonempty compact convex sets. Specifically, we show how to setup a nonlinear programming problem to solve for the feasible region in terms of an ordered union of pairwise adjacently overlapping polytopes with nonempty interiors by maximizing the sum of their volumes and starting from a feasible region described by an ordered union of pairwise adjacently overlapping nonempty convex compact simi-algebraic sets. Finally, we show how this strategy can be implemented practically for an autonomous system traversing an urban environment.

Finally, this work culminated in the filing of patent US20070179685 on behalf of Northrop Grumman for the Space Technology sector and in the development of the NURBSbasedOTG software package. This C++ package contains the theoretical results of this thesis in the form of an object-oriented implementation optimized for real-time trajectory generation.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Many control policies used for guiding real systems are obtained through the numerical computation of minimizers to *optimal control* problems. In general, determining a minimizer to an optimal control problem consists of computing the control and state evolutions that achieve the lowest value of a cost functional that captures the effort in the system's motions (e.g., energy, momentum, entropy) while taking into account the system's behavior (e.g., linear, nonlinear, hybrid), boundary conditions (e.g., initial and/or final system's configuration — both position and attitude), trajectory constraints (e.g., obstacle avoidance), and actuator constraints (e.g., limits in forces and torques). The appeal for this paradigm resides, in part, on the ease with which a large number of useful engineering problems are able to be modeled by using this mathematical framework and on the ever increasing theoretical and computational resources that are at our disposal for solving them.

The mathematical structure of the optimal control problem dictates how deep we can proceed into understanding the relevant issues that define it as well as the set of analytical and/or numerical tools available for obtaining its minimizers. In general, three major issues can be resolved regarding a posed optimal control problem, depending on its complexity: existence of minima, Cesari [1983]; computation of minimizers, Bryson and Ho [1975], Pontryagin et al. [1962], Pesch [1989], von Stryk and Bulirsch [1992]; and on-board implementation of control policies for guiding a real system, Betts [1998], Diehl [2001], Milam [2003]. In particular, if an optimal control problem is made up of a quadratic cost functional, linear dynamics, and no trajectory nor actuator constraints, one can obtain minimizers analytically (or numerically with very low computational effort) in the form of feedback laws, and they can be implemented (if physically realizable) on a real system to guide its behavior in real time. On the other hand, if the optimal control problem is made up of a non-convex

cost functional, nonlinear dynamics, non-convex trajectory constraint sets, and compact actuator constraint sets, our only resort is to pursue numerical computation of open-loop control policies that may or may not exist. A feedback law can be constructed using the open-loop control policies by exploiting *receding horizon control* (RHC) results, Mayne et al. [2000], Chen and Allgower [1998]. However, in this case, implementation on board a real system would require significant computational effort and sophisticated decision trees to deal with contingencies (e.g., failure to obtain a minimizer).

Numerical methods for generating minimizers to optimal control problems may be classified into two general approaches: direct and indirect methods. Indirect methods are developed through the *calculus of variations* and arise in the form of first-order and second-order necessary and sufficient conditions of optimality, Bryson and Ho [1975], Pontryagin et al. [1962], Pesch [1989], von Stryk and Bulirsch [1992]. Direct methods, on the other hand, rely on the direct transcription of the optimal control problem to a *nonlinear programming* problem via parameterization of the inputs and states, followed by a balanced discretization, Betts [1998], von Stryk and Bulirsch [1992], Kraft [1985]. The reader is referred to Betts [1998] for a survey on numerical methods for trajectory generation, weighing their pros and cons. Finally, a combination of the two approaches above has also been studied with relative success, Gath [2002], von Stryk and Bulirsch [1992].

Despite the large number of methods developed for trajectory generation, many issues still remain to be resolved either partially or in their entirety (e.g., existence of minima for optimal control problems lacking convexity, reachability results for constrained systems, trajectory generation for nonlinear hybrid systems, real-time system guiding in uncertain environments). Research on overcoming these and many other new challenges has recently been re-energized by the funding of many efforts targeted towards the design of *autonomous systems*. Some of these efforts include: Unmanned Aerial Vehicles (UAV), Autonomous Terrestrial Vehicles (DARPA Grand Challenge), Mars Explorer Rovers (MER), and Autonomous Underwater Vehicles, as illustrated in Figure 1.1.

The optimal control problem framework happens to be remarkably well-suited to model guidance questions for these types of systems. This is evident if we consider some of their salient characteristics. Mainly, an autonomous system must be highly flexible for operating under ever-changing conditions (i.e., be able to formulate the state trajectory constraint set), be able to overcome reasonable failures (i.e., be able to construct control-relevant dynamic and actuator constraint sets), possess adaptive problem solving capabilities (i.e., be able to modify and pose new cost functionals), and have on-board, real-time computational resources (i.e., be able to solve the optimal control problem in real-time).

Figure 1.1: Research efforts on autonomous systems

However, traditional methods for the solving of optimal control problems when applied to autonomous systems tend to be either intractable in real-time through computational complexity or extremely conservative. To illustrate this, consider the use of a traditional numerical method for generating optimal trajectories for an autonomous system that must traverse an urban environment (obstacles, without loss of generality, can be modeled in this setting as polytopes). From the onset, a difficulty arises in the modeling of free space or the trajectory feasible set. This is a direct result of using an optimization-based approach. In optimization-based approaches, one must model the feasible set through the *intersection* of constraints sets. However, in obstacle avoidance problems the opposite is true: obstacles lead naturally to the trajectory feasible set being described by a conjunction of disjunctions. There are two different approaches that have been studied in the literature for overcoming this issue: a) approximating the feasible set with a single convex set, Faiz et al. [2001] and b) recasting the conjunction of disjunctions in terms of an intersection of constraint sets, Nemhauser and Wolsey [1988]. As one might expect, depending on the number of obstacles and their position in space, the first approach, if at all successful, can be very conservative in approximating the feasible set (e.g., because the feasible set is, generally, highly non-convex). In the second approach, there are two main ways of recasting the original problem. Namely, one can model polytopes by ellipses or other basic closed semi-algebraic sets (i.e., sets formed by intersecting a finite number of polynomial inequalities), or one can use the operations research approach of modeling disjunctive sets by introducing boolean variables. In both situations, one must deal with a large number of constraints present in the optimal control problem (nonlinear and/or mixed-integer), already a dire situation. But in addition, in the former case one trades modeling ease

for conservative results (i.e., over-approximation leads to the significant reduction of the trajectory feasible set, artificially obstructing, in some instances, free sections of space connecting the initial and final path conditions), while in the latter, the optimal control problem after transcription would become a nonlinear mixed-integer programming problem for which tools are scarce and, in any event, would be extremely computationally costly for real-time applications.

Fortunately, partial results for overcoming some of these issues can be found in the literature. Most notably, in the field of *motion planning* there has been a detailed study of methods for characterizing the geometry of the space which complements the one delineated by obstacles (free space), Latombe [1991]. In general, these methods decompose free space into simpler manageable regions whose structure can be exploited to determine sections of space that connect the initial and final configurations of the system and, therefore, are collectively known as *cell decomposition* methods. The most general method in this collection is *cylindrical algebraic decomposition*, Collins [1975], Canny [1988]. However, this method requires the use of computational algebraic geometry tools (e.g., Gröbner bases, multivariate polynomial resultants), Cox et al. [2005], which are very computationally expensive. In practical applications, one usually resorts to some type of compromise between generality and computational tractability by using *approximate cell decomposition* methods.

In like manner, the field of *control and dynamical systems* has infused the literature with methods targeted to obtaining, in real-time, control policies for guiding dynamical systems with actuator constraints (trajectory constraints usually neglected or simplified). In particular, methods restricted to the study of *differentially flat* systems, Fliess et al. [1995], have been able to reduce the computational burden often associated with direct methods. This has been possible because for differentially flat systems there exists a set of *flat outputs* (equal in number to the control inputs) such that all states and inputs can be determined from these outputs without integration. Consequently, one can rewrite the optimal control problem in terms of the flat outputs and then find minimizers in the *flat-output space*. Since flat outputs implicitly contain all the information about the dynamics of the system, by introducing this transformation, no explicit dynamic constraints remain in the transformed optimal control problem (i.e., removal of dynamic constraints), van Nieuwstadt [1996], Rathinam [1997], Milam [2003]. Particularly in Milam [2003], the real-time tractability of this approach was shown by successfully guiding a ducted fan through various useful scenarios. The reader is referred to his work for more information on these and other results, including numerical comparisons of this method against other more traditional methodologies such as direct collocation, shooting, adjoints, and differential inclusions. Finally, we should comment on the severeness of this restriction. It turns out that at worst, differentially flat systems include controllable linear systems

and nonlinear systems which are feedback linearizable either by static or dynamic feedback, Fliess et al. [1995]. In addition, other specific systems such as aircraft in forward flight and some classes of vertical take-off and landing (VTOL) aircraft have been determined to be differentially flat. In the end, even if the system is not differentially flat, many times one is able to approximate it by control-relevant models that are differentially flat.

1.2 Approach

In this thesis, the focus will be on the computation of minimizers for optimal control problems that model the effort required for nonlinear dynamical systems to evolve across state configurations while satisfying trajectory and actuator constraints. For the most part, the structure of these optimal control problems will be such that our only recourse in determining their minimizers will be by numerical means. Consequently, we will implement for this task a direct method (i.e., transcribe the original optimal control problem to a nonlinear programming problem). The main distinction in this implementation from all others previously published resides in the choice of basis functions used to describe each member of the restricted search space and in the exploitation of the properties of the resulting curves for modeling feasible constraint regions, leading to significant improvement of the computational burden required for obtaining minimizers for optimal control problems.

Mathematically, we will describe optimal control problems in the following manner:

$$\min_{\mathbf{x}(t),\mathbf{u}(t)} \mathcal{F}_0(\mathbf{x}(t_0),\mathbf{u}(t_0)) + \int_{t_0}^{t_f} \mathcal{F}_t(\mathbf{x}(t),\mathbf{u}(t))\ dt + \mathcal{F}_f(\mathbf{x}(t_f),\mathbf{u}(t_f)) \tag{1.2.1}$$

subject to

$$\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t),\mathbf{u}(t)), \qquad t \in [t_0, t_f] \tag{1.2.2}$$

$$\begin{aligned}
\boldsymbol{\ell}_0 \leq\ & \mathbf{A}_0\mathbf{x}(t_0) + \mathbf{B}_0\mathbf{u}(t_0) && \leq \mathbf{u}_0, \\
\boldsymbol{\ell}_t \leq\ & \mathbf{A}_t\mathbf{x}(t) + \mathbf{B}_t\mathbf{u}(t) && \leq \mathbf{u}_t, \qquad t \in [t_0, t_f] \\
\boldsymbol{\ell}_f \leq\ & \mathbf{A}_f\mathbf{x}(t_f) + \mathbf{B}_f\mathbf{u}(t_f) && \leq \mathbf{u}_f, \qquad \text{and}
\end{aligned} \tag{1.2.3}$$

$$\begin{aligned}
\mathbf{L}_0 \leq\ & \mathbf{c}_0(\mathbf{x}(t_0),\mathbf{u}(t_0)) && \leq \mathbf{U}_0, \\
\mathbf{L}_t \leq\ & \mathbf{c}_t(\mathbf{x}(t),\mathbf{u}(t)) && \leq \mathbf{U}_t, \qquad t \in [t_0, t_f] \\
\mathbf{L}_f \leq\ & \mathbf{c}_f(\mathbf{x}(t_f),\mathbf{u}(t_f)) && \leq \mathbf{U}_f,
\end{aligned} \tag{1.2.4}$$

where the state and input evolutions are described by the mappings $\mathbf{x} : [t_0, t_f] \to \mathcal{X} \subset \mathbb{R}^n$ and

$\mathbf{u} : [t_0, t_f] \to \mathcal{U} \subset \mathbb{R}^m$. Moreover, the cost functional (1.2.1), dynamic constraints (1.2.2), and trajectory and actuator constraints (1.2.3)-(1.2.4) will be assumed to be sufficiently smooth. In addition, the cost functional is expressed as a sum of three terms (i.e. initial, trajectory, and final). Each function $\mathcal{F}_\ell$, $\ell \in \{0, t, f\}$ in the cost functional is a scalar-valued function $\mathcal{F}_\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$. The linear and nonlinear constraints are also divided into three terms (i.e., initial, trajectory, and final) and they are vector-valued functions $\mathcal{L}_\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{N_\ell^l}$, $c_\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{N_\ell^n}$, $\ell \in \{0, t, f\}$. Each of these sets of constraints is allowed to be set to equality, one-sided bounded, two-sided bounded, or unbounded by manipulating the values of the lower and upper bounds. If in specific applications we are able to exploit available theoretical results, we will impose further structure on some or all of the components making up the optimal control problem (e.g., convexity). The explicit partition of linear and nonlinear constraints shown here is simply to emphasize that we are able to exploit the detailed structure of the posed optimal control problem, including secondary structures such as sparseness of matrices. Moreover, this optimal control problem formulation also includes time varying functions. In that case, one simply augments the states vector with time as a state with zero dynamics, Pontryagin et al. [1962].

Obtaining the true minimizer to the optimal control problem (1.2.1)–(1.2.4) (i.e., computing the pair $\mathbf{x}(t)$, $\mathbf{u}(t)$) is, in general, an intractable problem. This is because, in order to obtain the true state and input evolutions, one must seek for minimizers in the space of all k times continuously differentiable functions, which is an infinite-dimensional space. As a consequence, we are forced to search for approximations to the minimizers in finite-dimensional vector spaces. The most versatile and useful choice among these is the space of *all piecewise polynomial functions with a prescribed number of polynomial pieces, order, and smoothness.* However, this restriction by itself does not ensure that the problem is tractable since we must still satisfy constraints at an infinite number of points (minimizers are made up of an infinite number of points, and the constraints are functions of them). So we proceed with a balanced discretization of the optimal control problem. The result of the previous search space restriction and subsequent discretization is a transcription of the optimal control problem (1.2.1)–(1.2.4) to a nonlinear programming problem whose mathematical structure may be expressed in general as follows:

$$\begin{aligned} &\min_{\mathbf{y}} f(\mathbf{y}) \\ \text{subject to} \quad & \\ &\mathbf{L} \leq \begin{bmatrix} \mathbf{y} \\ \mathbf{A}\mathbf{y} \\ \mathbf{c}(\mathbf{y}) \end{bmatrix} \leq \mathbf{U}, \end{aligned} \tag{1.2.5}$$

where all of the functions are, roughly speaking, dependent on the coefficients of the polynomials only. In addition to the nonlinear programming problem having a simpler structure than the original optimal control problem, it also has the advantage of having been studied in detail for the past few decades, giving rise to a large number of efficient solvers, Gill et al. [2005], Byrd et al. [2006]. In particular, methods using the *sequential quadratic programming* (SQP) paradigm have been very successful at solving large-scale nonlinear constrained optimization problems of this type, Gill et al. [2005]. Additionally, in recent years, newer methods based on *sequential linear-quadratic programming* (SLQP) have shown similar solving capabilities of large-scale nonlinear programming problems at competing efficiency, Byrd et al. [2006]. However, even in this form, the problem might remain real-time intractable if the number of decision variables and constraints are too many to solve the problem fast enough. Consequently, it is useful, in general, to reduce the number of constraints in the optimal control problem as far as possible so as not to overdiscretize the problem or to seek for piecewise polynomial functions with a large number of coefficients (e.g., high degree). Finally, in lieu of the structure of the nonlinear programming problems that we will be generating, it is important to realize that, in general, the methods at our disposal can only guarantee to locate local minimizers. This is, in general, not a limitation for engineering applications where feasibility usually is sufficient. In fact, even when considering applications to autonomous systems where sensor data is only available for the immediate surroundings of the autonomous unit, one is pressed to justify the use of a computationally expensive global optimization solver, especially since one cannot guarantee that the global minimizers of the local problems will converge to the global minimizer of the whole mission.

As mentioned before, we will restrict our search for minimizers to the vector space of *all piecewise polynomial functions with a prescribed set of polynomial pieces of a given order and required smoothness*. We will identify this space by $\mathcal{P}_{\mathbf{b},o,s}$, where $\mathbf{b} = \{b_0, \ldots, b_{N_p}\}$ are the $(N_p + 1)$ *breakpoints* (i.e., the sites at which the endpoints of the N_p polynomial pieces reside), o is the order of the polynomial pieces, and s dictates that the resulting curve be at least sth continuously differentiable. In particular, we will express each member of the vector space $\mathcal{P}_{\mathbf{b},o,s}$ as a linear combination of *non-uniform rational B-spline* (NURBS) basis functions. That is,

$$c(t) = \sum_{j=0}^{N_c-1} \mathcal{R}_{j,d}^{(0)}(t, w_0, \ldots, w_{N_c-1})\ \mathbf{p}_j, \tag{1.2.6}$$

where $\mathcal{R}_{j,d}^{(0)}$ is the jth NURBS function of degree, d, dependent on time, $t \in \mathbb{R}$, and weights, $w_j \in \mathbb{R}$, and $\mathbf{p}_j \in \mathbb{R}^d$ is the jth *control point* (coefficient of the linear combination).

Curves described in this manner and their extensions to surfaces and solids have been studied extensively in the field of *computer graphics*, where they have become an industry standard for the representation and design of geometry, de Boor [1978], Piegle and Tiller [1997]. This adoption has resulted in a proliferation of efficient algorithms for computing every aspect of their makeup. In this thesis, we will harness their structure to improve upon previous trajectory generation methods. Even though it is not readily apparent, there are two important structures implicitly combined in equation (1.2.6) that we will exploit (see Figure 1.2). One of them is provided by the NURBS basis functions themselves (which after discretization depend only on the *weights*) with local-support, non-negativity, and partition-of-unity properties. The other comes from a union of overlapping polytopes whose vertices are subsets of the coefficients of the linear combination (*control points*) with endpoint-interpolation and strong-convex-hull properties. Finally, we remark that this choice of basis was a very conscious one. From the onset, we realized that by collapsing NURBS basis functions to B-spline basis functions by manipulating the weights, we were able to recover, with little effort, the results obtained in Milam [2003], including the real-time capabilities of the method. This afforded us with a great vantage point from where to start building new results.

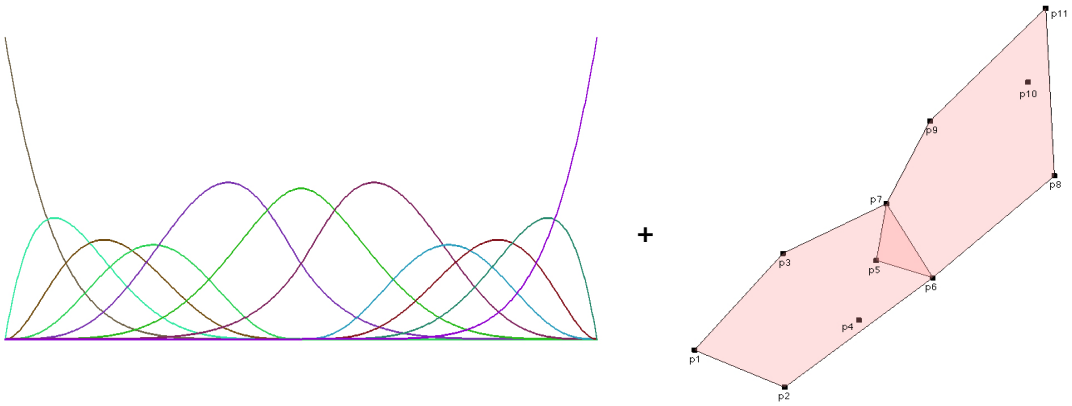

Figure 1.2: NURBS basis functions and union of overlapping polytopes

In summary, in order to determine minimizers for optimal control problems of the form (1.2.1)–(1.2.4), we begin by restricting the state and input evolutions to lie in independent vector spaces. That is, each state and input evolution is allowed to be a member of its own vector space, $\mathcal{P}_{\mathbf{b},o,s}$. In the current context, we may search for $n+m$ curves, all in distinct vector spaces. After defining the search spaces for all the curves and rewriting the optimal control problem accordingly, we proceed with a balanced discretization along the t parameter. That is, we selectively choose sites at which all the constraints will be guaranteed to be satisfied. The decision variables of the resulting

nonlinear programming problem are then the weights and control points pertaining to each of the state and input curves. Unfortunately, solving optimal control problems in this manner shows little advantage over previously studied methods at dealing with guidance questions arising in the study of autonomous systems (e.g., autonomous system traversing an urban environment) because of the same reasons discussed above (namely, the treatment of intersections of disjunctive constraint sets and the presence of a large number of constraints and decision variables).

However, by exploiting the structure of both the NURBS basis functions and the union of overlapping polytopes, we are able in some cases (e.g., the path of the system lies in $\mathbb{R}^d$, $d = \{2, 3\}$) to separately treat the guidance and obstacle-avoidance problems, making the original problem tractable. Recall that NURBS basis functions, after discretization, depend only on the weights and that the union of overlapping polytopes are completely defined by the control points (coefficients of the linear combination). Parametric curves resulting from the combination of these two structures possess many useful properties. Among them we have that each polynomial piece is guaranteed to be bounded by a corresponding polytope. This idea extends to the whole piecewise polynomial function in which case the parametric curve is guaranteed to remain inside the union of a set of overlapping polytopes. Therefore, if the control points defining such polytopes happen to be fixed in space and we manipulate the NURBS basis functions by varying the weights, then we are able to obtain an infinite number of curves, all lying inside the space delineated by the polytopes, regardless of the values of the weights (see Figure 1.3). As a consequence, if the polytopes delineate a section of space that is feasible with respect to some set of trajectory constraints, then the resulting parametric curve will also be feasible with respect to the same set of trajectory constraints.

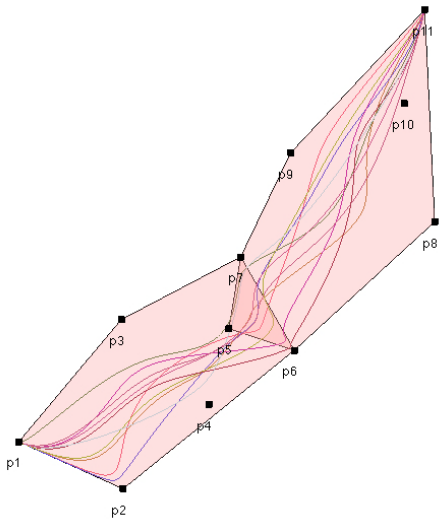

Figure 1.3: NURBS curves generated by setting weights randomly.

It is now evident how to proceed to separately treat the guidance and obstacle-avoidance subproblems. We first design a section of space (in general, non-convex) that is feasible with respect to trajectory constraints and that contains the initial and final path conditions by judiciously placing the control points (e.g., to form a union of overlapping polytopes avoiding all obstacles). Then, we generate paths which minimize the original cost functional and satisfy the dynamic and actuator constraints (i.e., omit the trajectory constraints), using the active weights and control points as decision variables (some of the control points having been fixed in the previous phase). The key observation is that paths generated by solving the latter problem will be guaranteed to be contained inside the feasible region generated in the former phase (i.e., be feasible with respect to trajectory constraints) without the need of explicitly writing the trajectory constraints into the optimal control problem and independent of the discretization. Since in obstacle-avoidance problems the bulk of the trajectory constraints are used to describe free space, taking advantage of this ability will improve the computational complexity of the problem. Furthermore, if the dynamical system happens to be differentially flat, then we are also able to exploit this fact and remove the dynamic constraints from the optimal control problem, improving further the computational complexity of the problem.

We remark that the above description represents only one of the ways in which we can exploit the properties of NURBS curves. In general, we are able to remove any set of constraints in $\mathbb{R}^d$, $d = \{2, 3\}$ from the optimal control problem so long as these constraints are amenable to being mutually satisfied by a union of overlapping polytopes and so long as there exists a natural parametric relationship among the variables of interest. In addition, it is also possible to remove more than one set of constraints at a time, requiring only that they be consistent.

One of the challenges in the before-mentioned procedure is the design of the feasible region with respect to trajectory constraints by control point manipulation. This is a relative easy task (but long and tedious) if one is able to use a *computer aided design* (CAD) application because, in this setting, one is able to draw all the obstacles and interactively place the control points until the desired result is achieved. This is especially useful for those applications where one can afford off-line computations. In general, though, we are more interested in designing such feasible regions using some type of automatic procedure where the input is a set of trajectory constraints for a parametric NURBS curve, and the output is a union of overlapping polytopes which satisfies the given set of constraints. To determine such a procedure without any further restrictions is complicated by the following issues: a) a lack of uniqueness of a feasible region unless it is a singleton (in general there exist many feasible regions connecting the initial and final path conditions), b) must guarantee that the smallest convex sets containing the relevant control points per polynomial piece are feasible with

respect to the constraints (the smallest convex set of a set of points is, by definition, the convex hull of them, and there is no analytical way of expressing this structure ahead of time), c) the feasible set is described by a conjunction of disjunctions, and d) it is not clear what objective function to use to drive a desired structure. Consequently, we are forced to determine a pragmatic set of conditions under which it is possible to construct feasible regions reliably. We address theoretically such a problem by requiring that a section of space, feasible with respect to trajectory constraints, be given in the form of an ordered union of pairwise adjacently overlapping non-empty convex compact sets (see Figure 1.4). Practically, the individual sets making up the union are compact convex semi-algebraic sets with nonempty interiors (the simplest sets being polytopes). The construction of such regions can be obtained using a cell decomposition method or constructed online using sensor information followed by a pairwise adjacently overlapping procedure.

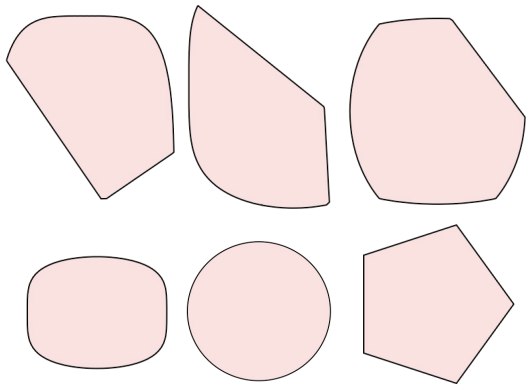

Figure 1.4: Examples of non-empty convex compact semi-algebraic sets

1.3 Thesis Contributions

The contributions of this thesis include the introduction of a new method for numerically computing minimizers to optimal control problems that model the effort required for nonlinear dynamical systems to evolve across state configurations while satisfying trajectory and actuator constraints. In particular, we implement a direct method (i.e. transcription of the original optimal control problem to a nonlinear programming problem). One of the main distinctions in this implementation from all others previously published resides in the choice of basis functions used to describe each member of the restricted search space. Specifically, the restricted search space consist of the space of all piecewise polynomial functions with a prescribed number of polynomial pieces, order, and smoothness. This space happens to be finite dimensional and, consequently, each member in the space is expressed in terms of some set of basis functions. We choose NURBS basis functions for

this purpose and show how to judiciously exploit the properties of the resulting NURBS curve to improve the computational effort often associated with solving optimal control problems for constrained dynamical systems.

More specifically, we show in detail how to exploit the structure of both the NURBS basis functions and the associated union of overlapping polytopes resulting from the placement in d-space ($d \in \{2, 3\}$) of the coefficients of the linear combination of the NURBS curve. One of the main realizations uncovered through the careful study of these structures is the fact that one can use the strong convex hull property of NURBS curves to separately treat the guidance and obstacle-avoidance problems, making the original optimal control problem tractable. This separation is accomplished by manipulating both the weights and control points; one set of parameters used at a time to solve one of the problems. That is, we show how one can first design a connected section of space (in general, non-convex) by judiciously placing the control points in such a way that the union of the associated polytopes is feasible with respect to trajectory constraints and contains the initial and final path conditions. Then, generate paths which minimize the original cost functional and satisfy the dynamic and actuator constraints (omitting the trajectory constraints), using the active weights as decision variables. The key observation is that paths generated by solving the later problem are guaranteed to be contained inside the feasible region generated in the former phase, without the need to explicitly writing the trajectory constraints into the optimal control problem, and independent of the discretization.

In addition, we show how one can construct systematically a feasible region that corresponds to a NURBS parameterization starting from an ordered union of pairwise adjacently overlapping non-empty compact convex sets. Specifically, we show how to setup a nonlinear programming problem to solve for the feasible region in terms of an ordered union of pairwise adjacently overlapping polytopes with nonempty interiors by maximizing the sum of their volumes and starting from a feasible region described by an ordered union of pairwise adjacently overlapping nonempty convex compact semi-algebraic sets. In addition, we show in simulation how this strategy can be implemented practically for an autonomous system traversing an urban environment.

Finally, this work culminated in the filing of patent US20070179685 on behalf of Northrop Grumman for the Space Technology sector and in the development of the NURBSbasedOTG software package. This C++ package contains the theoretical results of this thesis in the form of an object-oriented implementation optimized for real-time trajectory generation.

1.4 Thesis Context

It is important to place the topic of this thesis in context with respect to other ongoing research directions. In particular, there is an interest in developing algorithms for the control of distributed, intelligent, multi-agent systems. Some of the most salient research directions include gathering information through multi-sensor networks, resource-constrained scheduling of tasks to be completed by multiple heterogeneous/homogeneous agents, rerouting traffic dynamically in the National Airspace System, and formation flying of spacecraft/aircraft, Pan et al. [2006], Lynch [1996], Wall [1996], Tomlin [1998], Wooldridge [2002], Murray [2003].

Ultimately, these research directions impose high demands upon the individual agents, requiring from them higher performance and efficiency. Consequently, research on real-time feedback strategies (i.e. Receding Horizon Control) that achieve disturbance rejection and tracking under the uncertain conditions imposed by the new systems is essential. At the heart of modern feedback strategies resides a reconfigurable optimal control problem that needs to be solved in time to implement the appropriate forcing to guide the system. Reconfigurability, in this context, is defined as the ability to modify any of the components (i.e., cost functional, state and inputs constraints, and system dynamics) making up the open-loop optimal control problem. Figure 1.5 illustrates the flow of information from sensors to and through the various components making up an intelligent system being controlled through feedback.

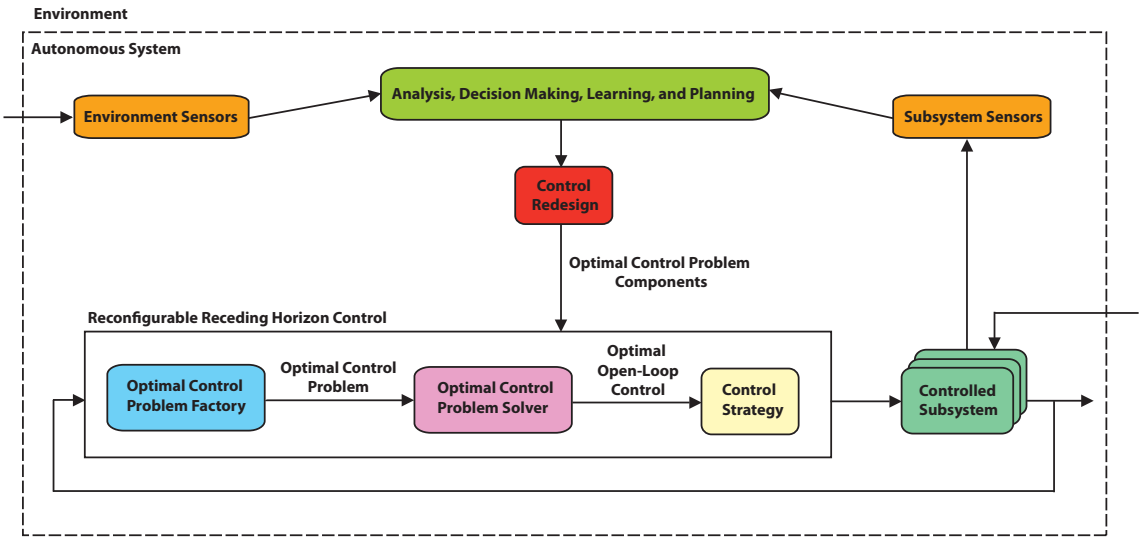

Figure 1.5: Flow of information from sensors to and through the various components making up an intelligent system being controlled through feedback

Mathematically, scheduling, routing, formation flying, and distributed networking are problems that are resolved through the computation of minimizers to either mixed-integer programming problems or convex optimization problems. However, the complexity of resolving these issues in addition to the determination of safe trajectories for a multi-agent system is computationally intractable. This realization has led researchers to pursue for resolution of these problems through the decoupling of decision-making issues and trajectory generation.

Figure 1.6, illustrates a procedure by which the new approach presented in this thesis can be applied to an unmanned aerial vehicle (UAV) for terrain avoidance or urban reconnaissance: a) a global solver (e.g., through convex optimization) is used to determine feasible corridors through trajectory space and to isolate an optimal corridor with respect to some decision-making objective, b) the corridor is inner approximated by the union of a set of convex polytopes whose vertices arise from a NURBS definition c) the inner approximation in b) allows for the removal of explicit trajectory constraints during trajectory generation, significantly reducing the effort required to solve the optimal control problem and d) a local optimal trajectory living in the feasible region with respect to trajectory constraints is generated which is also feasible with respect to dynamic and actuator constraints. The last step may fail to succeed due to various causes, among them, the infeasibility of the posed optimal control problem. In this event, apart from reviewing the optimal control problem itself for inconsistencies, one could re-iterate the above procedure starting from b) and inner approximate one of the other alternative routes leading to the final position.

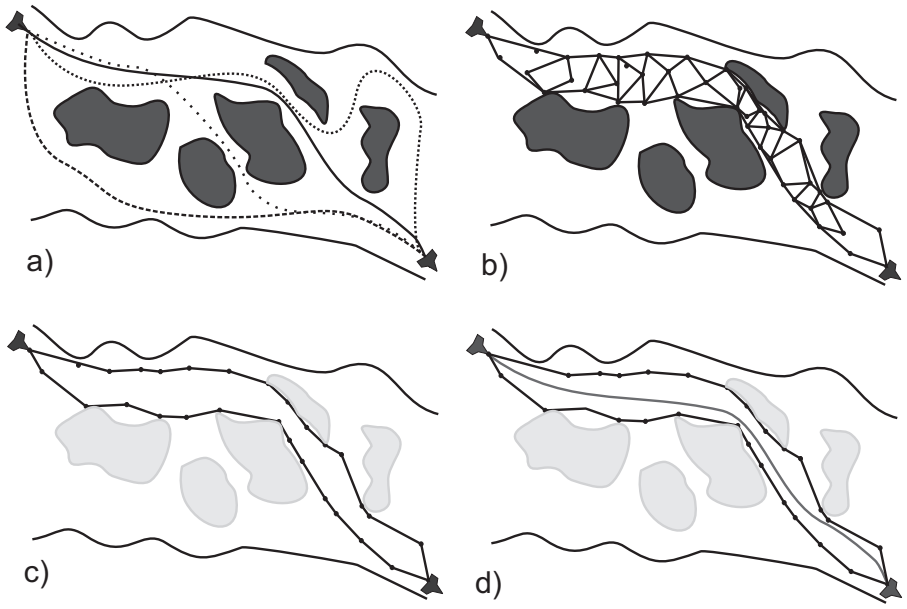

Figure 1.6: Illustration of how method can be applied to UAV with obstacle avoidance

1.5 Highlights of the Thesis

This thesis is arranged in the following manner: Chapter 2 contains a brief introduction to convex sets with emphasis on computations for polytopes. Chapter 3 describes piecewise polynomial functions expressed in terms of a linear combination of B-spline and NURBS basis functions. Moreover, their most salient properties and trajectory-generation relevant computations are exposed. Chapter 4 describes in detail the transcription of the optimal control problem to a nonlinear programming problem. Chapter 5 demonstrates the solution of optimal control problems where the decision variables are restricted to control points only. In particular, we treat optimal control problems containing explicit dynamic constraints. Chapter 6 is used to expand on some of the most prominent properties of NURBS curves for trajectory generation purposes. In Chapter 7, we use the full power of the methodology and apply it to differentially flat systems with obstacle avoidance. In addition, we present a method for inner-approximating a feasible region by an ordered union of pairwise adjacently overlapping polytopes. Finally, Chapter 8 summarizes the results and suggests extensions to the current body of work.

Chapter 2

Convex Sets

In this chapter, we present a brief introduction to convex sets. The material contained in this chapter has mostly been obtained from the following references: Boyd and Vandenberghe [2004], Barvinok [2002], and Webster [1994].

The inner product of two vectors $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^n$ is expressed by

$$\langle \mathbf{x}, \mathbf{y} \rangle = x_1 y_1 + \cdots + x_n y_n. \tag{2.0.1}$$

The norm of a vector $\mathbf{x} \in \mathbb{R}^n$ is defined by

$$\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}. \tag{2.0.2}$$

The unit ball in $\mathbb{R}^n$ is denoted by $\mathcal{B}_n$:

$$\mathcal{B}_n = \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\| \leq 1\}. \tag{2.0.3}$$

Let $\mathcal{A} \subset \mathbb{R}^n$. The distance function $d(\cdot, \mathcal{A}) : \mathbb{R}^n \to \mathbb{R}$ is defined by

$$d(\mathbf{x}, \mathcal{A}) = \inf\{\|\mathbf{x} - \mathbf{a}\| \mid \mathbf{a} \in \mathcal{A}\},\ \mathbf{x} \in \mathbb{R}^n. \tag{2.0.4}$$

Let $\lambda \in \mathbb{R}$. Then, put by definition

$$\lambda \mathcal{A} = \{\lambda \mathbf{a} \mid \mathbf{a} \in \mathcal{A}\}. \tag{2.0.5}$$

For two sets $\mathcal{A}$ and $\mathcal{B}$ in $\mathbb{R}^n$ their sum is defined by

$$\mathcal{A} + \mathcal{B} = \{\mathbf{a} + \mathbf{b} \mid \mathbf{a} \in \mathcal{A}, \mathbf{b} \in \mathcal{B}\}. \tag{2.0.6}$$

The *closure* cl $\mathcal{A}$ and *interior* int $\mathcal{A}$ of $\mathcal{A}$ are defined by

$$\text{cl}(\mathcal{A}) = \bigcap_{\varepsilon>0}(\mathcal{A} + \varepsilon\mathcal{B}_n), \qquad \text{and} \tag{2.0.7}$$

$$\text{int}(\mathcal{A}) = \{\mathbf{a} \in \mathcal{A} \mid \exists\varepsilon > 0, \mathbf{a} + \varepsilon\mathcal{B}_n \subset \mathcal{A}\}. \tag{2.0.8}$$

The *boundary* of $\mathcal{A}$ is defined by

$$\text{bd } \mathcal{A} = \text{cl } \mathcal{A} \setminus \text{int } \mathcal{A}. \tag{2.0.9}$$

The support function of a set $\mathcal{A} \subset \mathbb{R}^n$ $(\mathcal{A} \neq \emptyset)$ is defined as

$$h(\mathbf{u}) = \sup\{\mathbf{u}^T\mathbf{a} \mid \mathbf{a} \in \mathcal{A}\}. \tag{2.0.10}$$

2.1 Sets and Hulls

Definition 2.1 (Affine set)**.** A set $\mathcal{S} \subset \mathbb{R}^n$ is *affine* if the line through any two distinct points in $\mathcal{S}$ lies in $\mathcal{S}$, i.e., if for any $\mathbf{s}_1, \mathbf{s}_2 \in \mathcal{S}$ and $\theta \in \mathbb{R}$, we have $\theta\mathbf{s}_1 + (1-\theta)\mathbf{s}_2 \in \mathcal{S}$.

Definition 2.2 (Affine Hull)**.** The set of all affine combinations of points in some set $\mathcal{S} \subseteq \mathbb{R}^n$ is called the *affine hull* of $\mathcal{S}$, and it is denoted by *aff* $\mathcal{S}$; that is,

$$\text{aff } \mathcal{S} = \{\theta_1\mathbf{s}_1 + \cdots + \theta_k\mathbf{s}_k \mid \theta_1 + \cdots + \theta_k = 1, \mathbf{s}_i \in \mathcal{S}, \theta_i \in \mathbb{R}, i = 1, \ldots, k\}. \tag{2.1.1}$$

The *affine hull* is the smallest affine set that contains $\mathcal{S}$, in the following sense: if $\mathcal{A}$ is any affine set with $\mathcal{S} \subseteq \mathcal{A}$, then aff $\mathcal{S} \subseteq \mathcal{A}$.

Definition 2.3 (Affinely Independent)**.** $k+1$ points $\mathbf{s}_0, \ldots, \mathbf{s}_k \in \mathbb{R}^n$ are *affinely independent* if the points $\mathbf{s}_1 - \mathbf{s}_0, \ldots, \mathbf{s}_k - \mathbf{s}_0$ are linearly independent.

Definition 2.4 (Convex set)**.** A set $\mathcal{S}$ is *convex* if the line segment between any two points in $\mathcal{S}$ lies in $\mathcal{S}$, i.e., if for any $\mathbf{s}_1, \mathbf{s}_2 \in \mathcal{S}$ and any θ with $0 \leq \theta \leq 1$, we have $\theta\mathbf{s}_1 + (1-\theta)\mathbf{s}_2 \in \mathcal{S}$.

In particular, the empty set, any single point, and the whole space $\mathbb{R}^n$ are convex subsets of $\mathbb{R}^n$. In addition, the intersection of an arbitrary family of convex sets in $\mathbb{R}^n$ is convex.

Definition 2.5 (Convex Hull)**.** The *convex hull* of a set $\mathcal{S}$, denoted *conv* $\mathcal{S}$, is the set of all convex combinations of points in $\mathcal{S}$:

$$\text{conv } \mathcal{S} = \{\theta_1\mathbf{s}_1 + \cdots + \theta_k\mathbf{s}_k \mid \theta_1 + \cdots + \theta_k = 1, \theta_i \geq 0, \mathbf{s}_i \in \mathcal{S}, \theta_i \in \mathbb{R}, i = 1, \ldots, k\}. \tag{2.1.2}$$

Theorem 2.1. Let $\mathcal{V}$ be a vector space and let $\mathcal{S} \subset \mathcal{V}$ be a set. Then, the convex hull of $\mathcal{S}$ is a convex set, and any convex set containing $\mathcal{S}$ also contains *conv* $\mathcal{S}$. In other words, *conv* $\mathcal{S}$ is the smallest convex set containing $\mathcal{S}$.

In addition, if $\mathcal{A} \subset \mathcal{B}$, then *conv* $\mathcal{A} \subset$ *conv* $\mathcal{B}$, and *conv* $\mathcal{A} \cup$ *conv* $\mathcal{B} \subset$ *conv* $(\mathcal{A} \cup \mathcal{B})$.

Definition 2.6. Let $\mathcal{S} \subset \mathbb{R}^d$ be a set. A point $\mathbf{s} \in \mathcal{S}$ is called an *interior* point of $\mathcal{S}$ if there exits an $\varepsilon > 0$ such that the open ball $\mathcal{B}(\mathbf{s}, \varepsilon) = \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{s}\| < \varepsilon\}$, centered at $\mathbf{s}$ and of radius ε, is contained in $\mathcal{S}$; that is, $\mathcal{B}(\mathbf{s}, \varepsilon) \subset \mathcal{S}$. The set of all interior points of $\mathcal{S}$ is called the *interior* of $\mathcal{S}$ and denoted *int* $\mathcal{S}$. The set of all non-interior points of $\mathcal{S}$ is called the *boundary* of $\mathcal{S}$ and denoted $\partial\mathcal{S}$.

Lemma 2.2. Let $\mathcal{S} \subset \mathbb{R}^d$ be a convex set, and let $\mathbf{s}_0 \in$ *int* $\mathcal{S}$ be an interior point of $\mathcal{S}$. Then, for any point $\mathbf{s}_1 \in \mathcal{S}$ and any $0 \leq \alpha < 1$, the point $\mathbf{s}_\alpha = (1 - \alpha)\, \mathbf{s}_0 + \alpha\, \mathbf{s}_1$ is an interior point of $\mathcal{S}$.

Corollary 2.3. Let $\mathcal{S} \subset \mathbb{R}^d$ be a convex set. Then, the *int* $\mathcal{S}$ is a convex set.

Definition 2.7 (Halfspace)**.** A closed halfspace is a set of the form

$$\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{a}^T\mathbf{x} \leq b\}, \tag{2.1.3}$$

where $\mathbf{a} \in \mathbb{R}^n$, $\mathbf{a} \neq \mathbf{0}$, and $b \in \mathbb{R}$.

Halfspaces are convex.

Definition 2.8 (Hyperplane)**.** A *hyperplane* is a set of the form

$$\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{a}^T\mathbf{x} = b\}, \tag{2.1.4}$$

where $\mathbf{a} \in \mathbb{R}^n$, $\mathbf{a} \neq \mathbf{0}$, and $b \in \mathbb{R}$.

A hyperplane divides $\mathbb{R}^n$ into two *halfspaces*. Hyperplanes are affine and convex. In addition, for both halfspaces and hyperplanes, $\mathbf{a}$ is the normal vector.

Theorem 2.4. Let $\mathcal{S} \subset \mathbb{R}^d$ be a convex set with a non-empty interior, and let $\mathbf{s} \in \partial\mathcal{S}$ be a point. Then, there exists an affine hyperplane $\mathcal{H}$, called a support hyperplane at $\mathbf{s}$, such that $\mathbf{s} \in \mathcal{H}$ and $\mathcal{H}$ isolates $\mathcal{S}$.

Theorem 2.5. Let $\mathcal{S} \subset \mathbb{R}^d$ be a non-empty convex set, and let $\mathbf{s} \notin \partial\mathcal{S}$ be a point. Then, there is an affine hyperplane $\mathcal{H} \subset \mathbb{R}^d$ such that $\mathbf{s} \in \mathcal{H}$, and $\mathcal{H}$ isolates $\mathcal{S}$.

Definition 2.9 (Conic set)**.** A set $\mathcal{S}$ is called a *cone*, or *nonnegative homogeneous*, if for every $\mathbf{s} \in \mathcal{S}$ and $\theta \geq 0$ we have $\theta\mathbf{s} \in \mathcal{S}$. A set $\mathcal{S}$ is a *convex cone* if it is convex and a cone, which means that for

any $\mathbf{s}_1, \mathbf{s}_2 \in \mathcal{S}$, and $\theta_1, \theta_2 \geq 0$, we have

$$\theta_1 \mathbf{s}_1 + \theta_2 \mathbf{s}_2 \in \mathcal{S}. \tag{2.1.5}$$

Definition 2.10 (Conic Hull)**.** The *conic hull* of a set $\mathcal{S}$ is the set of all conic combinations of points in $\mathcal{S}$, i.e.,

$$\{\theta_1 \mathbf{s}_1 + \cdots + \theta_k \mathbf{s}_k \mid \mathbf{s}_i \in \mathcal{S}, \theta_i \geq 0, i = 1, \ldots, k\}, \tag{2.1.6}$$

which is also the smallest convex cone that contains $\mathcal{S}$.

A *ray* which has the form $\{\mathbf{s}_0 + \theta \mathbf{v} \mid \theta \geq 0\}$, where $\mathbf{v} \neq \mathbf{0}$ is convex but not affine. It is a convex cone if its base $\mathbf{s}_0$ is $\mathbf{0}$.

2.2 Polyhedra

Definition 2.11 (Polyhedron)**.** A *polyhedron* is defined as the solution set of a finite number of linear equalities and inequalities:

$$\mathcal{P} = \{\mathbf{x} \mid \mathbf{a}_j^T \mathbf{x} \leq b_j, j = 1, \ldots, m, \mathbf{c}_k^T \mathbf{x} = d_k, k = 1, \ldots, p\}, \tag{2.2.1}$$

where $\mathbf{a}_j \in \mathbb{R}^n$ and $b_j \in \mathbb{R}$ for $j = 1, \ldots, m$ and $\mathbf{c}_k \in \mathbb{R}^n$ and $d_k \in \mathbb{R}$ for $k = 1, \ldots, p$.

Alternatively, in compact form

$$\mathcal{P} = \{\mathbf{x} \mid \mathbf{A}\ \mathbf{x} \leq \mathbf{b}, \mathbf{C}\ \mathbf{x} = \mathbf{d}\}, \tag{2.2.2}$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{C} \in \mathbb{R}^{p \times n}$, $\mathbf{b} \in \mathbb{R}^m$, and $\mathbf{d} \in \mathbb{R}^p$. A polyhedron is thus the intersection of a finite number of halfspaces and hyperplanes.

In particular, polyhedra are convex sets.

Polytopes

A bounded polyhedron is called *polytope*.

Definition 2.12 ($\mathcal{V}$-polytope)**.** A *$\mathcal{V}$-polytope* is the convex hull of a finite set $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ of points in $\mathbb{R}^d$:

$$\mathcal{P} = \operatorname{conv}(\mathcal{V}) := \left\{ \sum_{i=1}^{n} \lambda_i \mathbf{v}_i \in \mathbb{R}^d \mid \lambda_i \geq 0,\ \sum_{i=1}^{n} \lambda_i = 1 \right\}. \tag{2.2.3}$$

Definition 2.13 ($\mathcal{H}$-polytope)**.** An *$\mathcal{H}$-polytope* is a bounded solution set of a finite system of linear inequalities:

$$\mathcal{P} = \mathcal{P}(\mathbf{A}, \mathbf{b}) := \left\{ \mathbf{x} \in \mathbb{R}^d \mid \mathbf{a}_j^T \mathbf{x} \leq b_j,\ j = 1, \ldots, m \right\}, \tag{2.2.4}$$

where $\mathbf{A} \in \mathbb{R}^{m \times d}$ is a real matrix with rows $\mathbf{a}_i^T$, and $\mathbf{b} \in \mathbb{R}^m$ is a real vector with entries b_i. Here, boundedness means that there is a constant N such that $\|\mathbf{x}\| \leq N$ holds of all $\mathbf{x} \in \mathcal{P}$.

Definition 2.14. Let $\mathcal{P}$ and $\mathcal{Q}$ be polytopes in $\mathbb{R}^n$, and let $\lambda \geq 0$ be a real number.

a) The *Minkowski sum* of $\mathcal{P}$ and $\mathcal{Q}$, denoted $\mathcal{P} + \mathcal{Q}$, is

$$\mathcal{P} + \mathcal{Q} = \{\mathbf{p} + \mathbf{q} \mid \mathbf{p} \in \mathcal{P} \text{ and } \mathbf{q} \in \mathcal{Q}\}. \tag{2.2.5}$$

b) The polytope $\lambda\mathcal{P}$ is defined by

$$\lambda\mathcal{P} = \{\lambda\mathbf{p} \mid \mathbf{p} \in \mathcal{P}\}, \tag{2.2.6}$$

where $\lambda\mathbf{p}$ is the usual scalar multiplication on $\mathbb{R}^n$.

Theorem 2.6. Let $\mathcal{A}, \mathcal{B}$ be polytopes in $\mathbb{R}^n$, and let $\alpha \in \mathbb{R}$. Then, $\mathcal{A} + \mathcal{B}$ and $\alpha\mathcal{A}$ are polytopes.

Corollary 2.7. Let $\mathcal{A}_1, \ldots, \mathcal{A}_k$ be polytopes in $\mathbb{R}^n$ and let $\alpha_1, \ldots, \alpha_k \in \mathbb{R}$. Then $\alpha_1\mathcal{A}_1 + \cdots + \alpha_k\mathcal{A}_k$ is a polytope.

Definition 2.15 (Simplex)**.** A *simplex* is a polytope that is the convex hull of an affinely independent set of points. That is,

$$\mathcal{S} = \operatorname{conv}(\mathbf{v}_0, \ldots, \mathbf{v}_k) = \{\theta_0\mathbf{v}_0 + \cdots + \theta_k\mathbf{v}_k \mid \boldsymbol{\theta} \geq \mathbf{0}, \mathbf{1}^T\boldsymbol{\theta} = 1\}, \tag{2.2.7}$$

where $\mathbf{v}_0, \ldots, \mathbf{v}_k$ are affinely independent.

Lemma 2.8. Let $\mathcal{P} = \operatorname{conv}(\mathbf{v}_1, \ldots, \mathbf{v}_m) \subset \mathbb{R}^d$ be a polytope and let $\mathcal{F} \subset \mathcal{P}$ be a face. Then, $\mathcal{F} = \operatorname{conv}(\{\mathbf{v}_i \mid \mathbf{v}_i \in \mathcal{F}\})$. In particular, a face of a polytope is a polytope, and the number of faces of a polytope is finite.

Definition 2.16. A 0-dimensional face of a polytope is called a *vertex*. A 1-dimensional face of a polytope is called an *edge*. A $(d-1)$ dimensional face of a d-dimensional polytope is called a *facet*. A $(d-2)$-dimensional face of a d-dimensional polytope is called a *ridge*.

Theorem 2.9. Let $\mathcal{P} \subset \mathbb{R}^d$ be a polytope

$$\mathcal{P} = \{\mathbf{x} \mid \mathbf{a}_j^T\mathbf{x} \leq b_j, j = 1, \ldots, m\}, \tag{2.2.8}$$

where $\mathbf{a}_j \in \mathbb{R}^n$ and $b_j \in \mathbb{R}$ for $j = 1, \ldots, m$. For $\mathbf{p} \in \mathcal{P}$ let

$$\mathcal{I}(\mathbf{p}) = \{i \mid \mathbf{a}_j^T \mathbf{p} = b_j\} \tag{2.2.9}$$

be the set of the inequalities that are active on $\mathbf{p}$. Then, $\mathbf{p}$ is a vertex of $\mathcal{P}$ if an only if the set of vectors $\{\mathbf{a}_j \mid j \in \mathcal{I}(\mathbf{p})\}$ linearly spans the vector space $\mathbb{R}^d$. In particular, if $\mathbf{p}$ is a vertex of $\mathcal{P}$, the set $\mathcal{I}(\mathbf{p})$ contains at least d indices; that is $|\mathcal{I}(\mathbf{p})| \geq d$.

Theorem 2.10 (Carathéodory)**.** Let $\mathcal{S} \subset \mathbb{R}^d$ be a set. Then, every point $\mathbf{s} \in \mathit{conv}\ \mathcal{S}$ can be represented as a convex combination of $d+1$ points from $\mathcal{S}$:

$$\mathbf{s} = \alpha_1\ \mathbf{y}_1 + \cdots + \alpha_{d+1}\ \mathbf{y}_{d+1},\ \text{where}\ \sum_{i=1}^{d+1} \alpha_i = 1,\ \alpha_i \geq 0, \tag{2.2.10}$$

and $\mathbf{y}_i \in \mathcal{S}$ for $i = 1, \ldots, d+1$.

Theorem 2.11. In $\mathbb{R}^n$, the convex hull of an open set is open, and the convex hull of a compact set is compact.

Proposition 2.12. Any finite union of closed sets is closed. The intersection of any family of closed sets is closed.

Corollary 2.13. A finite union of compact sets is compact. The intersection of any family of compact sets is compact.

Theorem 2.14 (Main Theorem of Polytope Theory)**.** The definition of $\mathcal{V}$-polytope and of $\mathcal{H}$-polytopes are equivalent. That is, every $\mathcal{V}$-polytope has a description by a finite system of inequalities, and every $\mathcal{H}$-polytope can be obtained as the convex hull of a finite set of points (its vertices).

Under some regularity conditions (polyhedron is fully dimensional and contains at least one vertex), we are able to transform between the two $\mathcal{H}$- to $\mathcal{V}$-polytope representations starting from either representation. The transformation from $\mathcal{H}$- to $\mathcal{V}$-polytope representation is known as the *vertex enumeration* problem, and its counterpart is the *facet enumeration* problem. The vertex enumeration problem can be solved by using the Fourier-Motzkin elemination algorithm. This algorithm works by eliminating variables from a system of linear inequalities over the reals. Alternatively, the facet enumeration (convex hull) problem could be obtained using the Beneath-Beyond algorithm. In this thesis, we will use instead the *Double Description* method, Fukuda and Prodon [1995].

Definition 2.17. A pair $(\mathbf{A}, \mathbf{R})$ of real matrices $\mathbf{A}$ and $\mathbf{R}$ is said to be a double description pair if the set $\mathcal{P}(\mathbf{A})$ represented by $\mathbf{A}$ as

$$\mathcal{P}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{R}^d \mid \mathbf{A}\mathbf{x} \geq \mathbf{0}\} \tag{2.2.11}$$

is simultaneously represented by $\mathbf{R}$ as

$$\{\mathbf{x} \in \mathbb{R}^d \mid \mathbf{x} = \mathbf{R}\boldsymbol{\lambda} \text{ for some } \boldsymbol{\lambda} \geq \mathbf{0}\}. \tag{2.2.12}$$

Theorem 2.15 (Minkowski's Theorem for Polyhedral Cones)**.** For any $m \times d$ real matrix $\mathbf{A}$, there exists some $d \times n$ real matrix $\mathbf{R}$ such that $(\mathbf{A}, \mathbf{R})$ is a double description pair, or in other words, the cone $\mathcal{P}(\mathbf{A})$ is generated by $\mathbf{R}$.

Theorem 2.16 (Weyl's Theorem for Polyhedral Cones)**.** For any $d \times n$ real matrix $\mathbb{R}$, there exists some $m \times d$ real matrix $\mathbf{A}$ such that $(\mathbf{A}, \mathbf{R})$ is a double description pair, on in other words, the set generated by $\mathbf{R}$ is the cone $\mathcal{P}(\mathbf{A})$.

2.3 Volume for Bounded Convex Sets

It will also become indispensable to be able to calculate the volume of polytopes. One issue to keep in mind is that, in general, computing the exact volume of a polyhedron is NP hard. However, this might be acceptable for problems with low dimension.

Definition 2.18 (Cell)**.** In $\mathbb{R}^1$, a cell is simply a bounded convex subset of the real line (i.e., $\emptyset$, $\{a\}$, $[a, b]$, $[a, b)$, $(a, b]$, (a, b) for $a, b \in \mathbb{R}$ with $a < b$). A *cell* $\mathcal{I}$ in $\mathbb{R}^n$ is a set of the form

$$\mathcal{I} = \mathcal{I}_1 \times \cdots \times \mathcal{I}_n = \{(x_1, \ldots, x_n) \mid x_1 \in \mathcal{I}_1, \ldots x_n \in \mathcal{I}_n\},$$

where $\mathcal{I}_1, \ldots, \mathcal{I}_n$ are cells in $\mathbb{R}^1$.

Definition 2.19 (Elementary set)**.** An elementary set in $\mathbb{R}^n$ is a set which can be expressed as a finite union of pairwise disjoint cells in $\mathbb{R}^n$.

Theorem 2.17. Let $\mathcal{A}$ and $\mathcal{B}$ be elementary sets in $\mathbb{R}^n$. Then, $\mathcal{A} \cap \mathcal{B}$, $\mathcal{A} \setminus \mathcal{B}$, $\mathcal{A} \cup \mathcal{B}$, and $\mathcal{A} + \mathcal{B}$ are elementary sets.

Corollary 2.18. Every union of a finite number and every intersection of a finite non-zero number of elementary sets in $\mathbb{R}^n$ is an elementary set.

Corollary 2.19. The closure, the interior, and the boundary of an elementary set in $\mathbb{R}^n$ are elementary sets.

Definition 2.20. The *length* $\mathcal{L}(\mathcal{I})$ of a cell $\mathcal{I}$ in $\mathbb{R}^1$ is defined to be zero when $\mathcal{I}$ is empty or a singleton and to be $b-a$ when $\mathcal{I}$ is a cell of one of the forms $[a,b]$, $[a,b)$, $(a,b]$, (a,b) for $a,b \in \mathbb{R}$ with $a<b$. Suppose next that $\mathcal{I}$ is the cell $\mathcal{I}_1 \times \ldots \mathcal{I}_n$ in $\mathbb{R}^n$, where $\mathcal{I}_1,\ldots,\mathcal{I}_n$ are cells in $\mathbb{R}^1$. Then, the *volume* $v(\mathcal{I})$ of $\mathcal{I}$ is (uniquely) defined by the equation

$$v(\mathcal{I}) = \mathcal{L}(\mathcal{I}_1)\mathcal{L}(\mathcal{I}_2)\ldots\mathcal{L}(\mathcal{I}_n).$$

Theorem 2.20. Let $\mathcal{I}_0,\mathcal{I}_1,\ldots,\mathcal{I}_m$, where $m \geq 1$ be cells in $\mathbb{R}^n$ with $\mathcal{I}_1,\ldots,\mathcal{I}_m$ pairwise disjoint and having union $\mathcal{I}_0$. Then,

$$v(\mathcal{I}_0) = \sum_{i=1}^{m} v(\mathcal{I}_i).$$

Corollary 2.21. Suppose that $\mathcal{I}_1,\ldots,\mathcal{I}_m$ and $\mathcal{J}_1,\ldots,\mathcal{I}_p$ are partitions of an elementary set $\mathcal{A}$ in $\mathbb{R}^n$ into cells. Then,

$$\sum_{i=1}^{m} v(\mathcal{I}_i) = \sum_{j=1}^{p} v(\mathcal{J}_j).$$

Theorem 2.22. Let $\mathcal{A}$ and $\mathcal{B}$ be elementary sets in $\mathbb{R}^n$. Then,

$$v(\mathcal{A}\cup\mathcal{B}) + v(\mathcal{A}\cap\mathcal{B}) = v(\mathcal{A}) + v(\mathcal{B}).$$

Corollary 2.23. Let $\mathcal{A}_1,\ldots,\mathcal{A}_m$ be elementary sets in $\mathbb{R}^n$. Then,

$$v(\mathcal{A}_1\cup\cdots\cup\mathcal{A}_m) \leq v(\mathcal{A}_1)+\cdots+v(\mathcal{A}_m).$$

Corollary 2.24. Let $\mathcal{A}$ be an elementary set in $\mathbb{R}^n$. Then,

$$v(\text{int } \mathcal{A}) = v(\mathcal{A}) = v(\text{cl } \mathcal{A}),$$

and

$$v(\text{bd}\mathcal{A}) = 0.$$

Definition 2.21. Let $\mathcal{E}$ be the class of elementary sets in $\mathbb{R}^n$. Let $\mathcal{A}$ be a bounded set in $\mathbb{R}^n$. Its *inner-volume* $\underline{v}(\mathcal{A})$ is defined by

$$\underline{v}(\mathcal{A}) = \sup\{v(\mathcal{B}) \mid \mathcal{B}\subseteq\mathcal{A} \text{ and } \mathcal{B}\in\mathcal{E}\}. \tag{2.3.1}$$

Its *outer-volume* $\overline{v}(\mathcal{A})$ is defined by

$$\overline{v}(\mathcal{A}) = \inf\{v(\mathcal{B}) \mid \mathcal{B}\subseteq\mathcal{A} \text{ and } \mathcal{B}\in\mathcal{E}\}. \tag{2.3.2}$$

Theorem 2.25. Let $\mathcal{A}$ and $\mathcal{B}$ be bounded sets in $\mathbb{R}^n$. Then,

a) $\underline{v}(\mathcal{A}) \leq \overline{v}(\mathcal{A})$

b) $\underline{v}(\mathcal{A}) = v(\mathcal{A}) = \overline{v}(\mathcal{A})$ when $\mathcal{A}$ is an elementary set

c) $\underline{v}(\mathcal{A}) \leq \underline{v}(\mathcal{B})$ and $\overline{v}(\mathcal{A}) \leq \overline{v}(\mathcal{B})$ whenver $\mathcal{A} \subseteq \mathcal{B}$

d) $\underline{v}(\mathcal{A}) = \underline{v}(\text{int } \mathcal{A})$ and $\overline{v}(\mathcal{A}) = \overline{v}(\text{cl } \mathcal{A})$

e) $\underline{v}(\mathcal{A} \cup \mathcal{B}) + \underline{v}(\mathcal{A} \cap \mathcal{B}) \geq \underline{v}(\mathcal{A}) + \underline{v}(\mathcal{B})$ and $\overline{v}(\mathcal{A} \cup \mathcal{B}) + \overline{v}(\mathcal{A} \cap \mathcal{B}) \leq \overline{v}(\mathcal{A}) + \overline{v}(\mathcal{B})$

Theorem 2.26. The set $\mathcal{A}$ in $\mathbb{R}^n$ has volume if and only if, for each $\varepsilon > 0$, there are elementary sets $\mathcal{E}$ and $\mathcal{F}$ in $\mathbb{R}^n$ such that $\mathcal{E} \subseteq \mathcal{A} \subseteq \mathcal{F}$, and $v(\mathcal{F} \setminus \mathcal{E}) < \varepsilon$.

Theorem 2.27. Let $\mathcal{A}$ and $\mathcal{B}$ be sets in $\mathbb{R}^n$ which have volume. Then, the sets $\mathcal{A} \cup \mathcal{B}$, $\mathcal{A} \cap \mathcal{B}$ and $\mathcal{A} \setminus \mathcal{B}$ have volume.

Corollary 2.28. All unions of a finite number, and all intersection of a finite non-zero number, of sets in $\mathbb{R}^n$ which have volume also have volume.

Theorem 2.29. Let $\mathcal{A}$ be a set in $\mathbb{R}^n$ which has volume. Then, the sets int $\mathcal{A}$ and cl $\mathcal{A}$ have volume with

$$v(\text{int } \mathcal{A}) = v(\mathcal{A}) = v(\text{cl } \mathcal{A}).$$

Theorem 2.30. Let $\mathcal{A}$ and $\mathcal{B}$ be sets in $\mathbb{R}^n$ which have volume. Then,

$$v(\mathcal{A} \cup \mathcal{B}) + v(\mathcal{A} \cap \mathcal{B}) = v(\mathcal{A}) + v(\mathcal{B}).$$

Corollary 2.31. Let $\mathcal{A}$ and $\mathcal{B}$ be sets in $\mathbb{R}^n$ which have volume and are such that $\mathcal{A} \subseteq \mathcal{B}$. Then $v(\mathcal{B} \setminus \mathcal{A}) = v(\mathcal{B}) - v(\mathcal{A})$ and $v(\mathcal{A}) \leq v(\mathcal{B})$.

Corollary 2.32. Let $\mathcal{A}_1, \ldots, \mathcal{A}_m$ be sets in $\mathbb{R}^n$ which have volume. Then,

$$v(\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_m) \leq v(\mathcal{A}_1) + \cdots + v(\mathcal{A}_m),$$

with equality holding when $v(\mathcal{A}_i \cap \mathcal{A}_j) = 0$ for $1 \leq i < j \leq m$.

Theorem 2.33. A bounded set $\mathcal{A}$ in $\mathbb{R}^n$ has volume if and only if its boundary bd $\mathcal{A}$ has volume zero.

Theorem 2.34. Every bounded convex set in $\mathbb{R}^n$ has volume.

Corollary 2.35. Every bounded subset of a hyperplane in $\mathbb{R}^n$ has volume zero.

Lemma 2.36. Let $T : \mathbb{R}^n \to \mathbb{R}^n$ be the linear transformation given by $T(\mathbf{x}) = \mathbf{A}\mathbf{x}$ for $\mathbf{x} \in \mathbb{R}^n$, where $\mathbf{A}$ is an elementary matrix. Then, for each cell $\mathcal{I}$ in $\mathbb{R}^n$, the set $T(\mathcal{I})$ has volume $|\det \mathbf{A}| v(\mathcal{I})$.

Theorem 2.37. Let $T : \mathbb{R}^n \to \mathbb{R}^n$ be the affine transformation given by $T(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$ for $\mathbf{x} \in \mathbb{R}^n$, where $\mathbf{A}$ is an $n \times n$ real matrix, and $\mathbf{b} \in \mathbb{R}^n$. Then, for each set $\mathcal{S}$ in $\mathbb{R}^n$ that has volume, the set $T(\mathcal{S})$ has volume $|\det \mathbf{A}| v(\mathcal{S})$.

Corollary 2.38. Let $\mathcal{S}$ be a set in $\mathbb{R}^n$ which has volume. Then, $v(\lambda\mathcal{S} + \mathbf{s}) = \lambda^n v(\mathcal{S})$ for all $\lambda \geq 0$, and $\mathbf{s} \in \mathbb{R}^n$.

Corollary 2.39. Let $\mathcal{A}$ and $\mathcal{B}$ be congruent sets in $\mathbb{R}^n$, with $\mathcal{A}$ having volume. Then, $v(\mathcal{B}) = v(\mathcal{A})$.

If two sets $\mathcal{F}$ and $\mathcal{G}$ are congruent, then there exists an affine transformation $T : \mathbb{R}^n \to \mathbb{R}^n$ given by $T(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$ for $\mathbf{x} \in \mathbb{R}^n$, where $\mathbf{A}$ is an $n \times n$ orthogonal matrix, and $\mathbf{b} \in \mathbb{R}^n$, such that $T(F) = G$.

Theorem 2.40. Let $\mathcal{A}, \mathcal{A}_1, \ldots, \mathcal{A}_k, \ldots$ be non-empty compact convex sets in $\mathbb{R}^n$ such that $\mathcal{A}_k \to \mathcal{A}$ as $k \to \infty$. Then, $v(\mathcal{A}_k) \to v(\mathcal{A})$ as $k \to \infty$.

From now on we will indicate the dependence of v on d the dimension of the space. That is, v_d is the volume function in $\mathbb{R}^d$ (length in $\mathbb{R}^1$, area in $\mathbb{R}^2$, and volume in $\mathbb{R}^3$).

Theorem 2.41. Let $\mathcal{A}$ be a bounded convex set in $\mathbb{R}^d$. For each real number x, denote by $\mathcal{A}_x$ the intersection of $\mathcal{A}$ with the hyperplane x in $\mathbb{R}^d$. Let a and b be real numbers such that $a < b$, and $\mathcal{A}_x$ is empty whenever $x < a$ or $x > b$. Then,

$$v_d(\mathcal{A}) = \int_a^b v_{d-1}(\mathcal{A}_x)\ dx.$$

Corollary 2.42. Let $\mathcal{A}$ be a bounded convex set in $\mathbb{R}^d$, and let $\mathbf{u}$ be a unit vector in $\mathbb{R}^d$. For each real number x, denote by $\mathcal{A}_x$ the intersection of $\mathcal{A}$ with the hyperplane $\langle \mathbf{u}, \mathbf{x} \rangle = x$. Then,

$$v_d(\mathcal{A}) = \int_\infty^{-\infty} v_{d-1}(\mathcal{A}_x)\ dx.$$

All known algorithms for exact volume computation decompose a given polytope into simplices, and thus they all rely, explicitly or implicitly, on the volume formula of a simplex. The volume of a simplex $\Delta(\mathbf{v}_0, \ldots, \mathbf{v}_d)$ is expressed as follows:

$$v_d(\Delta(\mathbf{v}_0, \ldots, \mathbf{v}_d)) = \left| \frac{1}{d!} \det \left(\begin{bmatrix} \mathbf{v}_1 - \mathbf{v}_0 & \ldots & \mathbf{v}_d - \mathbf{v}_0 \end{bmatrix} \right) \right| = \left| \frac{1}{d!} \det \left(\begin{bmatrix} \mathbf{v}_0 & \ldots & \mathbf{v}_d \\ 1 & \ldots & 1 \end{bmatrix} \right) \right| \tag{2.3.3}$$

Theorem 2.43. Let $\mathbf{u}_1, \ldots, \mathbf{u}_m$ be the outward unit normals to the facets of an d-polytope $\mathcal{P}$ in $\mathbb{R}^d$ corresponding to the facets $\mathcal{F}_1, \ldots, \mathcal{F}_m$. Let h be the support function of $\mathcal{P}$. Then,

$$v_d(\mathcal{P}) = \frac{1}{d} \sum_{i=1}^{m} h(\mathbf{u}_i) v_{d-1}(\mathcal{F}_i), \quad \text{and}$$

$$\sum_{i=1}^{m} v_{d-1}(\mathcal{F}_i)\mathbf{u}_i = 0.$$

Lemma 2.44. Let $\mathcal{P}_1, \ldots, \mathcal{P}_m$ be polytopes in $\mathbb{R}^d$, and let $\lambda_1, \ldots, \lambda_m > 0$. Then, $\lambda_1\mathcal{P}_1 + \cdots + \lambda_m\mathcal{P}_m$ and $\mathcal{P}_1 + \cdots + \mathcal{P}_m$ have the same dimension, and the sets of outward unit normals to the $(d-1)$-faces of the two polytopes are equal.

Definition 2.22 (Monomial)**.** A *monomial* in a collection of variables $x_1, \ldots, x_n$ is a product

$$\mathbf{x}^{\mathbf{e}} = x_1^{e_1} x_2^{e_2} \ldots x_n^{e_n}, \tag{2.3.4}$$

where all of the exponents $e_1, \ldots, e_n$ are non-negative integers. The *total degree* of the monomial $\mathbf{x}^{\mathbf{e}}$ is the sum of the exponents: $|\mathbf{e}| = e_1 + \cdots + e_n$.

Definition 2.23 (Polynomial)**.** Let $\mathbb{F}$ be any field. We can form finite linear combinations of monomials with coefficients in $\mathbb{F}$. The resulting objects are known as *polynomials* in $x_1, \ldots, x_n$. Thus, a general polynomial in the variables $x_1, \ldots, x_n$ with coefficients in $\mathbb{F}$ has the form

$$f = \sum_{i=1}^{N} c_i \, \mathbf{x}^{\mathbf{e}_i} = \sum_{i=1}^{N} c_i \, x_1^{e_{1,i}} x_2^{e_{2,i}} \ldots x_n^{e_{n,i}}, \quad c_i \in \mathbb{F}. \tag{2.3.5}$$

We will denote by $\mathbb{F}[x_1, \ldots, x_n]$ the collection of all polynomials in $x_1, \ldots, x_n$ with coefficients in $\mathbb{F}$.

We will use the following terminology in dealing with polynomials:

Definition 2.24. Let $f = \sum_{i=1}^{N} c_i \, x_1^{e_{1,i}} x_2^{e_{2,i}} \ldots x_n^{e_{n,i}}$ be a polynomial in $\mathbb{F}[x_1, \ldots, x_n]$.

a) We call $c_i \in \mathbb{F}$ the ith coefficient of the monomial $x_1^{e_{1,i}} x_2^{e_{2,i}} \ldots x_n^{e_{n,i}}$.

b) If $c_i \neq 0$, then we call $c_i \, x_1^{e_{1,i}} x_2^{e_{2,i}} \ldots x_n^{e_{n,i}}$ a *term* of f.

c) The *total degree* of f, denoted $\deg(f)$, is the maximum $|\mathbf{e}_i| = e_{1,i} + \cdots + e_{n,i}$ such that the coefficient c_i is nonzero.

A polynomial $f(x_1, \ldots, x_n)$ is said to be *homogeneous* if all the monomials appearing in it with nonzero coefficients have the same total degree. Each homogeneous polynomial $p(\lambda_1, \ldots, \lambda_m)$ of

degree d can be uniquely represented in the form:

$$p(\lambda_1,\ldots,\lambda_m)=\sum_{\alpha_1=0}^{d}\cdots\sum_{\alpha_m=0}^{d}\delta_{d-(\alpha_1+\cdots+\alpha_m)}\ \frac{(\alpha_1+\cdots+\alpha_m)!}{\alpha_1!\ldots\alpha_m!}\ c_{\alpha_1\ldots\alpha_m}\ \lambda_1^{\alpha_1}\ldots\lambda_m^{\alpha_m} \tag{2.3.6}$$

Lemma 2.45. Let $\mathcal{P}_1,\ldots,\mathcal{P}_m$ be polytopes in $\mathbb{R}^d$. Then $v_d(\lambda_1\mathcal{P}_1+\cdots+\lambda_m\mathcal{P}_m)$ is, for all $\lambda_1,\ldots,\lambda_m>0$, a homogeneous polynomial of degree d in $\lambda_1,\ldots,\lambda_m$ with non-negative coefficients.

Theorem 2.46. Let $\mathcal{A}_1,\ldots,\mathcal{A}_m$ be compact convex sets in $\mathbb{R}^d$. Then $v_d(\lambda_1\mathcal{A}_1+\cdots+\lambda_m\mathcal{A}_m)$ is, for all $\lambda_1,\ldots,\lambda_m\geq 0$, a homogeneous polynomial of degree d in $\lambda_1,\ldots,\lambda_m$ with non-negative coefficients.

For integers $i_1,\ldots,i_d$ lying in $1,\ldots,m$, let

$$\nu_{i_1\ldots i_d}=c_{\alpha_1\ldots\alpha_m}\quad\text{and}\quad\lambda_{i_1}\ldots\lambda_{i_d}=\lambda_1^{\alpha_1}\ldots\lambda_m^{\alpha_m} \tag{2.3.7}$$

; then,

a) $\nu_{i_1\ldots i_d}$ remains unchanged when $i_1\ldots i_d$ are permuted, and

b) $p(\lambda_1,\ldots,\lambda_m)=\sum_{i_1=1}^{m}\cdots\sum_{i_d=1}^{m}\nu_{i_1\ldots i_d}\lambda_{i_1}\ldots\lambda_{i_d}$.

Moreover, the $\nu_{i_1\ldots i_d}$ are uniquely determined by a) and b). When

$$p(\lambda_1,\ldots,\lambda_m)=v_d(\lambda_1\mathcal{A}_1+\cdots+\lambda_m\mathcal{A}_m), \tag{2.3.8}$$

where $\mathcal{A}_1,\ldots,\mathcal{A}_m$ are compact compact sets in $\mathbb{R}^n$, and $\lambda_1,\ldots,\lambda_m\geq 0$, the numbers $\nu_{i_1\ldots i_d}$ are called the *mixed volumes* of $\mathcal{A}_1,\ldots,\mathcal{A}_m$.

Example 2.1. Let $d=3$ and $m=2$, then

$$\begin{aligned}
p(\lambda_1,\lambda_2) &= \sum_{i_1=1}^{2}\sum_{i_2=1}^{2}\sum_{i_3=1}^{2}\nu_{i_1i_2i_3}\ \lambda_{i_1}\lambda_{i_2}\lambda_{i_2}\\
&= \sum_{i_2=1}^{2}\sum_{i_3=1}^{2}\nu_{1i_2i_3}\ \lambda_1\lambda_{i_2}\lambda_{i_2}+\sum_{i_2=1}^{2}\sum_{i_3=1}^{2}\nu_{2i_2i_3}\ \lambda_2\lambda_{i_2}\lambda_{i_2}\\
&= \sum_{i_3=1}^{2}\nu_{11i_3}\ \lambda_1\lambda_1\lambda_{i_2}+\sum_{i_3=1}^{2}\nu_{12i_3}\ \lambda_1\lambda_2\lambda_{i_2}\sum_{i_3=1}^{2}\nu_{21i_3}\ \lambda_2\lambda_1\lambda_{i_2}+\sum_{i_3=1}^{2}\nu_{22i_3}\ \lambda_2\lambda_2\lambda_{i_2}\\
&= \nu_{111}\ \lambda_1\lambda_1\lambda_1+\nu_{112}\ \lambda_1\lambda_1\lambda_2+\nu_{121}\ \lambda_1\lambda_2\lambda_1+\nu_{122}\ \lambda_1\lambda_2\lambda_2+\\
&\quad \nu_{211}\ \lambda_2\lambda_1\lambda_1+\nu_{212}\ \lambda_2\lambda_1\lambda_2+\nu_{221}\ \lambda_2\lambda_2\lambda_1+\nu_{222}\ \lambda_2\lambda_2\lambda_2\\
&= \nu_{111}\ \lambda_1^3+(\nu_{112}+\nu_{121}+\nu_{211})\ \lambda_1^2\lambda_2+(\nu_{122}+\nu_{212}+\nu_{221})\ \lambda_1\lambda_2^2+\nu_{222}\ \lambda_2^3
\end{aligned}$$

Recall that $\nu_{112} = \nu_{121} = \nu_{211}$ and $\nu_{122} = \nu_{212} = \nu_{221}$. Consequently,

$$p(\lambda_1, \lambda_2) \quad = \quad \nu_{111}\ \lambda_1^3 + 3\ \nu_{112}\ \lambda_1^2\lambda_2 + 3\ \nu_{122}\ \lambda_1\lambda_2^2 + \nu_{222}\ \lambda_2^3$$

Example 2.2. Let $d = 3$ and $m = 2$, then

$$\begin{aligned}
p(\lambda_1, \lambda_2) &= \sum_{\alpha_1=0}^{3}\sum_{\alpha_2=0}^{3} \delta_{3-(\alpha_1+\alpha_2)}\ \frac{3!}{\alpha_1!\alpha_2!}\ c_{\alpha_1\alpha_2}\ \lambda_1^{\alpha_1}\lambda_2^{\alpha_2} \\
&= \sum_{\alpha_2=0}^{3} \delta_{3-(0+\alpha_2)}\ \frac{3!}{0!\alpha_2!}\ c_{0\alpha_2}\ \lambda_1^0\lambda_2^{\alpha_2} + \sum_{\alpha_2=0}^{3} \delta_{3-(1+\alpha_2)}\ \frac{3!}{1!\alpha_2!}\ c_{1\alpha_2}\ \lambda_1^1\lambda_2^{\alpha_2} + \qquad (2.3.9) \\
&\quad \sum_{\alpha_2=0}^{3} \delta_{3-(2+\alpha_2)}\ \frac{3!}{2!\alpha_2!}\ c_{2\alpha_2}\ \lambda_1^2\lambda_2^{\alpha_2} + \sum_{\alpha_2=0}^{3} \delta_{3-(3+\alpha_2)}\ \frac{3!}{3!\alpha_2!}\ c_{3\alpha_2}\ \lambda_1^3\lambda_2^{\alpha_2} \\
&= \delta_{3-(0+0)}\ \frac{3!}{0!0!}\ c_{00}\ \lambda_1^0\lambda_2^0 + \delta_{3-(0+1)}\ \frac{3!}{0!1!}\ c_{01}\ \lambda_1^0\lambda_2^1 + \delta_{3-(0+2)}\ \frac{3!}{0!2!}\ c_{02}\ \lambda_1^0\lambda_2^2 + \delta_{3-(0+3)}\ \frac{3!}{0!3!}\ c_{03}\ \lambda_1^0\lambda_2^3 + \\
&\quad \delta_{3-(1+0)}\ \frac{3!}{1!0!}\ c_{10}\ \lambda_1^1\lambda_2^0 + \delta_{3-(1+1)}\ \frac{3!}{1!1!}\ c_{11}\ \lambda_1^1\lambda_2^1 + \delta_{3-(1+2)}\ \frac{3!}{1!2!}\ c_{12}\ \lambda_1^1\lambda_2^2 + \delta_{3-(1+3)}\ \frac{3!}{1!3!}\ c_{13}\ \lambda_1^1\lambda_2^3 + \\
&\quad \delta_{3-(2+0)}\ \frac{3!}{2!0!}\ c_{20}\ \lambda_1^2\lambda_2^0 + \delta_{3-(2+1)}\ \frac{3!}{2!1!}\ c_{21}\ \lambda_1^2\lambda_2^1 + \delta_{3-(2+2)}\ \frac{3!}{2!2!}\ c_{22}\ \lambda_1^2\lambda_2^2 + \delta_{3-(2+3)}\ \frac{3!}{2!3!}\ c_{23}\ \lambda_1^2\lambda_2^3 + \\
&\quad \delta_{3-(3+0)}\ \frac{3!}{3!0!}\ c_{30}\ \lambda_1^3\lambda_2^0 + \delta_{3-(3+1)}\ \frac{3!}{3!1!}\ c_{31}\ \lambda_1^3\lambda_2^1 + \delta_{3-(3+2)}\ \frac{3!}{3!2!}\ c_{32}\ \lambda_1^3\lambda_2^2 + \delta_{3-(3+3)}\ \frac{3!}{3!3!}\ c_{33}\ \lambda_1^3\lambda_2^3 \\
&= \delta_{3-(0+3)}\ \frac{3!}{0!3!}\ c_{03}\ \lambda_1^0\lambda_2^3 + \delta_{3-(1+2)}\ \frac{3!}{1!2!}\ c_{12}\ \lambda_1^1\lambda_2^2 + \delta_{3-(2+1)}\ \frac{3!}{2!1!}\ c_{21}\ \lambda_1^2\lambda_2^1 + \delta_{3-(3+0)}\ \frac{3!}{3!0!}\ c_{30}\ \lambda_1^3\lambda_2^0 \\
&= \frac{3!}{0!3!}\ c_{03}\ \lambda_2^3 + \frac{3!}{1!2!}\ c_{12}\ \lambda_1^1\lambda_2^2 + \frac{3!}{2!1!}\ c_{21}\ \lambda_1^2\lambda_2^1 + \frac{3!}{3!0!}\ c_{30}\ \lambda_1^3 \\
&= c_{30}\ \lambda_1^3 + 3\ c_{21}\ \lambda_1^2\lambda_2^1 + 3\ c_{12}\ \lambda_1^1\lambda_2^2 + c_{03}\ \lambda_2^3
\end{aligned}$$

Theorem 2.47. Let $\mathcal{A}_1, \ldots, \mathcal{A}_m$ be non-empty compact convex sets in $\mathbb{R}^d$. Then, for all $\lambda_1, \ldots, \lambda_m \geq 0$:

$$v_d(\lambda_1\mathcal{A}_1 + \cdots + \lambda_m\mathcal{A}_m) = \sum_{i_1=1}^{m} \cdots \sum_{i_d=1}^{m} \mathcal{V}(\mathcal{A}_{i_1}, \ldots, \mathcal{A}_{i_d})\lambda_{i_1} \ldots \lambda_{i_d} \qquad (2.3.10)$$

That is, the $\nu_{i_1 \ldots i_d}$ depend only upon the sets $\mathcal{A}_{i_1}, \ldots, \mathcal{A}_{i_d}$.

Example 2.3. Let $d = 3$ and $m = 2$, and assume all of the compact convex sets are non-empty, then

$$p(\lambda_1, \lambda_2) \quad = \quad \nu_{111}\ \lambda_1^3 + 3\ \nu_{112}\ \lambda_1^2\lambda_2 + 3\ \nu_{122}\ \lambda_1\lambda_2^2 + \nu_{222}\ \lambda_2^3$$

and

$$\begin{aligned}
\nu_{111} &= \mathcal{V}(\mathcal{A}_1, \mathcal{A}_1, \mathcal{A}_1) \\
\nu_{112} &= \mathcal{V}(\mathcal{A}_1, \mathcal{A}_1, \mathcal{A}_2) \\
\nu_{122} &= \mathcal{V}(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_2) \\
\nu_{222} &= \mathcal{V}(\mathcal{A}_2, \mathcal{A}_2, \mathcal{A}_2).
\end{aligned}$$

Let $\mathcal{C}^d$ denote the set of compact, convex subsets of Euclidean d-space, $\mathbb{R}^d$. Let $\mathcal{K}^d$ denote the subset of $\mathcal{C}^d$ with non-empty interiors in $\mathbb{R}^d$.

For compact, convex sets $\mathcal{A}_1, \ldots, \mathcal{A}_m \in \mathcal{C}^d$ and real numbers $\lambda_1, \ldots, \lambda_m \geq 0$, the Minkowski linear combination $\lambda_1\mathcal{A}_1 + \cdots + \lambda_m\mathcal{A}_m \in \mathcal{C}^d$ is defined by

$$\lambda_1\mathcal{A}_1 + \cdots + \lambda_m\mathcal{A}_m = \{\lambda_1\mathbf{x}_1 + \cdots + \lambda_m\mathbf{x}_m \in \mathbb{R}^d \mid \mathbf{x}_i \in \mathcal{A}_i\}. \tag{2.3.11}$$

Definition 2.25 (basic properties of the mixed volumes)**.** The following is a list of the basic properties of the mixed volume functional

$$\mathcal{V} : \underbrace{\mathcal{C}^d \times \cdots \times \mathcal{C}^d}_{d} \to [0, \infty) \tag{2.3.12}$$

a) It is symmetric in its arguments.

b) It is linear in each of its arguments with respect to Minkowski linear combinations; i.e., if $\mathcal{A}_1, \ldots, \mathcal{A}_{d-1} \in \mathcal{C}^d$ and $\tilde{\mathcal{A}} = (\mathcal{A}_1, \ldots, \mathcal{A}_{d-1})$, then for $\mathcal{A}, \mathcal{B} \in \mathcal{C}^n$ and $\lambda, \mu \geq 0$,

$$\mathcal{V}(\tilde{\mathcal{A}}, \lambda\mathcal{A} + \mu\mathcal{B}) = \lambda\ \mathcal{V}(\tilde{\mathcal{A}}, \mathcal{A}) + \mu\ \mathcal{V}(\tilde{\mathcal{A}}, \mathcal{B}). \tag{2.3.13}$$

c) Its diagonal form reduces to ordinary volume; i.e., for $\mathcal{A} \in \mathcal{C}^d$,

$$\mathcal{V}(\mathcal{A}, \ldots, \mathcal{A}) = v_d(\mathcal{A}). \tag{2.3.14}$$

d) It is continuous (in fact, uniformly continuous) in each argument, with respect to the Hausdorff metric.

e) It is invariant under independent translations of its arguments; i.e., if $\mathcal{A}_i \in \mathcal{C}^d$ and $\mathbf{x}_i \in \mathbb{R}^d$, then

$$\mathcal{V}(\mathbf{x}_1 + \mathcal{A}_1, \ldots, \mathbf{x}_d + \mathcal{A}_d) = \mathcal{V}(\mathcal{A}_1, \ldots, \mathcal{A}_d). \tag{2.3.15}$$

It is well-known that the mixed volume of the convex polytopes $\mathcal{P}_1, \ldots, \mathcal{P}_d$ in $\mathbb{R}^d$, $\mathcal{V}(\mathcal{P}_1, \ldots, \mathcal{P}_d)$, is a non-negative continuous function in d variables on the set of convex polytopes, symmetric in the variables $\mathcal{P}_i$, and monotonic with respect to the subset partial order on convex polytopes.

The mixed volume function is linear in each of its arguments in some restricted sense.

Theorem 2.48. Let $\tilde{\mathcal{A}}, \mathcal{A}_1, \ldots, \mathcal{A}_d$ be non-empty compact convex sets in $\mathbb{R}^d$. Let $\tilde{\lambda}, \lambda_1 \geq 0$. Then,

$$\mathcal{V}(\tilde{\lambda}\ \tilde{\mathcal{A}} + \lambda_1\ \mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_d) = \tilde{\lambda}\ \mathcal{V}(\tilde{\mathcal{A}}, \mathcal{A}_2, \ldots, \mathcal{A}_d) + \lambda_1\ \mathcal{V}(\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_d). \tag{2.3.16}$$

Lemma 2.49. Let m be a positive integer. For each $i = 0, 1, 2, \ldots$, let

$$P_i(x) = a_{im}x^m + \cdots + a_{i1}x + a_{i0} \tag{2.3.17}$$

be a real polynomial. Suppose that for each $x \geq 0$, $P_i(x) \to P_0(x)$ as $i \to \infty$. Then, $a_{ij} \to a_{0j}$ as $i \to \infty$, for $j = 0, 1, \ldots, m$.

Theorem 2.50 (Continuity of mixed volumes)**.** For each $j = 1, \ldots, d$, let $\mathcal{A}_j^1, \ldots, \mathcal{A}_j^i, \ldots$ be a sequence of non-empty compact convex sets converging to a non-empty compact convex set $\mathcal{A}_j^0$ in $\mathbb{R}^d$. Then

$$\mathcal{V}(\mathcal{A}_1^i, \ldots, \mathcal{A}_d^i) \to \mathcal{V}(\mathcal{A}_1^0, \ldots, \mathcal{A}_d^0) \text{ as } i \to \infty. \tag{2.3.18}$$

Theorem 2.51. Let $\mathbf{u}_1, \ldots, \mathbf{u}_m$ be the outward unit normals to the $(d-1)$-faces of a polytope $\mathcal{P}$ in $\mathbb{R}^d$ corresponding to faces $\mathcal{F}_1, \ldots, \mathcal{F}_m$, respectively. Then, for any non-empty compact convex set $\mathcal{A}$ in $\mathbb{R}^d$ with support function h,

$$\lim_{\lambda \to 0_+} \frac{v_d(\mathcal{P} + \lambda\mathcal{A}) - v_d(\mathcal{P})}{\lambda} = \sum_{i=1}^{m} h(\mathbf{u}_i) v_{d-1}(\mathcal{F}_i). \tag{2.3.19}$$

Corollary 2.52. Let $\mathbf{u}_1, \ldots, \mathbf{u}_m$ be the outward unit normals to the $(d-1)$-faces of a non-empty polytope $\mathcal{P}$ in $\mathbb{R}^d$ corresponding to faces $\mathcal{F}_1, \ldots, \mathcal{F}_m$, respectively. Then, for any non-empty compact convex set $\mathcal{A}$ in $\mathbb{R}^d$ with support function h

$$\mathcal{V}(\mathcal{A}, \underbrace{\mathcal{P}, \ldots, \mathcal{P}}_{d-1}) = \frac{1}{d} \sum_{i=1}^{m} h(\mathbf{u}_i) v_{d-1}(\mathcal{F}_i) \tag{2.3.20}$$

Theorem 2.53. Let $\mathcal{P}_2, \ldots, \mathcal{P}_d$ be non-empty polytopes in $\mathbb{R}^d$ $(d \geq 2)$. Let $\mathbf{u}_1, \ldots, \mathbf{u}_m$ be the outward unit normals to the $(d-1)$-faces of $\mathcal{P}_2 + \cdots + \mathcal{P}_d$. Then, there are scalars $\alpha_1, \ldots, \alpha_m \geq 0$ such that, for every non-empty compact convex set $\mathcal{A}$ in $\mathbb{R}^d$ with support function h,

$$\mathcal{V}(\mathcal{A}, \mathcal{P}_2, \ldots, \mathcal{P}_d) = \frac{1}{d} \sum_{i=1}^{m} \alpha_i h(\mathbf{u}_i). \tag{2.3.21}$$

Theorem 2.54. Let $\mathcal{A}_1, \ldots, \mathcal{A}_d, \mathcal{B}_1, \ldots, \mathcal{B}_d$ be non-empty convex sets in $\mathbb{R}^d$ with $\mathcal{A}_1 \subseteq \mathcal{B}_1, \ldots, \mathcal{A}_d \subseteq \mathcal{B}_d$. Then, $\mathcal{V}(\mathcal{A}_1, \ldots, \mathcal{A}_d) \leq \mathcal{V}(\mathcal{B}_1, \ldots, \mathcal{B}_d)$.

Theorem 2.55 (Brunn's Inequality)**.** Let $\mathcal{A}$, $\mathcal{B}$, $\mathcal{A} + \mathcal{B}$ be non-empty sets in $\mathbb{R}^d$, all of which have

volume. Then,

$$v(\mathcal{A}+\mathcal{B})^{1/d} \geq v(\mathcal{A})^{1/d} + v(\mathcal{B})^{1/d}. \tag{2.3.22}$$

Corollary 2.56. Let $\mathcal{A}$, $\mathcal{B}$ be non-empty bounded convex sets in $\mathbb{R}^d$. Then, the function $f : [0,1] \to \mathbb{R}$, defined by the equation

$$f(t) = v((1-t)\mathcal{A} + t\mathcal{B})^{1/d} \quad \text{for } t \in [0,1], \tag{2.3.23}$$

is concave.

Theorem 2.57 (Minkowski's inequality for mixed volumes)**.** Let $\mathcal{A}$ and $\mathcal{B}$ be convex bodies in $\mathbb{R}^d$. Then,

$$\mathcal{V}(\underbrace{\mathcal{A},\ldots,\mathcal{A}}_{d-1},\mathcal{B})^d \geq v(\mathcal{A})^{d-1}v(\mathcal{B}) \tag{2.3.24}$$

with equality holding if and only if

$$v(\mathcal{A}+\mathcal{B})^{1/d} = v(\mathcal{A})^{1/d} + v(\mathcal{B})^{1/d}. \tag{2.3.25}$$

Theorem 2.58 (Brunn-Minkowski)**.** Let $\mathcal{A}$ and $\mathcal{B}$ be convex bodies in $\mathbb{R}^d$. Then

$$v(\mathcal{A}+\mathcal{B})^{1/d} \geq v(\mathcal{A})^{1/d} + v(\mathcal{B})^{1/d} \tag{2.3.26}$$

with equality holding if and only if $\mathcal{A}$ and $\mathcal{B}$ are homothetic.

Theorem 2.59. Let $\mathcal{A}$ and $\mathcal{B}$ be convex bodies in $\mathbb{R}^d$. Then,

$$\mathcal{V}(\underbrace{\mathcal{A},\ldots,\mathcal{A}}_{d-1},\mathcal{B})^d \geq v(\mathcal{A})^{d-1}v(\mathcal{B}) \tag{2.3.27}$$

with equality holding if and only if $\mathcal{A}$ and $\mathcal{B}$ are homothetic.

Theorem 2.60 (Aleksandrov-Fenchel Inequality)**.** Let $\mathcal{A}_1,\ldots,\mathcal{A}_d$ be non-empty compact convex sets in $\mathbb{R}^d$. Then,

$$\mathcal{V}(\mathcal{A}_1,\ldots,\mathcal{A}_d)^r \geq \prod_{j=1}^{r} \mathcal{V}(\underbrace{\mathcal{A}_j,\ldots,\mathcal{A}_j}_{r},\mathcal{A}_{r+1},\ldots,\mathcal{A}_d). \tag{2.3.28}$$

For the volume computation performed in this thesis, we will make use of the VINCI software package, Büeler et al. [2000]. This package contains many useful methods for computing the volume of a polytope.

Chapter 3

Piecewise Polynomial Curves and Surfaces

As previously stated, we will restrict our search for minimizers to optimal control problems in the finite-dimensional space of piecewise polynomial functions with a prescribed number of polynomial pieces, order and smoothness. This space will be identified by $\mathcal{P}_{\mathbf{b},o,s}$, where $\mathbf{b} = \{b_0, \ldots, b_{N_p}\}$ are the $(N_p + 1)$ breakpoints, specifying the sites at which the endpoints of the N_p polynomial pieces of order o reside and are being joined with smoothness $s_j = s$, $j = 1, \ldots, N_p - 1$.

Let us begin by showing how we intend to use piecewise polynomial pieces to approximate the minimizers for optimal control problems. Let $c(t),\ t \in [t_0, t_f]$, be a real-valued function defined on the interval $\mathcal{I} = [t_0, t_f]$. We want to construct a piecewise polynomial function $z(t)$ that interpolates $c(t)$ at the breakpoints. As we briefly mentioned before, the breakpoints set is an increasing sequence of real values representing the endpoints of the N_p polynomial pieces

$$t_0 = b_0 < b_1 < \cdots < b_{N_p-1} < b_{N_p} = t_f. \tag{3.0.1}$$

On each interval, $\mathcal{I}_\ell = [b_\ell, b_{\ell+1}]$, $z(t)$ is constructed as a polynomial $p_\ell(t)$ of degree d:

$$p_\ell(t) = c_{0,\ell} + c_{1,\ell}(t - b_\ell) + c_{2,\ell}(t - b_\ell)^2 + \cdots + c_{d,\ell}(t - b_\ell)^d, \quad t \in [b_\ell, b_{\ell+1}], \quad \ell = 0, \ldots, N_p - 1. \tag{3.0.2}$$

For N_p polynomial pieces we have that $c(t)$, $t \in [t_0, t_f]$ can be expressed as follows:

$$c(t) = \begin{cases} p_0(t) & t \in [b_0, b_1] \\ p_1(t) & t \in [b_1, b_2] \\ \vdots & \\ p_{N_p-1}(t) & t \in [b_{N_p-1}, b_{N_p}]. \end{cases} \tag{3.0.3}$$

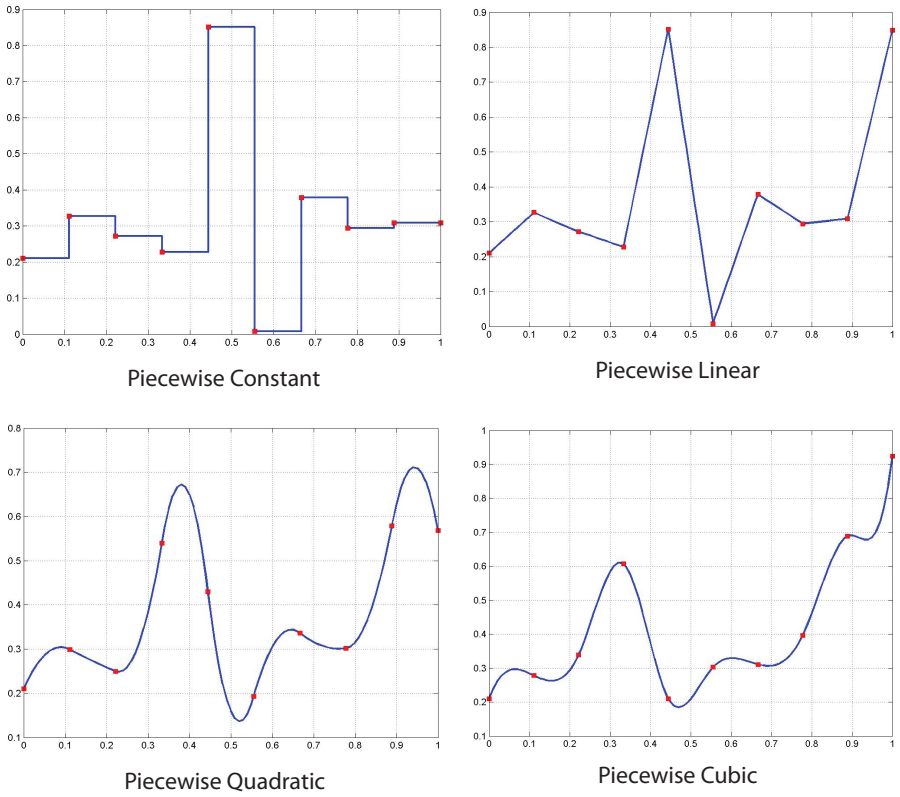

Figure 3.1: Examples of piecewise polynomial functions with polynomial pieces of the same degree.

From the interpolation properties

$$z(b_\ell) = c(b_\ell), \qquad \ell = 0, \ldots, N_p \tag{3.0.4}$$

we have that

$$p_\ell(b_\ell) = c(b_\ell), \qquad p_\ell(b_{\ell+1}) = c(b_{\ell+1}), \qquad \ell = 0, \ldots, N_p - 1, \tag{3.0.5}$$

and, thus, $z(t)$ is guaranteed to be continuous on $\mathcal{I}$:

$$p_{\ell-1}(b_\ell) = p_\ell(b_\ell), \qquad \ell = 1, \ldots, N_p - 1. \tag{3.0.6}$$

We continue in this manner matching derivatives until we reach the desired smoothness of the curve. For example, if we want to ensure C^1 continuity we would proceed by imposing the following conditions to be satisfied in addition to the one above, namely

$$\dot{p}_\ell(b_\ell) = \dot{c}(b_\ell), \qquad \dot{p}_\ell(b_{\ell+1}) = \dot{c}(b_{\ell+1}), \qquad \ell = 0, \ldots, N_p - 1, \tag{3.0.7}$$

which leads to C^1 continuity of $z(t)$ on $\mathcal{I}$:

$$\dot{p}_{\ell-1}(b_\ell) = \dot{p}_\ell(b_\ell), \qquad \ell = 1, \ldots, N_p - 1. \tag{3.0.8}$$

The objective is to determine the polynomial coefficients $c_{0,\ell}, c_{1,\ell}, c_{2,\ell}, \ldots, c_{d,\ell}$ for the N_p polynomials. In order to obtain unique solutions for the coefficients, there are only a few combinations of degree and smoothness that can be imposed. In particular, if $d = 1$ one must impose C^0 smoothness, if $d = 3$ one must impose C^1 smoothness, for $d = 5$ one must impose C^2 smoothness, and so forth. That is, if a C^s smooth curve is required, one must use a polynomial of degree $d = 2s + 1$.

The reader is referred to the work of Kraft [1985] and von Stryk and Bulirsch [1992] for a detailed discussion on how to use this parameterization to solve optimal control problems using the method of *collocation*.

As mentioned before, it will be more convenient to express piecewise polynomial curves as a linear combination of NURBS basis functions. NURBS basis functions have a large set of useful properties, which we will exploit for trajectory generation. In particular, NURBS are rational functions of B-spline basis functions, and many of their properties are inherited from them. Consequently, we will begin by studying B-splines first.

3.1 B-spline Curves and Surfaces

The vector space $\mathcal{P}_{\mathbf{b},o,\mathbf{s}}$ is finite dimensional, and it can be efficiently represented in terms of B-spline basis functions, de Boor [1978], which, naturally, are piecewise polynomial functions themselves. To specify a particular piecewise polynomial function $c(t)$ in this representation, one is required to supply the following parameters: the order of the required polynomial pieces, o, a union set of the position of the *breakpoints* or end-points of the polynomial pieces being pasted together, $\mathbf{b} = \{b_0, \ldots, b_{N_p}\}$ (where N_p is equal to the number of polynomial pieces), and the desired smoothness of the curve at the breakpoints, $\mathbf{s} = \{s_0, \ldots, s_{N_p}\}$ with $s_j \in [-1, d]$, $j = 0, \ldots, N_p$. Then, at each of the breakpoints, the curve $c(t)$ has the following continuity: $c(b_j) \in C^{s_j}$, where $s_j = -1$ indicates that the curve $c(t)$ is discontinuous at that point, $s_j = 0$ indicates that the curve $c(t)$ is continuous at that point, $s_j = 1$ indicates that the curve $c(t)$ is differentiable at that point, and so on. At all other points $c(t)$ is C^∞.

The set of all piecewise polynomial functions of order o on the breakpoint set $\{b_j\}$ at which $c(b_j)$ is C^{s_j} continuous forms a vector space $\mathcal{P}_{\boldsymbol{b},o,\boldsymbol{s}}$. The dimension of this vector space is $N_c = \dim(\mathcal{P}_{\boldsymbol{b},o,\boldsymbol{s}})$.

To determine the size of this space we must construct the *knotpoints* set. If no continuity constraints are imposed ($s_j = -1$ for all j), then the dimension of $\mathcal{P}_{\mathbf{b},o,\mathbf{s}}$ is equal to $N_c = N_p\ o$ (that is, the number of polynomials times their order). Each continuity constraint decreases the dimension by one; thus

$$N_c = \dim(\mathcal{P}_{\mathbf{b},o,\mathbf{s}}) = N_p\ o - \sum_{j=0}^{N_p}(s_j + 1). \tag{3.1.1}$$

Let us define the multiplicity of the breakpoint j as $m_j = d - s_j$. Substituting for s_j in the above definition, we have

$$\begin{aligned} N_c &= N_p\ o - \sum_{j=0}^{N_p}(s_j+1) = N_p\ o - \sum_{j=0}^{N_p}(d - m_j + 1) = N_p\ o - (N_p+1)o + \sum_{j=0}^{N_p} m_j = \sum_{j=0}^{N_p} m_j - o \\ N_c &= N_k - o. \end{aligned} \tag{3.1.2}$$

The dimension of the vector space, N_c, for a specific curve is then related to the order of the polynomials being pieced together and the dimension of the knotpoints vector:

$$N_k = \dim(\mathbf{k}) = \sum_{j=0}^{N_p} m_j. \tag{3.1.3}$$

The knotpoints vector, $\mathbf{k}$, is constructed from the breakpoints vector and the desired smoothness or continuity at those points. To be more precise, the knotpoints vector contains the breakpoints with their respective multiplicity, m_j. The breakpoints have multiplicity $m_j = d - s_j$, where s_j is the desired continuity of the curve $c(t)$ at the breakpoint b_j. With this information, one can build the knotpoints vector:

$$\begin{aligned} \mathbf{k} &= \left(b_0^1, \ldots, b_0^{m_0}, b_1^1, \ldots, b_1^{m_1}, \ldots, b_{N_p-1}^1, \ldots, b_{N_p-1}^{m_{N_p-1}}, b_{N_p}^1, \ldots, b_{N_p}^{m_{N_p}}\right) \\ &= \left(k_1, \ldots, k_{m_0}, k_{m_0+1}, \ldots, k_{m_0+m_1}, \ldots, k_{\sum_{j=0}^{N_p-2} m_j+1}, \ldots, k_{\sum_{j=0}^{N_p-1} m_j}, k_{\sum_{j=0}^{N_p-1} m_j+1}, \ldots, k_{\sum_{j=0}^{N_p} m_j}\right). \end{aligned} \tag{3.1.4}$$

In this thesis, we will make use of a specific type of knotpoints vector known as non-periodic (or clamped or open). This knotpoints vector is characterized for having the multiplicity of the end breakpoints to be of multiplicity equal to the order of the polynomial pieces (or by having the end points be discontinuous); that is, $m_0 = m_{N_p} = o$ (or $s_0 = s_{N_p} = -1$). In this case, we have that

$N_k = 2\ o + \sum_{j=1}^{N_p-1} m_j$; consequently,

$$N_c = o + \sum_{j=1}^{N_p-1} m_j. \tag{3.1.5}$$

Although one can specify the smoothness or continuity of the curve $c(t)$ at each of the internal breakpoints independently, in this thesis we will be interested instead in specifying the smoothness of the whole curve (e.g., $c(t) \in C^s$). In this case we have that $s_1 = \cdots = s_{N_p-1} = s$ and $c(t) \in C^s$, $s \in [-1, d]$. Since the smoothness is the same for all breakpoints, we have that the multiplicity is also constant (say, m) and, therefore, $N_k = 2\ o + m(N_p - 1)$, and

$$N_c = o + m(N_p - 1) = (d + 1) + (d - s)(N_p - 1). \tag{3.1.6}$$

As a consequence of the above modifications, the knotpoints vector now reflects the new multiplicity rules of the breakpoints: end breakpoints have multiplicity equal to o, and the internal breakpoints have multiplicity equal to m

$$\begin{aligned} \mathbf{k} &= \left(b_0^1, \ldots, b_0^o, b_1^1, \ldots, b_1^m, \ldots, b_{N_p-1}^1, \ldots, b_{N_p-1}^m, b_{N_p}^1, \ldots, b_{N_p}^o\right) \\ &= \left(k_1, \ldots, k_o, k_{o+1}, \ldots, k_{o+m}, \ldots, k_{o+m(N_p-2)+1}, \ldots, k_{o+m(N_p-1)}, k_{o+m(N_p-1)+1}, \ldots, k_{2o+m(N_p-1)}\right). \end{aligned} \tag{3.1.7}$$

That is, the end breakpoints with multiplicity o are repeated o times, and the internal breakpoints with multiplicity m are repeated m times.

One way in which the breakpoints can be set is by evenly distributing them in the interval $[t_0, t_f]$ (uniformly), i.e., each $b_j \in \mathbf{b}$ is calculated from $b_j = t_0 + jb_s$, $j = 0, \ldots, N_p$, where $b_s = \frac{t_f - t_0}{N_p}$. This will be the convention adopted in this thesis unless stated otherwise in a specific application.

In this new context, the set of all piecewise polynomial functions of order o on the uniformly distributed breakpoint sequence $\{b_j\}$ at which $c(b_j)$ is C^s continuous forms a vector space $\mathcal{P}_{\mathbf{b},o,s}$. The dimension of the space is obtained by

$$\dim\left(\mathcal{P}_{\mathbf{b},o,s}\right) = N_c = N_p(o - (s + 1)) + (s + 1). \tag{3.1.8}$$

Consequently, a curve $c(t) \in \mathcal{P}_{\mathbf{b},o,s}$ can be expressed in terms of the B-spline basis functions as follows:

$$c(t, p_0, \ldots, p_{N_c-1}) = \sum_{j=0}^{N_c-1} \mathcal{B}_{j,d}^{(0)}(t)\ p_j,\ t \in [t_0, t_f],\ p_j \in (-\infty, \infty), \tag{3.1.9}$$

where $\mathcal{B}_{j,d}^{(0)}(t)$ is the 0th time derivative of the jth B-Spline basis function of degree d, and p_j is the

corresponding jth control point.

Proposition 3.1. Consider $c(t, \mathbf{p})$ to be a smooth B-spline curve. Then, its rth order partial derivatives with respect to time and their first- and second-order partial derivatives with respect to control points are obtained as follows:

$$\begin{aligned} c^{(r)}(t,\mathbf{p}) &= \left[\mathcal{B}_d^{(r)}(t)\right]^T \mathbf{p} \\ \mathbf{D_p}\left[c^{(r)}(t,\mathbf{p})\right] &= \left[\mathcal{B}_d^{(r)}(t)\right]^T \\ \mathbf{D_{pp}}\left[c^{(r)}(t,\mathbf{p})\right] &= \mathbf{0}, \end{aligned}$$

with $t \in [t_0, t_f]$, $p_j \in (-\infty, \infty)$, $\left[\mathcal{B}_d^{(r)}(t)\right] = \begin{bmatrix} \mathcal{B}_{0,d}^{(r)}(t) \\ \vdots \\ \mathcal{B}_{N_c-1,d}^{(r)}(t) \end{bmatrix}$, and $\mathbf{p} = \begin{bmatrix} p_0 \\ \vdots \\ p_{N_c-1} \end{bmatrix}$.

A B-spline surface is obtained by taking a bidirectional net of control points, two knotpoints vectors and the products of the univariate B-spline functions:

$$\mathcal{S}(x,y) = \sum_{i=0}^{N_c^u-1} \sum_{j=0}^{N_c^v-1} \mathcal{B}_{i,p}^{(0)}(x)\ \mathcal{B}_{j,q}^{(0)}(y)\ \mathbf{p}_{i,j}, \tag{3.1.10}$$

with

$$\mathbf{k}^x = \left(b_0^1, \ldots, b_0^{p+1}, b_1^1, \ldots, b_1^{m_1^x}, \ldots, b_{N_p-1}^1, \ldots, b_{N_p-1}^{m_{N_p-1}^x}, b_{N_p}^1, \ldots, b_{N_p}^{p+1}\right) \tag{3.1.11}$$

$$\mathbf{k}^y = \left(d_0^1, \ldots, d_0^{q+1}, d_1^1, \ldots, d_1^{m_1^y}, \ldots, d_{N_p-1}^1, \ldots, d_{N_p-1}^{m_{N_p-1}^y}, d_{N_p}^1, \ldots, d_{N_p}^{q+1}\right) \tag{3.1.12}$$

$\mathbf{k}^x$ has N_k^x knotpoints, and $\mathbf{k}^y$ has N_k^y knotpoints, and $N_c^x = N_k^x - (p+1)$, and $N_c^y = N_k^y - (q+1)$.

Let (x, y) be fixed. Generally, one is interested in computing all partial derivatives of $\mathcal{S}(x, y)$ up to and including order o ($o > p, q$ is allowed), that is

$$\frac{\partial^{k+\ell}}{\partial^k x\ \partial^\ell y}\mathcal{S}(x,y), \qquad k+\ell \in [0, o]. \tag{3.1.13}$$

As for curves, we obtain these derivatives by computing derivatives of the basis functions. In particular,

$$\frac{\partial^{k+\ell}}{\partial^k x\ \partial^\ell y}\mathcal{S}(x,y) = \sum_{i=0}^{N_c^x-1} \sum_{j=0}^{N_c^y-1} \mathcal{B}_{i,p}^{(k)}(x)\ \mathcal{B}_{j,q}^{(\ell)}(y)\ \mathbf{p}_{i,j}. \tag{3.1.14}$$

The B-spline basis functions are computed recursively as follows:

$$\mathcal{B}_{j,d}^{(0)}(t) = \begin{cases} 1, & \text{if } t \in [k_j, k_{j+1}) \\ 0, & \text{otherwise} \end{cases}, \quad d = 0 \tag{3.1.15}$$

$$\mathcal{B}_{j,d}^{(0)}(t) = \frac{t - k_j}{k_{j+d} - k_j}\mathcal{B}_{j,d-1}^{(0)}(t) + \frac{k_{j+d+1} - t}{k_{j+d+1} - k_{j+1}}\mathcal{B}_{j+1,d-1}^{(0)}(t), \quad d > 0, \tag{3.1.16}$$

and the rth time derivative of the basis function $\mathcal{B}_{j,d}^{(0)}(t)$ is given by

$$\mathcal{B}_{j,d}^{(r)}(t) = \frac{d}{d-r}\left(\frac{t - k_j}{k_{j+d} - k_j}\mathcal{B}_{j,d-1}^{(r)}(t) + \frac{k_{j+d+1} - t}{k_{j+d+1} - k_{j+1}}\mathcal{B}_{j+1,d-1}^{(r)}(t)\right), \; r = 0, \ldots, d-1. \tag{3.1.17}$$

Alternatively, the kth derivative of $\mathcal{B}_{j,d}^{(0)}(t)$

$$\mathcal{B}_{j,d}^{(r)}(t) = d\left(\frac{1}{k_{j+d} - k_j}\mathcal{B}_{j,d-1}^{(r-1)}(t) - \frac{1}{k_{j+d+1} - k_{j+1}}\mathcal{B}_{j+1,d-1}^{(r-1)}(t)\right), \; r = 1, \ldots, d-1, \tag{3.1.18}$$

where k_j are the non-periodic knotpoints. Figure 3.2 illustrates the B-spline basis for different degrees.

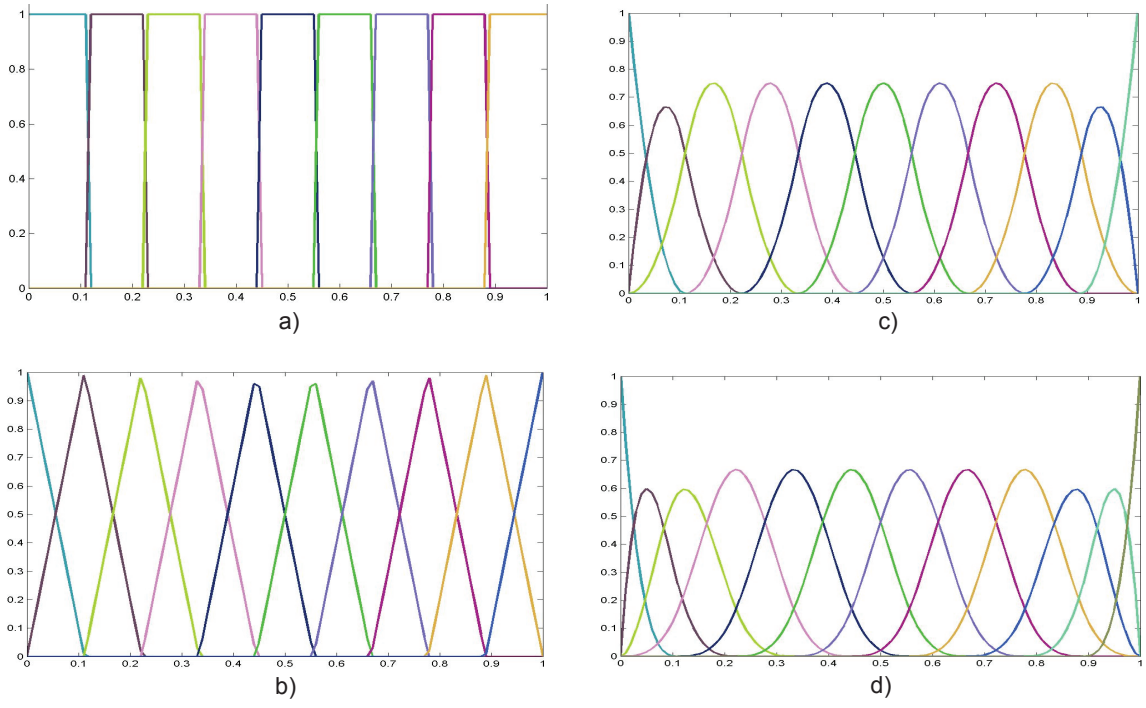

Figure 3.2: B-spline basis functions of degree 0 through 3: a) $\mathcal{B}_{j,0}^{(0)}$, b) $\mathcal{B}_{j,1}^{(0)}$, c) $\mathcal{B}_{j,2}^{(0)}$ and d) $\mathcal{B}_{j,3}^{(0)}$

Moreover, Figure 3.3 depicts a piecewise polynomial function and the basis functions that give rise to the resulting curve.

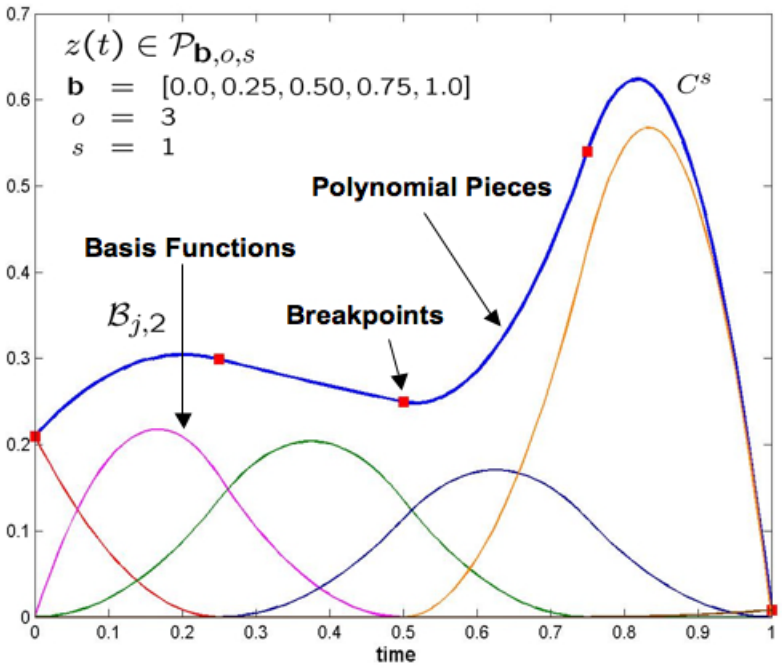

Figure 3.3: Piecewise polynomial curve expressed in terms of a linear combination of B-spline basis functions.

In addition, a B-Spline curve possesses a rich set of useful properties.

Property 1: (Local Support) $\mathcal{B}_{\ell,d}^{(r)}(t) = 0$ if t is outside the interval $[k_\ell, k_{\ell+d+1})$.

Property 2: In any given knot span $[k_\ell, k_{\ell+1})$, at most $d+1$ of the $\mathcal{B}_{j,d}^{(r)}(t)$ are nonzero, namely the functions $\mathcal{B}_{\ell-d,d}^{(r)}(t), \ldots, \mathcal{B}_{\ell,d}^{(r)}(t)$.

Property 3: (Nonnegativity) $\mathcal{B}_{j,d}^{(0)}(t) \geq 0$ for all j, d, and t.

Property 4: (Partition of Unity) For an arbitrary knot span $[k_\ell, k_{\ell+1})$, $\sum_{j=\ell-d}^{\ell} \mathcal{B}_{j,d}^{(0)}(t) = 1$ for all $t \in [k_\ell, k_{\ell+1})$.

Property 5: All derivatives of $\mathcal{B}_{j,d}^{(0)}(t)$ exist in the interior of a knot span. At a knot, $\mathcal{B}_{j,d}^{(0)}(t)$ is $d-m$ times continuously differentiable, where m is the multiplicity of the knot. Hence, increasing degree increases continuity, and increasing knot multiplicity decreases continuity.

Property 6: Except for the case $d = 0$, $\mathcal{B}_{j,d}^{(0)}(t)$ attains exactly one maximum value.

Property 7: (Strong convex hull property) The curve is contained in the convex hull of its control polygon. In fact, if $t \in [k_\ell, k_{\ell+1})$ for $\ell \in [d, d + m(N_p - 1)]$, then $c(t)$ is in the convex hull of the control points $p_{\ell-d}, \ldots, p_\ell$.

Property 8: The control polygon represents a piecewise linear approximation to the curve $c(t)$; this approximation is improved by knot insertion or degree elevation.

Property 9: (Endpoint interpolation) $c(t_0) = p_0$ and $c(t_f) = p_{N_c-1}$.

The reader is referred to de Boor [1978] and Piegle and Tiller [1997] for an expanded discussion of these and many other useful properties of B-spline basis functions, including their respective proves. In addition, one is able to find in these references efficient algorithms for the computation of B-spline basis functions.

3.2 NURBS Curves and Surfaces

In this thesis, we will use instead of B-spline basis functions, NURBS basis functions, which are themselves defined in terms of B-splines. Therefore, we will require all the B-spline definitions above for their development. As before, we are parameterizing the same vector space $\mathcal{P}_{\mathbf{b},o,s}$ whose dimension is N_c. A curve $c(t)$ is expressed in terms of NURBS basis functions as follows:

$$c(t, w_0, \ldots, w_{N_c-1}, p_0, \ldots, p_{N_c-1}) = \sum_{j=0}^{N_c-1} \mathcal{R}_{j,d}^{(0)}(t, w_0, \ldots, w_{N_c-1})\ p_j \tag{3.2.1}$$

with $t \in [t_0, t_f]$, $w_j \in (0, \infty)$, and $p_j \in (-\infty, \infty)$. In addition, $\mathcal{R}_{j,d}^{(0)}(t, w_0, \ldots, w_{N_c-1})$ is the 0th time derivative of the jth NURBS basis function of degree d, and p_j are the corresponding jth control points. The NURBS basis functions are expressed in terms of B-spline basis functions themselves as follows:

$$\mathcal{R}_{j,d}^{(0)}(t, w_0, \ldots, w_{N_c-1}) = \frac{\mathcal{B}_{j,d}^{(0)}(t)\ w_j}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t)\ w_i}, \tag{3.2.2}$$

where $w_j > 0$ is the jth weight corresponding to the jth control point. Consequently, we compute the NURBS basis functions by obtaining first the B-spline basis functions. By introducing NURBS basis functions, we have increased the number of decision parameters at our disposal: weights in addition to control points.

In order to use this curve parameterization for trajectory generation, we will be required to obtain the partial derivatives of the curve with respect to time, weights, and control points.

Proposition 3.2. Consider $c(t, \mathbf{w}, \mathbf{p})$ to be a smooth curve expressed in terms of a linear combination of NURBS basis functions. Then, its rth order partial derivatives with respect to time and their first- and second-order partial derivatives with respect to control points and weights are obtained as follows:

$$
\begin{aligned}
c^{(r)}(t,\mathbf{w},\mathbf{p}) &= \mathbf{w}^T\mathbf{G}^r\mathbf{p} - \sum_{k=1}^{r}\begin{pmatrix} r \\ k \end{pmatrix}\mathbf{E}^k\ c^{(r-k)}(t,\mathbf{w},\mathbf{p}) \\
\mathbf{D_p}\left[c^{(r)}(t,\mathbf{w},\mathbf{p})\right] &= \mathbf{w}^T\mathbf{G}^r - \sum_{k=1}^{r}\begin{pmatrix} r \\ k \end{pmatrix}\mathbf{E}^k\ \left[\mathcal{R}_d^{(r-k)}(t,\mathbf{w})\right]^T \\
\mathbf{D_{pp}}\left[c^{(r)}(t,\mathbf{w},\mathbf{p})\right] &= \mathbf{0} \\
\mathbf{D_w}\left[c^{(r)}(t,\mathbf{w},\mathbf{p})\right] &= -\left[\left(\mathbf{F}^0-\mathbf{I}\right)\mathbf{G}^r\ \mathbf{p}\right]^T \\
&\quad + \sum_{k=1}^{r}\begin{pmatrix} r \\ k \end{pmatrix}\left[\left(\mathbf{F}^0-\mathbf{I}\right)\left(\mathbf{H}^k\right)^T\right]^T c^{(r-k)}(t,\mathbf{w},\mathbf{p}) \\
&\quad - \sum_{k=1}^{r}\begin{pmatrix} r \\ k \end{pmatrix}\mathbf{E}^k\ \mathbf{D_w}\left[c^{(r-k)}(t,\mathbf{w},\mathbf{p})\right] \\
\mathbf{D_{ww}}\left[c^{(r)}(t,\mathbf{w},\mathbf{p})\right] &= \left(\mathbf{F}^0-\mathbf{I}\right)\mathbf{G}^r\ \mathbf{p}\ \mathbf{H}^0 + \left[\left(\mathbf{F}^0-\mathbf{I}\right)\mathbf{G}^r\ \mathbf{p}\ \mathbf{H}^0\right]^T \\
&\quad + \sum_{k=1}^{r}\begin{pmatrix} r \\ k \end{pmatrix}\left(\mathbf{AB}+[\mathbf{AB}]^T\right) \\
&\quad - \sum_{k=1}^{r}\begin{pmatrix} r \\ k \end{pmatrix}\mathbf{E}^k\ \mathbf{D_{ww}}\left[c^{(r-k)}(t,\mathbf{w},\mathbf{p})\right],
\end{aligned}
$$

with $t \in [t_0, t_f]$, $w_j \in (0,\infty)$, $p_j \in (-\infty,\infty)$, $\left[\mathcal{B}_d^{(r)}(t)\right] = \begin{bmatrix} \mathcal{B}_{0,d}^{(r)}(t) \\ \vdots \\ \mathcal{B}_{N_c-1,d}^{(r)}(t) \end{bmatrix}$, $\mathbf{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_{N_c-1} \end{bmatrix}$,

$\mathbf{p} = \begin{bmatrix} p_0 \\ \vdots \\ p_{N_c-1} \end{bmatrix}$, $\mathrm{diag}(\{\mathcal{B}_d^{(r)}(t)\}) = \begin{bmatrix} \mathcal{B}_{0,d}^{(r)}(t) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \mathcal{B}_{N_c-1,d}^{(r)}(t) \end{bmatrix}$, $\mathbf{H}^\ell = \frac{\left[\mathcal{B}_d^{(\ell)}(t)\right]^T}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}$,

$\mathbf{E}^\ell = \mathbf{H}^\ell\ \mathbf{w}$, $\mathbf{F}^\ell = \left(\mathbf{w}\ \mathbf{H}^\ell\right)^T$, $\mathbf{G}^\ell = \frac{\mathrm{diag}(\{\mathcal{B}_d^{(\ell)}(t)\})}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}$, $\mathbf{A} = \mathbf{E}^k\ \left[\mathcal{B}_d^{(0)}(t)\right] - \left[\mathcal{B}_d^{(k)}(t)\right]$ and

$\mathbf{B} = \frac{\mathbf{D_w}\left[c^{(r-k)}(t,\mathbf{w},\mathbf{p})\right]}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}} - \mathbf{H}^\ell\ \frac{c^{(r-k)}(t,\mathbf{w},\mathbf{p})}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}$.

A detail derivation of Proposition 3.2 can be found in Appendix A.

A NURBS surface of degree p in the x direction and degree q in the y direction is a bivariate vector-valued piecewise rational function of the form

$$\mathcal{S}(x,y) = \frac{\sum_{i=0}^{N_c^x-1}\sum_{j=0}^{N_c^y-1} \mathcal{B}_{i,p}(x)\ \mathcal{B}_{j,q}(y)\ w_{i,j}\ p_{i,j}}{\sum_{i=0}^{N_c^x-1}\sum_{j=0}^{N_c^y-1} \mathcal{B}_{i,p}(x)\ \mathcal{B}_{j,q}(y)\ w_{i,j}}, \qquad x \in [x_i, x_f],\ y \in [y_i, y_f]. \tag{3.2.3}$$

The $\{p_{i,j}\}$ for a bidirectional control net, the $\{w_{i,j}\}$ are the weights, and the $\mathcal{B}_{i,p}(x)$ and $\mathcal{B}_{j,q}(y)$ are the nonrational B-Spline basis functions defined on the knotpoints vectors

$$\mathbf{k}^x = \left(b_0^1, \ldots, b_0^{p+1}, b_1^1, \ldots, b_1^{m_1^x}, \ldots, b_{N_p-1}^1, \ldots, b_{N_p-1}^{m_{N_p-1}^x}, b_{N_p}^1, \ldots, b_{N_p}^{p+1}\right) \tag{3.2.4}$$

$$\mathbf{k}^y = \left(d_0^1, \ldots, d_0^{q+1}, d_1^1, \ldots, d_1^{m_1^y}, \ldots, d_{N_p-1}^1, \ldots, d_{N_p-1}^{m_{N_p-1}^y}, d_{N_p}^1, \ldots, d_{N_p}^{q+1}\right) \tag{3.2.5}$$

$\mathbf{k}^x$ has N_k^x knotpoints and $\mathbf{k}^y$ has N_k^y knotpoints and $N_c^x = N_k^x - (p+1)$ and $N_c^y = N_k^y - (q+1)$.

NURBS curves have many useful properties, which we will exploit.

Property 1: (Nonnegativity) $\mathcal{R}_{j,d}^{(0)}(t) \geq 0$ for all j, d, and $t \in [t_0, t_f]$;

Property 2: (Partition of Unity) $\sum_{j=0}^{N_c-1} \mathcal{R}_{j,d}^{(0)}(t) = 1$ for all $t \in [t_0, t_f]$;

Property 3: $\mathcal{R}_{0,d}^{(0)}(t_0) = \mathcal{R}_{N_c-1,d}^{(0)}(t_f) = 1$;

Property 4: For $d > 0$, all $\mathcal{R}_{j,d}^{(0)}(t)$ attain exactly one maximum on the interval $t \in [t_0, t_f]$;

Property 5: (Local Support) $\mathcal{R}_{\ell,d}^{(r)}(t) = 0$ for $t \notin [k_\ell, k_{\ell+d+1})$. Furthermore, in any given knot span, at most $d+1$ of the $\mathcal{R}_{j,d}^{(r)}(t)$ are nonzero. In general, at most $\mathcal{R}_{\ell-d,d}^{(r)}(t), \ldots, \mathcal{R}_{\ell,d}^{(r)}(t)$ may be nonzero in $[k_\ell, k_{\ell+1})$.

Property 6: All derivatives of $\mathcal{R}_{j,d}^{(0)}(t)$ exist in the interior of a knot span, where it is a rational function with nonzero denominator. At a knot, $\mathcal{R}_{j,d}^{(0)}(t)$ is $d - m$ times continuously differentiable, where m is the multiplicity of the knot.

Property 7: If $w_j = 1$ for all j, the $\mathcal{R}_{j,d}^{(r)}(t) = \mathcal{B}_{j,d}^{(r)}(t)$ for all j.

Property 8: (Strong convex hull property) The curve is contained in the convex hull of its control polygon. In fact, if $t \in [k_\ell, k_{\ell+1})$ for $\ell \in [d, d + m(N_p - 1)]$, then $c(t)$ is in the convex hull of the control points $p_{\ell-d}, \ldots, p_\ell$.

Property 9: (Endpoint interpolation) $c(t_0) = p_0$ and $c(t_f) = p_{N_c-1}$.

Chapter 4

Transcription of Optimal Control Problems to Nonlinear Programming Problems

In this chapter, we will describe how to compute minimizers to optimal control problems expressed in the following general form:

$$\min_{\mathbf{x}(t),\mathbf{u}(t)} \quad \mathcal{F}_0(\mathbf{x}(t_0),\mathbf{u}(t_0)) + \int_{t_0}^{t_f} \mathcal{F}_t(\mathbf{x}(t),\mathbf{u}(t))\, dt + \mathcal{F}_f(\mathbf{x}(t_f),\mathbf{u}(t_f)) \tag{4.0.1}$$

subject to

$$\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t),\mathbf{u}(t)), \qquad t \in [t_0,t_f] \tag{4.0.2}$$

$$\begin{aligned}
\boldsymbol{\ell}_0 \leq \quad & \mathbf{A}_0\mathbf{x}(t_0) + \mathbf{B}_0\mathbf{u}(t_0) && \leq \mathbf{u}_0 \\
\boldsymbol{\ell}_t \leq \quad & \mathbf{A}_t\mathbf{x}(t) + \mathbf{B}_t\mathbf{u}(t) && \leq \mathbf{u}_t, \qquad t \in [t_0,t_f] \\
\boldsymbol{\ell}_f \leq \quad & \mathbf{A}_f\mathbf{x}(t_f) + \mathbf{B}_f\mathbf{u}(t_f) && \leq \mathbf{u}_f \\
\mathbf{L}_0 \leq \quad & \mathbf{c}_0(\mathbf{x}(t_0),\mathbf{u}(t_0)) && \leq \mathbf{U}_0 \\
\mathbf{L}_t \leq \quad & \mathbf{c}_t(\mathbf{x}(t),\mathbf{u}(t)) && \leq \mathbf{U}_t, \qquad t \in [t_0,t_f] \\
\mathbf{L}_f \leq \quad & \mathbf{c}_f(\mathbf{x}(t_f),\mathbf{u}(t_f)) && \leq \mathbf{U}_f
\end{aligned} \tag{4.0.3}$$

by a direct method (i.e., by transcription to a nonlinear programming problem). In particular, let the state and input evolutions be described by the mappings $\mathbf{x} : [t_0,t_f] \to \mathcal{X} \subset \mathbb{R}^n$ and $\mathbf{u} : [t_0,t_f] \to \mathcal{U} \subset \mathbb{R}^m$. In addition, assume the cost functional (4.0.1), dynamic constraints (4.0.2), and trajectory and actuator constraints (4.0.3) to be sufficiently smooth. In addition, the cost functional is expressed as a sum of three terms (i.e., initial, trajectory, and final). Each function $\mathcal{F}_\ell$, $\ell \in \{0,t,f\}$ in the cost functional is a scalar-valued function $\mathcal{F}_\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$. The linear and

nonlinear constraints are also divided into three terms (i.e., initial, trajectory, and final) and they are vector-valued functions $\mathcal{L}_\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{N_\ell^l}$, $c_\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{N_\ell^n}$, $\ell \in \{0, t, f\}$. Each of these sets of constraints is allowed to be set to equality, one-sided bounded, two-sided bounded, or unbounded by manipulating the values of the lower and upper bounds.

The transcription of an optimal control problem to a nonlinear programming problem is accomplished by first parameterizing the states and inputs (flat outputs in the case of differentially flat systems) in terms of piecewise polynomial functions followed by a balanced discretization in the t parameter. In particular, we will parameterize the states and inputs (flat outputs) in terms of piecewise polynomial functions using a linear combination of NURBS basis functions. In general, we express a curve and its derivatives as follows:

$$z^{(r)}(t, w_0, \ldots, w_{N_c-1}, p_0, \ldots, p_{N_c-1}) = \sum_{j=0}^{N_c-1} \mathcal{R}_{j,d}^{(r)}(t, w_0, \ldots, w_{N_c-1})\ p_j \tag{4.0.4}$$

with $t \in [t_0, t_f]$, $w_j \in (0, \infty)$, and $p_j \in (-\infty, \infty)$. In particular, we will collect all the states and inputs (or flat outputs) and their derivatives stacked as follows: $\tilde{\mathbf{z}} = (\tilde{\mathbf{z}}_1, \ldots, \tilde{\mathbf{z}}_{N_o})$, where $\tilde{\mathbf{z}}_i = (z_i^{(0)}, \ldots, z_i^{(D_i)})$, $i = 1, \ldots, N_o$. D_i is the maximum required derivative of the ith z-variable, and N_o is the total number of z-variables which, depending on the application, can be the sum of the states and inputs (flat outputs). Consider the evaluation of the NURBS curve z_i and its derivatives at $\tau \in [t_0, t_f]$:

$$\begin{aligned}
z_i^{(0)}(\tau, \tilde{\mathbf{w}}_i, \tilde{\mathbf{p}}_i) &= \sum_{j=0}^{N_i^c-1} \mathcal{R}_{j,d_i}^{(0)}(\tau, \tilde{\mathbf{w}}_i)\ p_j^i \\
z_i^{(1)}(\tau, \tilde{\mathbf{w}}_i, \tilde{\mathbf{p}}_i) &= \sum_{j=0}^{N_i^c-1} \mathcal{R}_{j,d_i}^{(1)}(\tau, \tilde{\mathbf{w}}_i)\ p_j^i \\
&\vdots \\
z_i^{(D_i)}(\tau, \tilde{\mathbf{w}}_i, \tilde{\mathbf{p}}_i) &= \sum_{j=0}^{N_i^c-1} \mathcal{R}_{j,d_i}^{(D_i)}(\tau, \tilde{\mathbf{w}}_i)\ p_j^i.
\end{aligned}$$

In matrix form:

$$\underbrace{\begin{bmatrix} z_i^{(0)}(\tau, \tilde{\mathbf{w}}_i, \tilde{\mathbf{p}}_i) \\ z_i^{(1)}(\tau, \tilde{\mathbf{w}}_i, \tilde{\mathbf{p}}_i) \\ \vdots \\ z_i^{(D_i)}(\tau, \tilde{\mathbf{w}}_i, \tilde{\mathbf{p}}_i) \end{bmatrix}}_{\tilde{\mathbf{z}}_i(\tau, \tilde{\mathbf{w}}_i, \tilde{\mathbf{p}}_i)} = \underbrace{\begin{bmatrix} \mathcal{R}_{0,d_i}^{(0)}(\tau, \tilde{\mathbf{w}}_i) & \mathcal{R}_{1,d_i}^{(0)}(\tau, \tilde{\mathbf{w}}_i) & \ldots & \mathcal{R}_{N_i^c-1,d_i}^{(0)}(\tau, \tilde{\mathbf{w}}_i) \\ \mathcal{R}_{0,d_i}^{(1)}(\tau, \tilde{\mathbf{w}}_i) & \mathcal{R}_{1,d_i}^{(1)}(\tau, \tilde{\mathbf{w}}_i) & \ldots & \mathcal{R}_{N_i^c-1,d_i}^{(1)}(\tau, \tilde{\mathbf{w}}_i) \\ \vdots & \vdots & & \vdots \\ \mathcal{R}_{0,d_i}^{(D_i)}(\tau, \tilde{\mathbf{w}}_i) & \mathcal{R}_{1,d_i}^{(D_i)}(\tau, \tilde{\mathbf{w}}_i) & \ldots & \mathcal{R}_{N_i^c-1,d_i}^{(D_i)}(\tau, \tilde{\mathbf{w}}_i) \end{bmatrix}}_{\tilde{\mathcal{R}}_{d_i}(\tau, \tilde{\mathbf{w}}_i)} \underbrace{\begin{bmatrix} p_0^i \\ p_1^i \\ \vdots \\ p_{N_i^c-1}^i \end{bmatrix}}_{\tilde{\mathbf{p}}_i}, \tag{4.0.5}$$

or simply $\tilde{\mathbf{z}}_i(\tau, \tilde{\mathbf{w}}_i, \tilde{\mathbf{p}}_i) = \tilde{\mathcal{R}}_{d_i}(\tau, \tilde{\mathbf{w}}_i)\ \tilde{\mathbf{p}}_i$, where $\tilde{\mathbf{z}}_i \in \mathbb{R}^{D_i+1}$, $\tilde{\mathcal{R}}_{d_i}(\tau, \tilde{\mathbf{w}}_i) \in \mathbb{R}^{(D_i+1)\times N_i^c}$, $\tilde{\mathbf{w}}_i \in \mathbb{R}^{N_i^c}$, and $\tilde{\mathbf{p}}_i \in \mathbb{R}^{N_i^c}$. More generally, we will consider all the z-variables and their derivatives arranged in the following manner:

$$\underbrace{\begin{bmatrix} \tilde{\boldsymbol{z}}_1(\tau, \tilde{\mathbf{w}}, \tilde{\mathbf{p}}) \\ \tilde{\boldsymbol{z}}_2(\tau, \tilde{\mathbf{w}}, \tilde{\mathbf{p}}) \\ \vdots \\ \tilde{\boldsymbol{z}}_{N_o}(\tau, \tilde{\mathbf{w}}, \tilde{\mathbf{p}}) \end{bmatrix}}_{\tilde{\mathbf{z}}(\tau, \tilde{\mathbf{w}}, \tilde{\mathbf{p}})} = \underbrace{\begin{bmatrix} \tilde{\mathcal{R}}_{d_1}(\tau, \tilde{\mathbf{w}}_1) & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \tilde{\mathcal{R}}_{d_2}(\tau, \tilde{\mathbf{w}}_2) & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \tilde{\mathcal{R}}_{d_{N_o}}(\tau, \tilde{\mathbf{w}}_{N_o}) \end{bmatrix}}_{\tilde{\mathcal{R}}(\tau, \tilde{\mathbf{w}})} \underbrace{\begin{bmatrix} \tilde{\mathbf{p}}_1 \\ \tilde{\mathbf{p}}_2 \\ \vdots \\ \tilde{\mathbf{p}}_{N_o} \end{bmatrix}}_{\tilde{\mathbf{p}}} \tag{4.0.6}$$

or simply $\tilde{\mathbf{z}}(\tau, \tilde{\mathbf{w}}, \tilde{\mathbf{p}}) = \tilde{\mathcal{R}}(\tau, \tilde{\mathbf{w}})\ \tilde{\mathbf{p}}$, where $\tilde{\mathbf{z}} \in \mathbb{R}^{N_T^v}$, $\tilde{\mathcal{R}}(\tau, \tilde{\mathbf{w}}) \in \mathbb{R}^{N_T^v \times N_T^c}$, $\tilde{\mathbf{p}} \in \mathbb{R}^{N_T^c}$. We compute the total number of z-variables and decision variables as follows:

$$N_T^v = \sum_{i=1}^{N_o} N_i^v = \sum_{i=1}^{N_o} (D_i + 1) \qquad (z\text{-variables}) \tag{4.0.7}$$

$$N_T^c = \sum_{i=1}^{N_o} N_p^i(o_i - (s_i + 1)) + (s_i + 1) \qquad (\text{decision variables}). \tag{4.0.8}$$

There are many types of problems that are possible to setup using the NURBS curve parameterization. In particular, we can fix all the weights and solve for all the control points or vice-versa. It is also possible to fix some of the weights and some of the control points and solve for the remaining parameters. In any event, we compute the number of total *active decision variables* as follows:

$$\mathcal{N}_T^c = \sum_{i=1}^{N_o} \delta_i^w \left[N_p^i(o_i - (s_i + 1)) + (s_i + 1) \right] + \sum_{i=1}^{N_o} \delta_i^p \left[N_p^i(o_i - (s_i + 1)) + (s_i + 1) \right], \tag{4.0.9}$$

where $\delta_i^w \in \{0, 1\}$, $\delta_i^p \in \{0, 1\}$, 0 if the parameter is fixed, and 1 otheriwse. In addition, we will construct the active decision variables vector as follows:

$$\mathbf{y} = \begin{bmatrix} \tilde{\mathbf{w}} \\ \tilde{\mathbf{p}} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{w}}_1 \\ \vdots \\ \tilde{\mathbf{w}}_{N_o} \\ \tilde{\mathbf{p}}_1 \\ \vdots \\ \tilde{\mathbf{p}}_{N_o} \end{bmatrix} \in \mathbb{R}^{\mathcal{N}_T^c}. \tag{4.0.10}$$

At this point, we proceed by substituting the states, inputs, and their derivatives (flat outputs and their derivatives) in the optimal control problem with the NURBS curve parameterization described

above. This leads to the following general form:

$$\min_{\tilde{\mathbf{z}}} J[\tilde{\mathbf{z}}] \quad = \quad \mathcal{G}_0(\tilde{\mathbf{z}}(t_0)) + \int_{t_0}^{t_f} \mathcal{G}_t(\tilde{\mathbf{z}}(s))\ ds + \ \mathcal{G}_f(\tilde{\mathbf{z}}(t_f))$$

subject to

$$\begin{aligned}
& \mathcal{D}(\tilde{\mathbf{z}}(t)) = \mathbf{0}, && t \in [t_0, t_f] \\
\boldsymbol{\ell}_0 \leq \quad & \mathcal{A}_0\ \tilde{\mathbf{z}}(t_0) && \leq \mathbf{u}_0 \\
\boldsymbol{\ell}_t \leq \quad & \mathcal{A}_t\ \tilde{\mathbf{z}}(t) && \leq \mathbf{u}_t, \qquad t \in [t_0, t_f] \\
\boldsymbol{\ell}_f \leq \quad & \mathcal{A}_f\ \tilde{\mathbf{z}}(t_f) && \leq \mathbf{u}_f \\
\mathbf{L}_0 \leq \quad & \mathcal{C}_0(\tilde{\mathbf{z}}(t_0)) && \leq \mathbf{U}_0 \\
\mathbf{L}_t \leq \quad & \mathcal{C}_t(\tilde{\mathbf{z}}(t)) && \leq \mathbf{U}_t, \qquad t \in [t_0, t_f] \\
\mathbf{L}_f \leq \quad & \mathcal{C}_f(\tilde{\mathbf{z}}(t_f)) && \leq \mathbf{U}_f.
\end{aligned}$$

We have purposely only shown the t parameter dependence of the NURBS curves to make the discretization of the optimal control problem more readable. The goal of the discretization is to make the problem finite-dimensional. Moreover, our intent is to make this discretization a specification that the user has to make ahead of time. The advantage of this decision will become clearer in the next section, but, suffice it to say, this choice will allow us to perform lengthy computations outside the convergence loop, reducing the total computational time of the algorithm. Consider the following uniform time partition (collocation points): $t_0 = \tau_0 < \tau_1 < \ldots \tau_{N_\tau - 2} < \tau_{N_\tau - 1} = t_f$ where $\tau_i = t_0 + i\Delta\tau,\ i = 0, \ldots, N_\tau - 1,\ \Delta\tau = \frac{t_f - t_0}{N_\tau - 1}$. N_τ is the total number of collocation points. In addition, the integral must be approximated by a quadrature formula. We will define for this purpose the nodes $r_j = \tau_i + j\ \Delta r_i$ for $j = 0, 1, \ldots, N$, and $\Delta r_i = \frac{\tau_{i+1} - \tau_i}{N}$, where N is the number of points required by one of the following quadrature formulas:

$$\begin{aligned}
(N = 1) \quad & \int_{\tau_i}^{\tau_{i+1}} \mathcal{G}_t(\tilde{\mathbf{z}}(t))\ dt \quad \approx \quad \frac{\Delta r_i}{2}\Big\{\mathcal{G}_t(\tilde{\mathbf{z}}(r_0)) + \mathcal{G}_t(\tilde{\mathbf{z}}(r_1))\Big\} \\
(N = 2) \quad & \int_{\tau_i}^{\tau_{i+1}} \mathcal{G}_t(\tilde{\mathbf{z}}(t))\ dt \quad \approx \quad \frac{\Delta r_i}{3}\Big\{\mathcal{G}_t(\tilde{\mathbf{z}}(r_0)) + 4\ \mathcal{G}_t(\tilde{\mathbf{z}}(r_1)) + \mathcal{G}_t(\tilde{\mathbf{z}}(r_2))\Big\} \\
(N = 3) \quad & \int_{\tau_i}^{\tau_{i+1}} \mathcal{G}_t(\tilde{\mathbf{z}}(t))\ dt \quad \approx \quad \frac{3\Delta r_i}{8}\Big\{\mathcal{G}_t(\tilde{\mathbf{z}}(r_0)) + 3\ \mathcal{G}_t(\tilde{\mathbf{z}}(r_1)) + 3\ \mathcal{G}_t(\tilde{\mathbf{z}}(r_2)) + \mathcal{G}_t(\tilde{\mathbf{z}}(r_3))\Big\}
\end{aligned}$$

$$
\begin{aligned}
(N=4)\quad \int_{\tau_i}^{\tau_{i+1}} \mathcal{G}_t(\tilde{\mathbf{z}}(t))\,dt &\approx \frac{2\Delta r_i}{45}\Big\{7\,\mathcal{G}_t(\tilde{\mathbf{z}}(r_0)) + 32\,\mathcal{G}_t(\tilde{\mathbf{z}}(r_1)) + 12\,\mathcal{G}_t(\tilde{\mathbf{z}}(r_2)) + 32\,\mathcal{G}_t(\tilde{\mathbf{z}}(r_3)) \\
&\quad +7\,\mathcal{G}_t(\tilde{\mathbf{z}}(r_4))\Big\} \\
(N=5)\quad \int_{\tau_i}^{\tau_{i+1}} \mathcal{G}_t(\tilde{\mathbf{z}}(t))\,dt &\approx \frac{5\Delta r_i}{288}\Big\{19\,\mathcal{G}_t(\tilde{\mathbf{z}}(r_0)) + 75\,\mathcal{G}_t(\tilde{\mathbf{z}}(r_1)) + 50\,\mathcal{G}_t(\tilde{\mathbf{z}}(r_2)) + 50\,\mathcal{G}_t(\tilde{\mathbf{z}}(r_3)) \\
&\quad +75\,\mathcal{G}_t(\tilde{\mathbf{z}}(r_4)) + 19\,\mathcal{G}_t(\tilde{\mathbf{z}}(r_5))\Big\} \\
(N=6)\quad \int_{\tau_i}^{\tau_{i+1}} \mathcal{G}_t(\tilde{\mathbf{z}}(t))\,dt &\approx \frac{\Delta r_i}{140}\Big\{41\,\mathcal{G}_t(\tilde{\mathbf{z}}(r_0)) + 216\,\mathcal{G}_t(\tilde{\mathbf{z}}(r_1)) + 27\,\mathcal{G}_t(\tilde{\mathbf{z}}(r_2)) + 272\,\mathcal{G}_t(\tilde{\mathbf{z}}(r_3)) \\
&\quad +27\,\mathcal{G}_t(\tilde{\mathbf{z}}(r_4)) + 216\,\mathcal{G}_t(\tilde{\mathbf{z}}(r_5)) + 41\,\mathcal{G}_t(\tilde{\mathbf{z}}(r_6))\Big\}.
\end{aligned}
$$

These quadrature formulas have specific names depending on the value of N: a) trapezoidal rule ($N = 1$), b) Simpson's rule ($N = 2$), c) Simpson's three-eighths rule ($N = 3$), d) Milne's rule ($N = 4$) and e) Weddle's rule ($N = 6$). In general, all of these approximations have the following form:

$$
\int_{\tau_i}^{\tau_{i+1}} \mathcal{G}_t(\tilde{\mathbf{z}}(t))\,dt \approx \sum_{j=0}^{N} \gamma_j^i\,\mathcal{G}_t(\tilde{\mathbf{z}}(r_j)), \qquad \gamma_j^i = \Delta r_i \alpha_j
$$

The optimal control problem then becomes

$$
\min_{\tilde{\mathbf{z}}} J[\tilde{\mathbf{z}}] = \mathcal{G}_0(\tilde{\mathbf{z}}(\tau_0, \tilde{\mathbf{w}}, \tilde{\mathbf{p}})) + \sum_{i=0}^{N_\tau - 2} \left[\sum_{j=0}^{N} \gamma_j^i\,\mathcal{G}_t(\tilde{\mathbf{z}}(r_j, \tilde{\mathbf{w}}, \tilde{\mathbf{p}})) \right] + \mathcal{G}_f(\tilde{\mathbf{z}}(\tau_{N_\tau - 1}, \tilde{\mathbf{w}}, \tilde{\mathbf{p}}))
$$

subject to

$$
\begin{aligned}
& \mathcal{D}(\tilde{\mathbf{z}}(\tau_i, \tilde{\mathbf{w}}, \tilde{\mathbf{p}})) = \mathbf{0}, && i = 0, \ldots, N_\tau - 1 \\
\boldsymbol{\ell}_0 \le\ & \mathcal{A}_0\,\tilde{\mathbf{z}}(\tau_0, \tilde{\mathbf{w}}, \tilde{\mathbf{p}}) && \le \mathbf{u}_0 \\
\boldsymbol{\ell}_t \le\ & \mathcal{A}_t\,\tilde{\mathbf{z}}(\tau_i, \tilde{\mathbf{w}}, \tilde{\mathbf{p}}) && \le \mathbf{u}_t, \qquad i = 0, \ldots, N_\tau - 1 \\
\boldsymbol{\ell}_f \le\ & \mathcal{A}_f\,\tilde{\mathbf{z}}(\tau_{N_\tau - 1}, \tilde{\mathbf{w}}, \tilde{\mathbf{p}}) && \le \mathbf{u}_f \\
\mathbf{L}_0 \le\ & \mathcal{C}_0(\tilde{\mathbf{z}}(\tau_0, \tilde{\mathbf{w}}, \tilde{\mathbf{p}})) && \le \mathbf{U}_0 \\
\mathbf{L}_t \le\ & \mathcal{C}_t(\tilde{\mathbf{z}}(\tau_i, \tilde{\mathbf{w}}, \tilde{\mathbf{p}})) && \le \mathbf{U}_t, \qquad i = 0, \ldots, N_\tau - 1 \\
\mathbf{L}_f \le\ & \mathcal{C}_f(\tilde{\mathbf{z}}(\tau_{N_\tau - 1}, \tilde{\mathbf{w}}, \tilde{\mathbf{p}})) && \le \mathbf{U}_f.
\end{aligned}
$$

The result of the previous parameterization and discretization is a nonlinear programming problem where the unknowns are the active weights and control points of all the NURBS curves. The

structure of a general nonlinear programming problem is as follows:

$$\begin{aligned} &\min_{\mathbf{y}} f(\mathbf{y}) \\ &\text{subject to} \\ &\mathbf{L} \leq \begin{bmatrix} \mathbf{y} \\ \mathbf{A}\mathbf{y} \\ \mathbf{c}(\mathbf{y}) \end{bmatrix} \leq \mathbf{U}. \end{aligned} \tag{4.0.11}$$

For our setting, $\mathbf{y} \in \mathbb{R}^{\mathcal{N}_T^c}$, $\mathcal{N}_T^c$ is the total number of active decision variables, $f : \mathbb{R}^{\mathcal{N}_T^c} \rightarrow \mathbb{R}$ is a smooth real-valued function, $\mathbf{A} : \mathbb{R}^{\mathcal{N}_T^c} \rightarrow \mathbb{R}^{N_T^l = N_i^l + N_\tau N_t^l + N_f^l}$ is a linear operator, $\mathbf{c} : \mathbb{R}^{\mathcal{N}_T^c} \rightarrow \mathbb{R}^{N_T^n = N_i^n + N_\tau N_t^n + N_f^n}$ is a smooth vector-valued function and the lower bounds and upper bounds are vectors $\mathbf{L}, \mathbf{U} \in \mathbb{R}^{\mathcal{N}_T^c + N_T^l + N_T^n}$, where N_T^l and N_T^n are the total number of linear constraints and nonlinear constraints.

In order to solve for the active decision variables, we will make use of a Sequential Quadratic Programming (SQP) solver. An SQP method obtains search directions from a sequence of Quadratic Programming (QP) subproblems. Each QP subproblem minimizes a quadratic model of a certain Lagrangian function subject to linearized constraints. In particular, the structure of the QP subproblem for the SNOPT solver, Gill et al. [2005], is of the following form:

$$\begin{aligned} &\min_{\mathbf{y}} \mathcal{L}_q(\mathbf{y}_k, \mathbf{y}_k, \lambda_k) \\ &\text{subject to} \\ &\mathbf{L} \leq \begin{bmatrix} \mathbf{y} \\ \mathbf{A}\mathbf{y} \\ \mathbf{c}(\mathbf{y}_k) + \mathbf{D}_y \mathbf{c}(\mathbf{y}_k)(\mathbf{y} - \mathbf{y}_k) \end{bmatrix} \leq \mathbf{U}. \end{aligned}$$

The modified cost function $\mathcal{L}_q(\mathbf{y}, \mathbf{y}_k, \lambda_k)$ is defined as follows:

$$\mathcal{L}_q(\mathbf{y}, \mathbf{y}_k, \boldsymbol{\lambda}_k) = \mathcal{L}(\mathbf{y}_k, \mathbf{y}_k, \boldsymbol{\lambda}_k) + \mathbf{D}_\mathbf{y}\mathcal{L}(\mathbf{y}_k, \mathbf{y}_k, \boldsymbol{\lambda}_k)(\mathbf{y} - \mathbf{y}_k) + \frac{1}{2}(\mathbf{y} - \mathbf{y}_k)^T \mathbf{D}_{\mathbf{yy}}\mathcal{L}(\mathbf{y}_k, \mathbf{y}_k, \boldsymbol{\lambda}_k)(\mathbf{y} - \mathbf{y}_k)$$

This is a quadratic approximation of the modified Lagrangian $\mathcal{L}(\mathbf{y}, \mathbf{y}_k, \boldsymbol{\lambda}_k)$:

$$\mathcal{L}(\mathbf{y}, \mathbf{y}_k, \boldsymbol{\lambda}_k) = f(\mathbf{y}) - \boldsymbol{\lambda}_k^T \left[\mathbf{c}(\mathbf{y}) - \mathbf{c}(\mathbf{y}_k) - \mathbf{D}_\mathbf{y}\mathbf{c}(\mathbf{y}_k)(\mathbf{y} - \mathbf{y}_k)\right],$$

whose first- and second-order partial derivatives are computed to

$$\begin{aligned}
\mathbf{D}_{\mathbf{y}}\mathcal{L}(\mathbf{y},\mathbf{y}_k,\boldsymbol{\lambda}_k) &= \mathbf{D}_{\mathbf{y}}f(\mathbf{y}) - \boldsymbol{\lambda}_k^T\left[\mathbf{D}_{\mathbf{y}}\mathbf{c}(\mathbf{y}) - \mathbf{D}_{\mathbf{y}}\mathbf{c}(\mathbf{y}_k)\right] \\
\mathbf{D}_{\mathbf{yy}}\mathcal{L}(\mathbf{y},\mathbf{y}_k,\boldsymbol{\lambda}_k) &= \mathbf{D}_{\mathbf{yy}}f(\mathbf{y}) - \sum_i (\lambda_k)_i \, \mathbf{D}_{\mathbf{yy}}c_i(\mathbf{y}).
\end{aligned}$$

In particular, at $\mathbf{y} = \mathbf{y}_k$ we have that

$$\begin{aligned}
\mathcal{L}(\mathbf{y}_k,\mathbf{y}_k,\boldsymbol{\lambda}_k) &= f(\mathbf{y}_k) \\
\mathbf{D}_{\mathbf{y}}\mathcal{L}(\mathbf{y}_k,\mathbf{y}_k,\boldsymbol{\lambda}_k) &= \mathbf{D}_{\mathbf{y}}f(\mathbf{y}_k) \\
\mathbf{D}_{\mathbf{yy}}\mathcal{L}(\mathbf{y}_k,\mathbf{y}_k,\boldsymbol{\lambda}_k) &= \mathbf{D}_{\mathbf{yy}}f(\mathbf{y}_k) - \sum_i (\lambda_k)_i \, \mathbf{D}_{\mathbf{yy}}c_i(\mathbf{y}_k).
\end{aligned}$$

The QP subproblem then becomes

$$\min_{\mathbf{y}} f(\mathbf{y}_k) + \mathbf{D}_{\mathbf{y}}f(\mathbf{y}_k)(\mathbf{y}-\mathbf{y}_k) + \tfrac{1}{2}(\mathbf{y}-\mathbf{y}_k)^T\left[\mathbf{D}_{\mathbf{yy}}f(\mathbf{y}_k) - \sum_i (\lambda_k)_i \, \mathbf{D}_{\mathbf{yy}}c_i(\mathbf{y}_k)\right](\mathbf{y}-\mathbf{y}_k)$$

subject to

$$\mathbf{L} \leq \begin{bmatrix} \mathbf{y} \\ \mathbf{A}\mathbf{y} \\ \mathbf{c}(\mathbf{y}_k) + \mathbf{D}_y\mathbf{c}(\mathbf{y}_k)(\mathbf{y}-\mathbf{y}_k) \end{bmatrix} \leq \mathbf{U}.$$

As a consequence, we will also be required to compute the Jacobians and Hessians of the objective function and constraint functions. These can be computed analytically if the problem at hand allows it. Otherwise, solvers typically approximate the Jacobians and Hessians of all functions in the posed problem numerically at the expense of a slower rate of convergence.

In general, we obtain the Jacobian and Hessian of a function $\mathcal{F}(\tilde{\mathbf{z}}(\mathbf{y}))$ with respect to $\mathbf{y}$ using the chain rule:

$$\begin{aligned}
\mathbf{D}_{\mathbf{y}}\mathcal{F}(\tilde{\mathbf{z}}(\mathbf{y})) &= \mathbf{D}_{\tilde{\mathbf{z}}}\mathcal{F}(\tilde{\mathbf{z}}(\mathbf{y}))\ \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\mathbf{y}) \\
\mathbf{D}_{\mathbf{yy}}\mathcal{F}(\tilde{\mathbf{z}}(\mathbf{y})) &= \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\mathbf{y})^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\mathcal{F}(\tilde{\mathbf{z}}(\mathbf{y}))\ \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\mathbf{y}) + \sum_{i=1}^{N_T^v} \frac{\partial \mathcal{F}(\tilde{\mathbf{z}}(\mathbf{y}))}{\partial \hat{z}_i}\ \mathbf{D}_{\mathbf{yy}}\hat{z}_i,
\end{aligned}$$

where $\hat{z}_i$ is the ith element of the vector $\tilde{\mathbf{z}} = \left(z_1^{(0)}, \ldots, z_1^{(D_1)}, \ldots, z_{N_o}^{(0)}, \ldots, z_{N_o}^{(D_{N_o})}\right) \in \mathcal{N}_T^v$.

Chapter 5

Solving Optimal Control Problems with Control Points as Decision Variables

In this chapter, we will describe in detail the structure of the nonlinear programming problem used to obtain minimizers for optimal control problems that must deal with full or partial explicit dynamic constraints. We delay the treatment of differentially flat systems until Chapter 7. Moreover, although we could exploit the polytope structure of NURBS curves to avoid obstacles, we still need the material in Chapter 6 to formally do so, hence we will apply this technique in Chapter 7. The inclusion of explicit dynamic constraints in the optimal control problem forces us to obtain a flow line through the initial and final system configurations in such a way that the tangent vector of the curve coincides with the vector field everywhere. This set of constraints can only be verified at a discrete number of points and, therefore, once a minimizer is obtained one should use the input to integrate the dynamics and confirm that the state evolution actually matches the one found. An alternative strategy may be employed to deal with dynamic constraints using explicit Runge Kutta methods to discretize the dynamics, Betts [1998].

The main goal at this stage is to emphasize the generality of the methodology and the ease in which dynamics and constraints can be modeled. The ease of the formulation resides partly in the fact that we use a z-variable per state and input. As a result, a problem written on paper can be mapped to a computer implementation in a straightforward manner. Of course, one trades ease of implementation for an increase in the computational effort required to solve the formulated optimal control problem due to the large number of active decision variables and constraints present in the problem. In later chapters, we will expose an implementation that strives to improve the computational time and significantly reduce the number of constraints and active decision variables present in the problem at the expense of off-line analysis and recasting of the optimal control problem through algebraic

manipulations.

For now we will exploit only the basis functions structure and make use of Property 7 of NURBS basis functions. That is, we will set all the weight parameters equal to the same value $w_0 > 0$. By doing so, we will decrease the number of decision variables (i.e. the NURBS curves will not depend on weights). Moreover, $\mathcal{R}^{(r)}_{j,d}(t, \tilde{\mathbf{w}}_0) = \mathcal{B}^{(r)}_{j,d}(t)$ for all j. In this special case, we will only allow the control points to be decision variables. The optimal control problem then takes the following form after discretization and change of basis:

$$\min_{\tilde{\mathbf{z}}} J[\tilde{\mathbf{z}}] = \mathcal{G}_0(\tilde{\mathbf{z}}(\tau_0, \tilde{\mathbf{p}})) + \sum_{i=0}^{N_\tau - 2} \left[\sum_{j=0}^{N} \gamma_j^i \ \mathcal{G}_t(\tilde{\mathbf{z}}(r_j, \tilde{\mathbf{p}})) \right] + \mathcal{G}_f(\tilde{\mathbf{z}}(\tau_{N_\tau - 1}, \tilde{\mathbf{p}}))$$

subject to

$$\begin{aligned}
& \mathcal{D}(\tilde{\mathbf{z}}(\tau_i, \tilde{\mathbf{p}})) = \mathbf{0}, && i = 0, \ldots, N_\tau - 1 \\
\boldsymbol{\ell}_0 \leq \ & \mathcal{A}_0 \ \tilde{\mathbf{z}}(\tau_0, \tilde{\mathbf{p}}) && \leq \mathbf{u}_0 \\
\boldsymbol{\ell}_t \leq \ & \mathcal{A}_t \ \tilde{\mathbf{z}}(\tau_i, \tilde{\mathbf{p}}) && \leq \mathbf{u}_t, \quad i = 0, \ldots, N_\tau - 1 \\
\boldsymbol{\ell}_f \leq \ & \mathcal{A}_f \ \tilde{\mathbf{z}}(\tau_{N_\tau - 1}, \tilde{\mathbf{p}}) && \leq \mathbf{u}_f \\
\mathbf{L}_0 \leq \ & \mathcal{C}_0(\tilde{\mathbf{z}}(\tau_0, \tilde{\mathbf{p}})) && \leq \mathbf{U}_0 \\
\mathbf{L}_t \leq \ & \mathcal{C}_t(\tilde{\mathbf{z}}(\tau_i, \tilde{\mathbf{p}})) && \leq \mathbf{U}_t, \quad i = 0, \ldots, N_\tau - 1 \\
\mathbf{L}_f \leq \ & \mathcal{C}_f(\tilde{\mathbf{z}}(\tau_{N_\tau - 1}, \tilde{\mathbf{p}})) && \leq \mathbf{U}_f
\end{aligned}$$

As previously mentioned, the nonlinear programming problem has the following general structure:

$$\min_{\mathbf{y}} f(\mathbf{y})$$

subject to

$$\mathbf{L} \leq \begin{bmatrix} \mathbf{y} \\ \mathbf{A}\mathbf{y} \\ \mathbf{c}(\mathbf{y}) \end{bmatrix} \leq \mathbf{U}$$

For the current case, the decision variables are the control points of all the NURBS curves. The objective function f, the linear operator $\mathbf{A}$ and the nonlinear constraint function $\mathbf{c}$ are constructed as follows:

$$f(\tilde{\mathbf{z}}(\mathbf{y})) = \mathcal{G}_0(\tilde{\mathbf{z}}(\tau_0, \mathbf{y})) + \sum_{i=0}^{N_\tau - 2} \left[\sum_{j=0}^{N} \gamma_j^i \ \mathcal{G}_t(\tilde{\mathbf{z}}(r_j, \mathbf{y})) \right] + \mathcal{G}_f(\tilde{\mathbf{z}}(\tau_{N_\tau - 1}, \mathbf{y})) \tag{5.0.1}$$

The matrix $\mathbf{A}$ is constructed by stacking up the linear constraints as follows:

$$\mathbf{A}\tilde{\mathbf{z}}(\mathbf{y}) = \begin{bmatrix} \mathcal{A}_0\ \tilde{\mathbf{z}}(\tau_0, \mathbf{y}) \\ \mathcal{A}_t\ \tilde{\mathbf{z}}(\tau_0, \mathbf{y}) \\ \vdots \\ \mathcal{A}_t\ \tilde{\mathbf{z}}(\tau_{N_\tau - 1}, \mathbf{y}) \\ \mathcal{A}_f\ \tilde{\mathbf{z}}(\tau_{N_\tau - 1}, \mathbf{y}) \end{bmatrix} = \begin{bmatrix} \mathcal{A}_0\ \tilde{\mathcal{R}}(\tau_0, \tilde{\mathbf{w}}_0,) \\ \mathcal{A}_t\ \tilde{\mathcal{R}}(\tau_0, \tilde{\mathbf{w}}_0) \\ \vdots \\ \mathcal{A}_t\ \tilde{\mathcal{R}}(\tau_{N_\tau - 1}, \tilde{\mathbf{w}}_0) \\ \mathcal{A}_f\ \tilde{\mathcal{R}}(\tau_{N_\tau - 1}, \tilde{\mathbf{w}}_0) \end{bmatrix} \mathbf{y} = \begin{bmatrix} \mathcal{A}_0\ \tilde{\mathcal{B}}(\tau_0) \\ \mathcal{A}_t\ \tilde{\mathcal{B}}(\tau_0) \\ \vdots \\ \mathcal{A}_t\ \tilde{\mathcal{B}}(\tau_{N_\tau - 1}) \\ \mathcal{A}_f\ \tilde{\mathcal{B}}(\tau_{N_\tau - 1}) \end{bmatrix} \mathbf{y} \tag{5.0.2}$$

Likewise, we stack up the nonlinear constraints in the following way:

$$\mathbf{c}(\tilde{\mathbf{z}}(\mathbf{y})) = \begin{bmatrix} \mathcal{C}_0(\tilde{\mathbf{z}}(\tau_0, \mathbf{y})) \\ \mathcal{C}_t(\tilde{\mathbf{z}}(\tau_0, \mathbf{y})) \\ \vdots \\ \mathcal{C}_t(\tilde{\mathbf{z}}(\tau_{N_\tau - 1}, \mathbf{y})) \\ \mathcal{C}_f(\tilde{\mathbf{z}}(\tau_{N_\tau - 1}, \mathbf{y})) \end{bmatrix} \tag{5.0.3}$$

Finally, the lower bounds an the upper bounds are stacked up in the following manner:

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_c \\ \boldsymbol{\ell}_0 \\ (\boldsymbol{\ell}_t)_0 \\ \vdots \\ (\boldsymbol{\ell}_t)_{N_\tau - 1} \\ \boldsymbol{\ell}_f \\ \mathbf{L}_0 \\ (\mathbf{L}_t)_0 \\ \vdots \\ (\mathbf{L}_t)_{N_\tau - 1} \\ \mathbf{L}_f \end{bmatrix} \qquad \mathbf{U} = \begin{bmatrix} \mathbf{U}_c \\ \mathbf{u}_0 \\ (\mathbf{u}_t)_0 \\ \vdots \\ (\mathbf{u}_t)_{N_\tau - 1} \\ \mathbf{u}_f \\ \mathbf{U}_0 \\ (\mathbf{U}_t)_0 \\ \vdots \\ (\mathbf{U}_t)_{N_\tau - 1} \\ \mathbf{U}_f \end{bmatrix} \tag{5.0.4}$$

where $\mathbf{L}_c$ and $\mathbf{U}_c$ are the lower and upper bounds for the decision variables. In the problems considered here the coefficients themselves are not bounded; accordingly, we set the values of the lower bound and upper bound for each coefficient to $-\infty$ and $+\infty$ correspondingly. Then, we proceed to compute the Jacobians and Hessians of the objective function and constraint functions. In particular, we are able to express the Jacobian of the cost function in this form:

$$\begin{aligned}\mathbf{D_y} f(\tilde{\mathbf{z}}(\mathbf{y})) &= \mathbf{D_{\tilde{z}}}\mathcal{G}_0(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_0,\mathbf{y}) + \sum_{i=0}^{N_\tau-2}\left[\sum_{j=0}^{N}\gamma_j^i\ \mathbf{D_{\tilde{z}}}\mathcal{G}_t(\tilde{\mathbf{z}}(r_j,\mathbf{y}))\ \mathbf{D_y}\tilde{\mathbf{z}}(r_j,\mathbf{y})\right] \\ &\quad + \mathbf{D_{\tilde{z}}}\mathcal{G}_f(\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y}))\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y})\end{aligned}$$

Using Proposition 3.1 and the stacking of z-variables and their derivatives as in Equation 4.0.6:

$$\begin{aligned}\mathbf{D_y} f(\tilde{\mathbf{z}}(\mathbf{y})) &= \mathbf{D_{\tilde{z}}}\mathcal{G}_0(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))\ \tilde{\mathcal{B}}(\tau_0) + \sum_{i=0}^{N_\tau-2}\left[\sum_{j=0}^{N}\gamma_j^i\ \mathbf{D_{\tilde{z}}}\mathcal{G}_t(\tilde{\mathbf{z}}(r_j,\mathbf{y}))\ \tilde{\mathcal{B}}(r_j)\right] + \\ &\quad \mathbf{D_{\tilde{z}}}\mathcal{G}_f(\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y}))\ \tilde{\mathcal{B}}(\tau_{N_\tau-1})\end{aligned}$$

In addition, the Jacobian of the linear constraints become

$$\mathbf{D_y}\left[\mathbf{A}\tilde{\mathbf{z}}(\mathbf{y})\right] = \begin{bmatrix} \mathcal{A}_0\ \tilde{\mathcal{B}}(\tau_0) \\ \mathcal{A}_t\ \tilde{\mathcal{B}}(\tau_0) \\ \vdots \\ \mathcal{A}_t\ \tilde{\mathcal{B}}(\tau_{N_\tau-1}) \\ \mathcal{A}_f\ \tilde{\mathcal{B}}(\tau_{N_\tau-1}) \end{bmatrix}$$

Likewise, the Jacobian of the nonlinear constraints become

$$\mathbf{D_y}\mathbf{c}(\tilde{\mathbf{z}}(\mathbf{y})) = \begin{bmatrix} \mathbf{D_{\tilde{z}}}\mathcal{C}_0(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_0,\mathbf{y}) \\ \mathbf{D_{\tilde{z}}}\mathcal{C}_t(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_0,\mathbf{y}) \\ \vdots \\ \mathbf{D_{\tilde{z}}}\mathcal{C}_t(\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y}))\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y}) \\ \mathbf{D_{\tilde{z}}}\mathcal{C}_f(\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y}))\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y}) \end{bmatrix} = \begin{bmatrix} \mathbf{D_{\tilde{z}}}\mathcal{C}_0(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))\ \tilde{\mathcal{B}}(\tau_0) \\ \mathbf{D_{\tilde{z}}}\mathcal{C}_t(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))\ \tilde{\mathcal{B}}(\tau_0) \\ \vdots \\ \mathbf{D_{\tilde{z}}}\mathcal{C}_t(\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y}))\ \tilde{\mathcal{B}}(\tau_{N_\tau-1}) \\ \mathbf{D_{\tilde{z}}}\mathcal{C}_f(\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y}))\ \tilde{\mathcal{B}}(\tau_{N_\tau-1}) \end{bmatrix}$$

The Hessians of all the functions computed above then become:

$$\begin{aligned}\mathbf{D_{yy}} f(\tilde{\mathbf{z}}(\mathbf{y})) &= \tilde{\mathcal{B}}(\tau_0)^T\ \mathbf{D_{\tilde{z}\tilde{z}}}\mathcal{G}_0(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))\ \tilde{\mathcal{B}}(\tau_0) + \sum_{i=0}^{N_\tau-2}\left[\sum_{j=0}^{N}\gamma_j^i\ \tilde{\mathcal{B}}(r_j)^T\ \mathbf{D_{\tilde{z}\tilde{z}}}\mathcal{G}_t(\tilde{\mathbf{z}}(r_j,\mathbf{y}))\ \tilde{\mathcal{B}}(r_j)\right] \\ &\quad + \tilde{\mathcal{B}}(\tau_{N_\tau-1})^T\ \mathbf{D_{\tilde{z}\tilde{z}}}\mathcal{G}_f(\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y}))\ \tilde{\mathcal{B}}(\tau_{N_\tau-1}) \\ \mathbf{D_{yy}}\left[\mathbf{A}\tilde{\mathbf{z}}(\mathbf{y})\right] &= \mathbf{0} \\ \mathbf{D_{yy}}\mathcal{C}_0^1(\tilde{\mathbf{z}}(\tau_0,\mathbf{y})) &= \tilde{\mathcal{B}}(\tau_0)^T\ \mathbf{D_{\tilde{z}\tilde{z}}}\mathcal{C}_0^1(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))\ \tilde{\mathcal{B}}(\tau_0) \\ &\vdots \\ \mathbf{D_{yy}}\mathcal{C}_0^{N_i^n}(\tilde{\mathbf{z}}(\tau_0,\mathbf{y})) &= \tilde{\mathcal{B}}(\tau_0)^T\ \mathbf{D_{\tilde{z}\tilde{z}}}\mathcal{C}_0^{N_i^n}(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))\ \tilde{\mathcal{B}}(\tau_0)\end{aligned}$$

$$
\begin{aligned}
\mathbf{D}_{\mathbf{yy}}\mathcal{C}_t^1(\tilde{\mathbf{z}}(\tau_0,\mathbf{y})) &= \tilde{\mathcal{B}}(\tau_0)^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\mathcal{C}_t^1(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))\ \tilde{\mathcal{B}}(\tau_0)\\
&\vdots\\
\mathbf{D}_{\mathbf{yy}}\mathcal{C}_t^{N_t^n}(\tilde{\mathbf{z}}(\tau_0,\mathbf{y})) &= \tilde{\mathcal{B}}(\tau_0)^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\mathcal{C}_t^{N_t^n}(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))\ \tilde{\mathcal{B}}(\tau_0)\\
&\vdots\\
\mathbf{D}_{\mathbf{yy}}\mathcal{C}_t^1(\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y})) &= \tilde{\mathcal{B}}(\tau_{N_\tau-1})^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\mathcal{C}_t^1(\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y}))\ \tilde{\mathcal{B}}(\tau_{N_c-1})\\
&\vdots\\
\mathbf{D}_{\mathbf{yy}}\mathcal{C}_t^{N_t^n}(\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y})) &= \tilde{\mathcal{B}}(\tau_{N_\tau-1})^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\mathcal{C}_t^{N_t^n}(\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y}))\ \tilde{\mathcal{B}}(\tau_{N_c-1})\\
\mathbf{D}_{\mathbf{yy}}\mathcal{C}_f^1(\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})) &= \tilde{\mathcal{B}}(\tau_{N_\tau-1})^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\mathcal{C}_f^1(\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y}))\ \tilde{\mathcal{B}}(\tau_{N_\tau-1})\\
&\vdots\\
\mathbf{D}_{\mathbf{yy}}\mathcal{C}_f^{N_f^n}(\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})) &= \tilde{\mathcal{B}}(\tau_{N_\tau-1})^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\mathcal{C}_f^{N_f^n}(\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y}))\ \tilde{\mathcal{B}}(\tau_{N_\tau-1})
\end{aligned}
$$

Example 5.1 (Analytical)**.** Consider the following optimal control problem:

$$\min_{\mathbf{u}} \quad \tfrac{1}{2}\int_{t_0=0}^{t_f=10} \mathbf{u}(t)^T\mathbf{R}\mathbf{u}(t)\ dt, \quad \mathbf{R}=\begin{bmatrix} r_{11} & 0\\ 0 & r_{22}\end{bmatrix} > \mathbf{0}$$

subject to:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} u_1(t)\\ u_2(t)\\ x_2(t)\ u_1(t)\end{bmatrix};\ \ \mathbf{x}(t_0)=\mathbf{x}_0=\begin{bmatrix}10\\2\\20\end{bmatrix},\ \ \mathbf{x}(t_f)=\mathbf{x}_f=\begin{bmatrix}27\\8\\35\end{bmatrix}$$

The first-order necessary conditions for a minimizer $(\mathbf{x}^*(t),\mathbf{u}^*(t),\boldsymbol{\lambda}^*(t))$ are expressed as follows:

$$
\begin{aligned}
\dot{\boldsymbol{\lambda}}^*(t) &= -\mathbf{D}_{\mathbf{x}}\mathcal{H}(\boldsymbol{x}^*(t),\boldsymbol{u}^*(t),\boldsymbol{\lambda}^*(t))^T\\
\dot{\mathbf{x}}^*(t) &= \mathbf{D}_{\boldsymbol{\lambda}}\mathcal{H}(\boldsymbol{x}^*(t),\boldsymbol{u}^*(t),\boldsymbol{\lambda}^*(t))^T,\ \ \mathbf{x}^*(t_i)=\mathbf{x}_0,\ \mathbf{x}^*(t_f)=\mathbf{x}_f\\
\mathbf{0} &= \mathbf{D}_{\mathbf{u}}\mathcal{H}(\mathbf{x}^*(t),\mathbf{u}^*(t),\boldsymbol{\lambda}^*(t))^T.
\end{aligned}
$$

The Hamiltonian is defined by $\mathcal{H}(\mathbf{x}(t),\mathbf{u}(t),\boldsymbol{\lambda}(t)) = f^0(\mathbf{x}(t),\mathbf{u}(t)) + \boldsymbol{\lambda}^T(t)\ \mathbf{F}(\boldsymbol{x}(t),\boldsymbol{u}(t))$ where $f^0(\mathbf{x}(t),\mathbf{u}(t)) = \frac{1}{2}\begin{bmatrix} u_1(t) & u_2(t)\end{bmatrix}\mathbf{R}\begin{bmatrix}u_1(t)\\u_2(t)\end{bmatrix}$ and $\mathbf{F}(\mathbf{x}(t),\mathbf{u}(t)) = \begin{bmatrix} u_1(t) & u_2(t) & x_2(t)\ u_1(t)\end{bmatrix}^T$. As a consequence, the partial derivatives of the Hamiltonian with respect to the states, adjoint

variables and inputs are computed as follows:

$$\begin{aligned}
\mathbf{D}_{\mathbf{x}}\mathcal{H}(\mathbf{x}(t),\mathbf{u}(t),\boldsymbol{\lambda}(t)) &= \begin{bmatrix} 0 & \lambda_3(t)\ u_1(t) & 0 \end{bmatrix} \\
\mathbf{D}_{\boldsymbol{\lambda}}\mathcal{H}(\mathbf{x}(t),\mathbf{u}(t),\boldsymbol{\lambda}(t)) &= \begin{bmatrix} u_1(t) & u_2(t) & x_2(t)u_1(t) \end{bmatrix} \\
\mathbf{D}_{\mathbf{u}}\mathcal{H}(\mathbf{x}(t),\mathbf{u}(t),\boldsymbol{\lambda}(t)) &= \begin{bmatrix} r_{11}\ u_1(t) + \lambda_1(t) + \lambda_3(t)\ x_2(t), r_{22}\ u_2(t) + \lambda_2(t) \end{bmatrix}.
\end{aligned}$$

Then first-order necessary conditions become

$$-\dot{\boldsymbol{\lambda}}^*(t) = \begin{bmatrix} 0 \\ \lambda_3^*(t)\ u_1^*(t) \\ 0 \end{bmatrix} \tag{5.0.5}$$

$$\dot{\mathbf{x}}^*(t) = \begin{bmatrix} u_1^*(t) \\ u_2^*(t) \\ x_2^*(t)u_1^*(t) \end{bmatrix};\quad \mathbf{x}^*(t_0) = \mathbf{x}_0,\quad \mathbf{x}^*(t_f) = \mathbf{x}_f \tag{5.0.6}$$

$$\mathbf{0} = \begin{bmatrix} r_{11}\ u_1^*(t) + \lambda_1^*(t) + \lambda_3^*(t)\ x_2^*(t) \\ r_{22}\ u_2^*(t) + \lambda_2^*(t) \end{bmatrix}. \tag{5.0.7}$$

In particular, we are able to solve analytically for the inputs using (5.0.7)

$$\begin{aligned}
u_1^* &= -\frac{1}{r_{11}}\left(\lambda_1^*(t) + \lambda_3^*(t)\ x_2^*(t)\right) \\
u_2^* &= -\frac{1}{r_{22}}\lambda_2^*(t).
\end{aligned}$$

In addition, the second-order *strengthened Legendre-Clebsch* (or convexity) *Condition* for a minimizer $(\mathbf{x}^*(t),\mathbf{u}^*(t),\boldsymbol{\lambda}^*(t))$ are expressed as follows $\mathbf{D}_{\mathbf{uu}}\mathcal{H}(\mathbf{x}^*(t),\mathbf{u}^*(t),\boldsymbol{\lambda}^*(t)) > \mathbf{0}$. In our particular case, this condition requires the matrix $\mathbf{R}$ to be positive definite. Substituting the input expressions in equations (5.0.5) and (5.0.6) leads to the following two-point boundary problem:

$$\begin{bmatrix} \dot{\lambda}_1^*(t) \\ \dot{\lambda}_2^*(t) \\ \dot{\lambda}_3^*(t) \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{\lambda_3^*(t)}{r_{11}}\ \left(\lambda_1^*(t) + \lambda_3^*(t)\ x_2^*(t)\right) \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \dot{x}_1^*(t) \\ \dot{x}_2^*(t) \\ \dot{x}_3^*(t) \end{bmatrix} = \begin{bmatrix} -\frac{1}{r_{11}}\left(\lambda_1^*(t) + \lambda_3^*(t)\ x_2^*(t)\right) \\ -\frac{1}{r_{22}}\lambda_2^*(t) \\ -\frac{x_2(t)}{r_{11}}\left(\lambda_1^*(t) + \lambda_3^*(t)\ x_2^*(t)\right) \end{bmatrix};\quad \mathbf{x}^*(t_0) = \begin{bmatrix} 10 \\ 2 \\ 20 \end{bmatrix},\quad \mathbf{x}^*(t_f) = \begin{bmatrix} 27 \\ 8 \\ 35 \end{bmatrix}.$$

We begin by computing the solution of this two-point boundary problem numerically. The optimal trajectory and the optimal input are illustrated in Figure 5.1 and Figure 5.2 respectively.

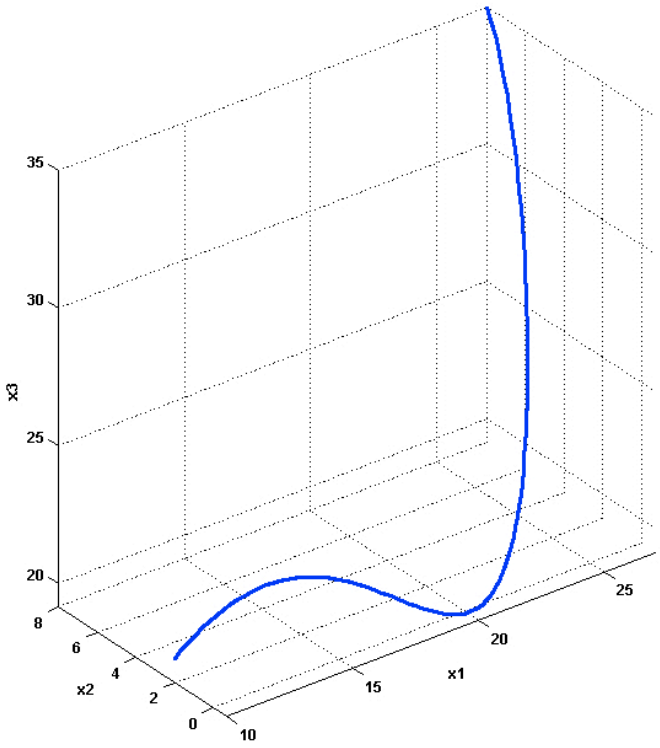

Figure 5.1: Optimal State Trajectory

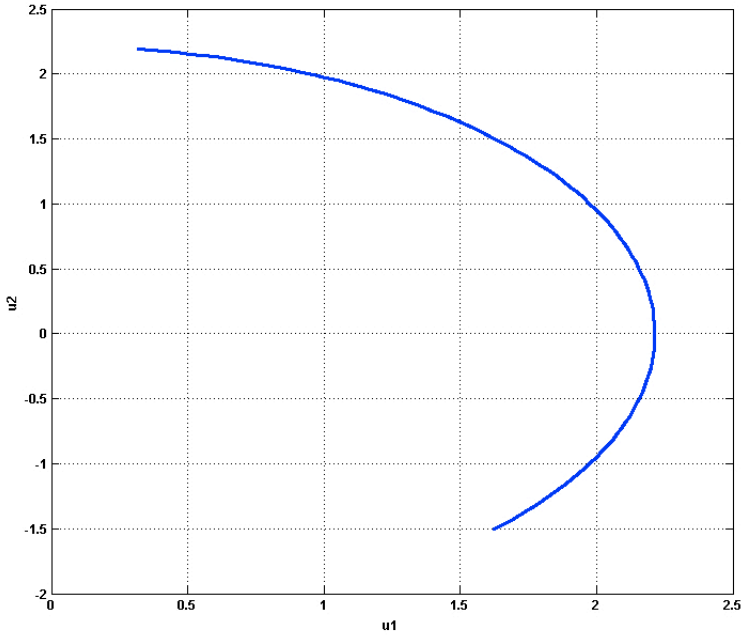

Figure 5.2: Optimal Input Trajectory

Example 5.2 (Analytical - Revisited using control points as decision variables)**.** Let us now apply the methodology described in this chapter to obtain a numerical solution to the same problem. That is, let us express the states and inputs in terms of piecewise polynomial functions expressed in terms of a linear combination of NURBS basis functions. In particular, we will parameterize the states and inputs as follows:

$$x_i \in \mathcal{P}_{\mathbf{b}_i,o_i,s_i}, \quad i=1,\dots 3, \text{ with } \mathbf{b}_i = \begin{bmatrix} 0 & 3.3333 & 6.6667 & 10.0000 \end{bmatrix},\ o_i = 6,\ s_i = 2$$
$$u_j \in \mathcal{P}_{\mathbf{b}_j,o_j,s_j}, \quad j=1,\dots 2, \text{ with } \mathbf{b}_j = \begin{bmatrix} 0 & 2.5000 & 5.0000 & 7.5000 & 10.0000 \end{bmatrix}, o_j = 5,\ s_j = 0$$

As mentioned at the beginning of this chapter, we will only allow control points as decision variables. That is, we will exploit Property 7 of NURBS curves to obtain:

$$\begin{aligned} x_i^{(r_i)}(t,\mathbf{w}_i,\mathbf{p}_i) &= \left[\mathcal{B}_{d_i}^{(r_i)}(t)\right]^T \mathbf{p}_i,\ i=1,\dots,3 \\ u_j^{(0)}(t,\mathbf{w}_j,\mathbf{p}_j) &= \left[\mathcal{B}_{d_j}^{(0)}(t)\right]^T \mathbf{p}_j,\ j=1,\dots,2 \end{aligned}$$

We then proceed to choose a uniform time partition (collocation points): $t_0 = \tau_0 < \tau_1 < \dots \tau_{N_\tau-2} < \tau_{N_\tau-1} = t_f$ where $\tau_i = t_0 + i\Delta\tau,\ i=0,\dots,N_\tau-1,\ \Delta\tau = \frac{t_f-t_0}{N_\tau-1}$. Then, at each of these points, τ_k, we evaluate the states and inputs as follows:

$$\begin{bmatrix} x_1^{(0)}(\tau_k,\mathbf{p}_1) \\ x_1^{(1)}(\tau_k,\mathbf{p}_1) \\ x_2^{(0)}(\tau_k,\mathbf{p}_2) \\ x_2^{(1)}(\tau_k,\mathbf{p}_2) \\ x_3^{(0)}(\tau_k,\mathbf{p}_3) \\ x_3^{(1)}(\tau_k,\mathbf{p}_3) \\ u_1^{(0)}(\tau_k,\mathbf{p}_4) \\ u_2^{(0)}(\tau_k,\mathbf{p}_5) \end{bmatrix} = \begin{bmatrix} \left[\mathcal{B}_5^{(0)}(\tau_k)\right]^T & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \left[\mathcal{B}_5^{(1)}(\tau_k)\right]^T & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \left[\mathcal{B}_5^{(0)}(\tau_k)\right]^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \left[\mathcal{B}_5^{(1)}(\tau_k)\right]^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \left[\mathcal{B}_5^{(0)}(\tau_k)\right]^T & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \left[\mathcal{B}_5^{(1)}(\tau_k)\right]^T & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \left[\mathcal{B}_4^{(0)}(\tau_k)\right]^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \left[\mathcal{B}_4^{(0)}(\tau_k)\right]^T \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \mathbf{p}_4 \\ \mathbf{p}_5 \end{bmatrix}.$$

The collocations points are chosen ahead of time and therefore the states and inputs depend only on the control points. By substituting these variables back into the optimal control problem, we obtain an nonlinear programming problem. Moreover, we make use of Proposition 3.1 to obtain the partial derivatives of all the functions involved in terms of the control points. Finally, we solve the nonlinear programming problem to obtain the coefficients of the piecewise polynomial parameterization and obtain a numerical approximations of the optimal minimizer. Figure 5.3 and Figure 5.4 show the numerical state and input trajectories respectively and they are plotted side-by-side with the the optimal state and input trajectories found analytically by using the first- and second-order necessary

conditions obtained through the calculus of variations.

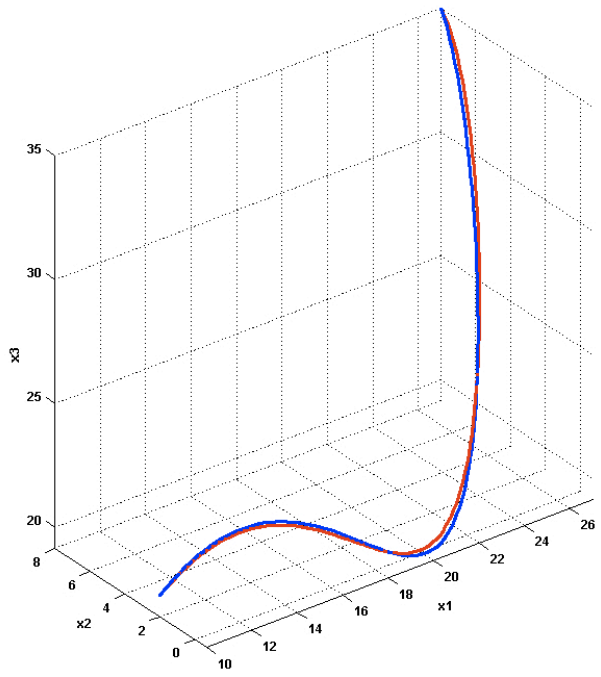

Figure 5.3: Numerical optimal state trajectory with control points only (red), and Analytical optimal state trajectory (blue).

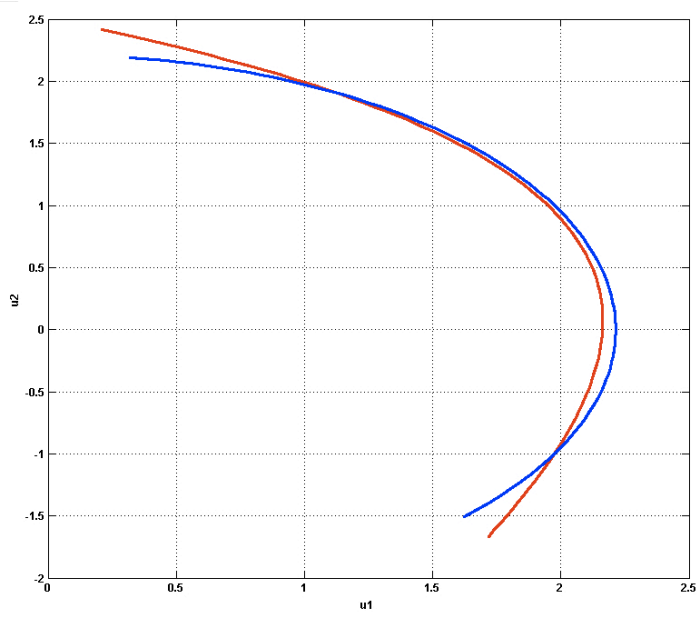

Figure 5.4: Numerical optimal input trajectory with control points only (red), and Analytical optimal input trajectory (blue).

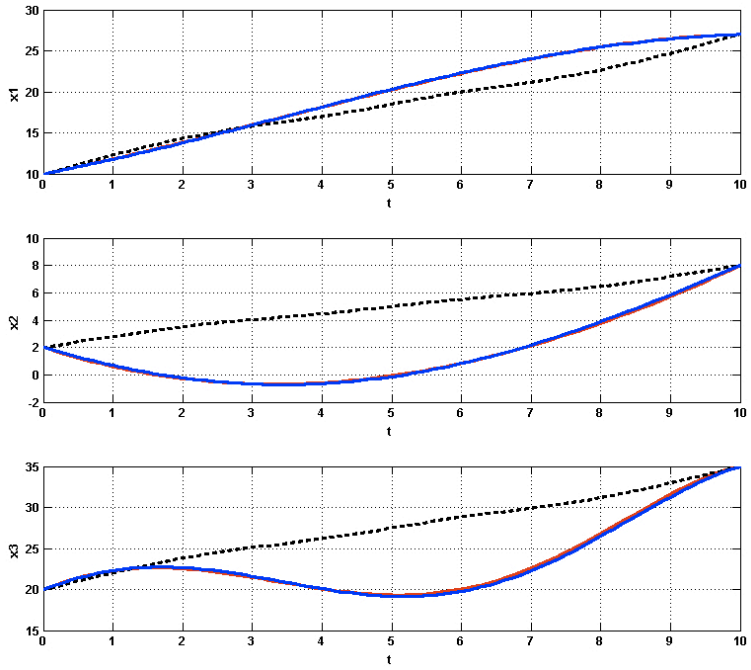

Figure 5.5: Initial guess for states (black), Numerical optimal states with control points only (red), and Analytical optimal states (blue).

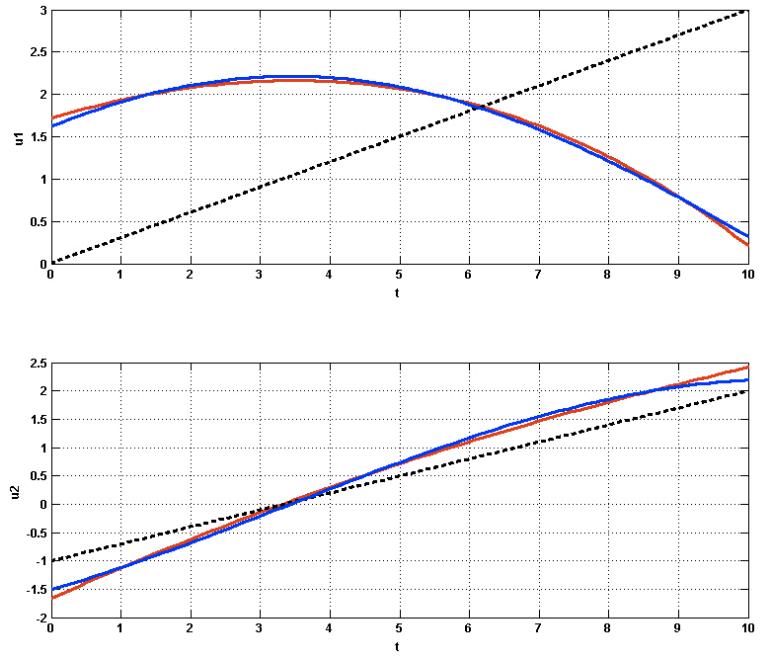

Figure 5.6: Initial guess for inputs (black), Numerical optimal inputs with control points only (red), and Analytical optimal inputs (blue).

Example 5.3 (Alice - Caltech's autonomous system)**.** Consider the following kinematic bicycle model for the vehicle:

$$\begin{aligned}
\frac{dx(t)}{dt} &= v(t)\ \cos(\theta(t)) \\
\frac{dy(t)}{dt} &= v(t)\ \sin(\theta(t)) \\
\frac{dv(t)}{dt} &= a(t) \\
\frac{d\theta(t)}{dt} &= \frac{v(t)}{L}\ \tan(\phi(t))
\end{aligned}$$

with states $\mathbf{x} = (x, y, v, \theta) \in \mathbb{R}^4$ and inputs $\mathbf{u} = (a, \phi) \in \mathbb{R}^2$. The equilibrium solutions of the system are $\mathbf{x}_e = (*, *, 0, *)$ and $\mathbf{u}_e = (0, (\Gamma - \frac{1}{2})\pi)$, $\Gamma \notin \mathbb{Z}$. The $*$ implies that the value is arbitrary. All equilibrium solutions are linearly controllable. The Controllability matrix is defined as follows:

$$\mathcal{C} = \begin{bmatrix}
0 & 0 & \cos(\theta) & 0 & \cos(\theta)^2 & 0 & \cos(\theta)^3 & 0 \\
0 & 0 & \sin(\theta) & 0 & \sin(\theta)^2 & 0 & \sin(\theta)^3 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{L}\tan(\pi(\Gamma - \frac{1}{2})) & 0 & \frac{1}{L^2}\tan(\pi(\Gamma - \frac{1}{2}))^2 & 0 & \frac{1}{L^3}\tan(\pi(\Gamma - \frac{1}{2}))^3 & 0
\end{bmatrix}$$

In particular, we will be interested in solving the following optimal control problem whose cost functional penalizes the inputs, the rate of steering and time required to perform a maneuver:

$$\min \int_{t_0}^{t_f} \left[a^2(t) + \phi^2(t) + \dot{\phi}^2(t) \right] \, dt + t_f$$

subject to

Dynamic Constraints:

$$\begin{aligned}
v(t)\ \cos(\theta(t)) - \dot{x}(t) &= 0 \\
v(t)\ \sin(\theta(t)) - \dot{y}(t) &= 0 \\
a(t) - \dot{v}(t) &= 0 \\
\frac{v(t)}{L}\ \tan(\phi(t)) - \dot{\theta}(t) &= 0
\end{aligned}$$

Trajectory Constraints:

$$\begin{aligned}
0 \leq\ & v(t) && \leq v_{max} \\
a_{\min} \leq\ & a(t) && \leq a_{\max} \\
\phi_{\min} \leq\ & \phi(t) && \leq \phi_{\max} \\
\dot{\phi}_{\min} \leq\ & \dot{\phi}(t) && \leq \dot{\phi}_{\max} \\
-\frac{gW}{2h_{cg}} \leq\ & \frac{v^2(t)\ \tan(\phi(t))}{L} && \leq \frac{gW}{2h_{cg}}
\end{aligned}$$

Initial and Final Boundary Conditions:

$$\begin{aligned}(x(t_0), y(t_0), v(t_0), \theta(t_0), a(t_0), \phi(t_0)) &= (0, 0, 0, 0, 0, 0)\\ (x(t_f), y(t_f), v(t_f), \theta(t_f), a(t_f), \phi(t_f)) &= (50, -30, 0, 0, 0, 0)\end{aligned}$$

Moreover, assume the following values for the set of parameters:

$$\begin{aligned} g &= 9.81 \text{ m}/s^2 & L &= 5.4356 \text{ m}\\ W &= 2.1336 \text{ m} & h_{cg} &= 1.0668 \text{ m}\\ v_{max} &= 60 \text{ mi/hr} & v_{min} &= 0.01 \text{ m/s}\\ a_{max} &= 1.0 \text{ m}/s^2 & a_{min} &= -3.0 \text{ m}/s^2\\ \phi_{max} &= 25.75 \ ^\circ & \phi_{min} &= -25.75 \ ^\circ\\ \dot{\phi}_{max} &= 75 \ ^\circ & \dot{\phi}_{min} &= -75 \ ^\circ \end{aligned}$$

Since we do not know ahead of time the amount of time required to perform a maneuver, we will dimensionalize the equations of motion with respect to time and allow the final time to vary. In particular, let $\tau = \frac{t}{\xi}$, $v = \frac{u}{\xi}$ and $a = \frac{\mathcal{A}}{\xi^2}$.

$$\begin{aligned}\frac{dx(\tau)}{d\tau} &= u(\tau)\ \cos(\theta(\tau))\\ \frac{dy(\tau)}{d\tau} &= u(\tau)\ \sin(\theta(\tau))\\ \frac{du(\tau)}{d\tau} &= \mathcal{A}(\tau)\\ \frac{d\theta(\tau)}{dt} &= \frac{u(\tau)}{L}\ \tan(\phi(\tau))\\ \frac{d\xi(\tau)}{d\tau} &= 0\end{aligned}$$

The optimal control problem then becomes:

$$\begin{aligned}\min \quad & \xi(1) + \int_0^1 \left[\left(\tfrac{\mathcal{A}(\tau)}{\xi^2(\tau)} \right)^2 + \phi(\tau)^2 + \left(\tfrac{\dot{\phi}(\tau)}{\xi(\tau)} \right)^2 \right] d\tau\\ \text{subject to}\end{aligned}$$

Dynamic Constraints:

$$\begin{aligned} u(\tau)\ \cos(\theta(\tau)) - \dot{x}(\tau) &= 0\\ u(\tau)\ \sin(\theta(\tau)) - \dot{y}(\tau) &= 0\\ \mathcal{A}(\tau) - \dot{u}(\tau) &= 0\\ \frac{u(\tau)}{L}\ \tan(\phi(\tau)) - \dot{\theta}(\tau) &= 0\\ \dot{\xi}(\tau) &= 0\end{aligned}$$

Trajectory Constraints:

$$\begin{array}{rcl} 0 \leq & \frac{u(\tau)}{v_{\max}\xi(\tau)} & \leq 1 \\ a_{\min} \leq & \frac{\mathcal{A}(\tau)}{\xi^2} & \leq a_{\max} \\ \phi_{\min} \leq & \phi(\tau) & \leq \phi_{\max} \\ \dot{\phi}_{\min} \leq & \frac{\dot{\phi}(\tau)}{\xi(\tau)} & \leq \dot{\phi}_{\max} \\ -\frac{gW}{2h_{cg}} \leq & \frac{u^2(\tau)\ \tan(\phi(\tau))}{L\ \xi^2(\tau)} & \leq \frac{gW}{2h_{cg}} \end{array}$$

Initial and Final Boundary Conditions:

$$\begin{array}{rcl} (x(\tau_0), y(\tau_0), u(\tau_0), \theta(\tau_0), \mathcal{A}(\tau_0), \phi(\tau_0)) & = & (0,0,0,0,0,0) \\ (x(\tau_f), y(\tau_f), u(\tau_f), \theta(\tau_f), \mathcal{A}(\tau_f), \phi(\tau_f)) & = & (50,-30,0,0,0,0) \end{array}$$

The minimizer of this optimal control problem is shown in Figure 5.7, together with the state and input evolutions (see Figure 5.8 and Figure 5.9).

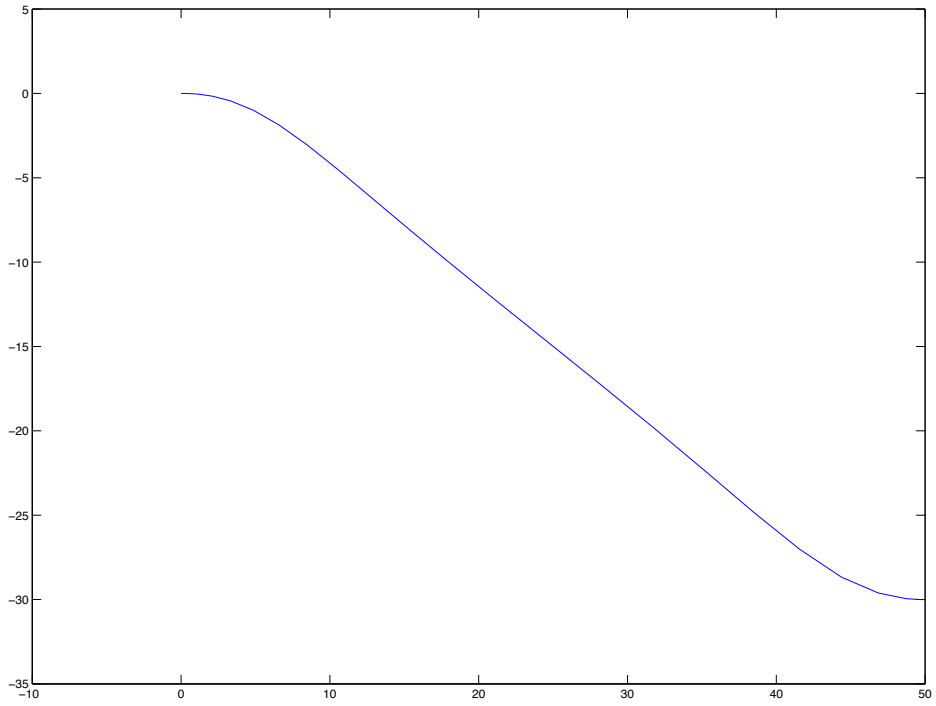

Figure 5.7: optimal trajectory

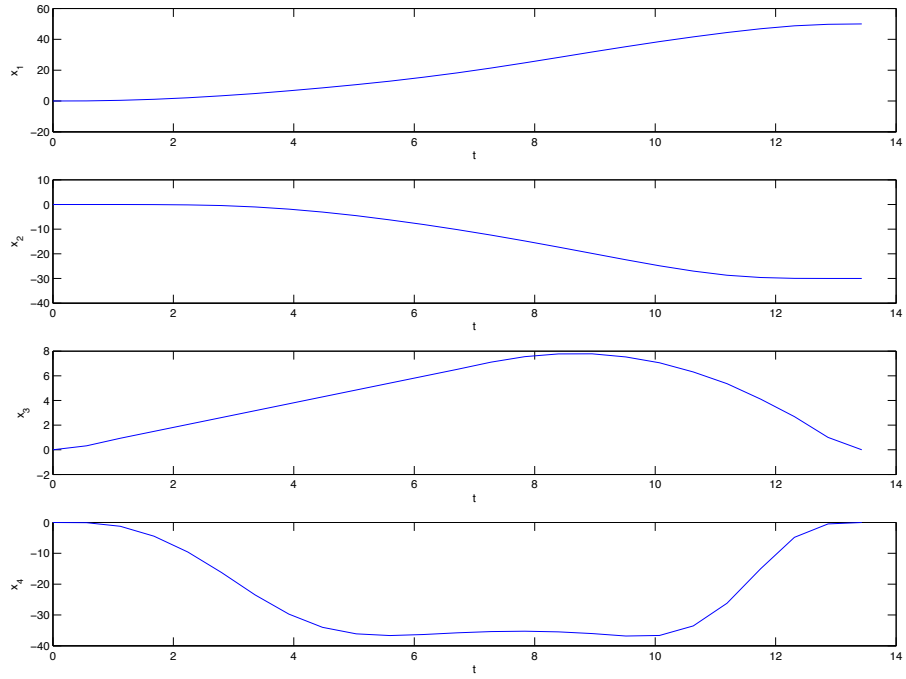

Figure 5.8: optimal states evolutions

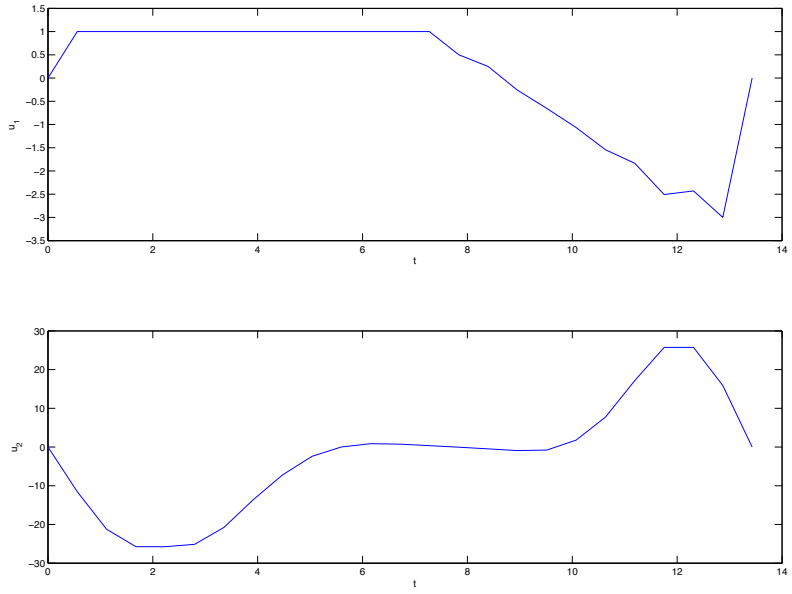

Figure 5.9: optimal input evolution

Chapter 6

Exploiting Properties of NURBS Curves

In this chapter, we will expand on some of the properties of NURBS curves that will be exploited for trajectory generation. In particular, we will unravel the connection between the two structures being combined in the NURBS curve definition: NURBS basis functions and the union of overlapping polytopes.

6.1 Local Support Property

We will begin by expanding on the local support property of NURBS basis functions. It is important to keep in mind that we are dealing with piecewise polynomial functions and, as a result, only a small subset of the NURBS basis functions give rise to each of the polynomial pieces. The relevant subset of NURBS basis functions per polynomial piece is specified through the location of the t parameter of the parameterization. The value of the t parameter singles out a specific knot span and, according to the local support property, for any given knot span at most $d+1$ of the $\mathcal{R}^{(r)}_{j,d}(t)$ basis are nonzero. More specifically, only the basis functions $\mathcal{R}^{(r)}_{\ell-d,d}(t), \ldots, \mathcal{R}^{(r)}_{\ell,d}(t)$ could be nonzero in the knot span $[k_\ell, k_{\ell+1})$, $\ell \in [d, d+m(N_p-1)]$. Therefore, by exploiting the local support property of the NURBS basis functions, we express a NURBS curve as follows:

$$c^{(r)}(t,\mathbf{w},\mathbf{p}) \;=\; \frac{\sum_{j=\ell-d}^{\ell} \mathcal{B}^{(r)}_{j,d}(t)\; w_j\; p_j}{\sum_{i=\ell-d}^{\ell} \mathcal{B}^{(0)}_{i,d}(t)\; w_i} - \sum_{k=1}^{r} \binom{r}{k} \frac{\sum_{j=\ell-d}^{\ell} \mathcal{B}^{(k)}_{j,d}(t)\; w_j}{\sum_{i=\ell-d}^{\ell} \mathcal{B}^{(0)}_{i,d}(t)\; w_i}\; c^{(r-k)}(t,\mathbf{w},\mathbf{p}), \quad t \in [k_\ell, k_{\ell+1})\,.$$

Alternatively, we may rewrite a NURBS curve in terms of the individual polynomial pieces:

$$c_n^{(r)}(t,\mathbf{w},\mathbf{p}) \;=\; \frac{\sum_{j=\ell-d}^{\ell}\mathcal{B}_{j,d}^{(r)}(t)\;w_j\;p_j}{\sum_{i=\ell-d}^{\ell}\mathcal{B}_{i,d}^{(0)}(t)\;w_i} - \sum_{k=1}^{r}\binom{r}{k}\frac{\sum_{j=\ell-d}^{\ell}\mathcal{B}_{j,d}^{(k)}(t)\;w_j}{\sum_{i=\ell-d}^{\ell}\mathcal{B}_{i,d}^{(0)}(t)\;w_i}\;c_n^{(r-k)}(t,\mathbf{w},\mathbf{p}),\quad t\in[b_n,b_{n+1})\,,$$

where $n=0,\ldots,N_p-1$, and $\ell=d+mn$, and m is the multiplicity of the inner breakpoints.

This is advantageous computationally because we only have to consider at most $(d+1)$ NURBS basis functions instead of N_c NURBS basis functions. Moreover, in later chapters, this property will give rise to sparse matrices of the first and second-order partial derivatives of the cost and constraint functions.

In matrix form:

$$c^{(r)}(t,\mathbf{w},\mathbf{p}) \;=\; \frac{\mathbf{w}^T\mathrm{diag}(\{\mathcal{B}_d^{(r)}(t)\})\mathbf{p}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}} - \sum_{k=1}^{r}\binom{r}{k}\frac{\left[\mathcal{B}_d^{(k)}(t)\right]^T\mathbf{w}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}\;c^{(r-k)}(t,\mathbf{w},\mathbf{p}),\quad t\in[k_\ell,k_{\ell+1})\,,$$

where

$$\left[\mathcal{B}_d^{(r)}(t)\right]^T = \left[\begin{array}{ccccccc}\mathcal{B}_{\ell-d,d}^{(r)}(t) & \mathcal{B}_{\ell-d+1,d}^{(r)}(t) & \mathcal{B}_{\ell-d+2,d}^{(r)}(t) & \ldots & \mathcal{B}_{\ell-2,d}^{(r)}(t) & \mathcal{B}_{\ell-1,d}^{(r)}(t) & \mathcal{B}_{\ell,d}^{(r)}(t)\end{array}\right]^T.$$

The active bases for $t\in[k_\ell,k_{\ell+1})$ and their dependence on lower degree active basis functions with the same rth derivative give rise to the following triangular expansion:

$$\begin{array}{ccccccccc}
\mathcal{B}_{\ell-d,d}^{(r)}(t) & \mathcal{B}_{\ell-d+1,d}^{(r)}(t) & \mathcal{B}_{\ell-d+2,d}^{(r)}(t) & \mathcal{B}_{\ell-d+3,d}^{(r)}(t) & \ldots & \mathcal{B}_{\ell-2,d}^{(r)}(t) & \mathcal{B}_{\ell-1,d}^{(r)}(t) & \mathcal{B}_{\ell,d}^{(r)}(t)\\
& \mathcal{B}_{\ell-d+1,d-1}^{(r)}(t) & \mathcal{B}_{\ell-d+2,d-1}^{(r)}(t) & \mathcal{B}_{\ell-d+3,d-1}^{(r)}(t) & \ldots & \mathcal{B}_{\ell-2,d-1}^{(r)}(t) & \mathcal{B}_{\ell-1,d-1}^{(r)}(t) & \mathcal{B}_{\ell,d-1}^{(r)}(t)\\
& & \mathcal{B}_{\ell-d+2,d-2}^{(r)}(t) & \mathcal{B}_{\ell-d+3,d-2}^{(r)}(t) & \ldots & \mathcal{B}_{\ell-2,d-2}^{(r)}(t) & \mathcal{B}_{\ell-1,d-2}^{(r)}(t) & \mathcal{B}_{\ell,d-2}^{(r)}(t)\\
& & & & \ddots & \vdots & \vdots & \vdots\\
& & & & & \mathcal{B}_{\ell-2,2}^{(r)}(t) & \mathcal{B}_{\ell-1,2}^{(r)}(t) & \mathcal{B}_{\ell,2}^{(r)}(t)\\
& & & & & & \mathcal{B}_{\ell-1,1}^{(r)}(t) & \mathcal{B}_{\ell,1}^{(r)}(t)\\
& & & & & & & \mathcal{B}_{\ell,0}^{(r)}(t).
\end{array}$$

Replacing the B-Spline basis definition, we are able to express their actual numerical dependence. Let us begin by expressing the numerical relation between the basis functions of the first two rows:

$$\begin{bmatrix} \mathcal{B}^{(r)}_{\delta-1,d}(t) \\ \mathcal{B}^{(r)}_{\delta,d}(t) \\ \mathcal{B}^{(r)}_{\delta+1,d}(t) \\ \vdots \\ \mathcal{B}^{(r)}_{\delta+d-3,d}(t) \\ \mathcal{B}^{(r)}_{\delta+d-2,d}(t) \\ \mathcal{B}^{(r)}_{\delta+d-1,d}(t) \end{bmatrix} = \left[\bar{\mathcal{B}}^{(r)}_{d,d-1}(t)\right] \begin{bmatrix} \mathcal{B}^{(r)}_{\delta,d-1}(t) \\ \mathcal{B}^{(r)}_{\delta+1,d-1}(t) \\ \mathcal{B}^{(r)}_{\delta+2,d-1}(t) \\ \vdots \\ \mathcal{B}^{(r)}_{\delta+d-3,d-1}(t) \\ \mathcal{B}^{(r)}_{\delta+d-2,d-1}(t) \\ \mathcal{B}^{(r)}_{\delta+d-1,d-1}(t) \end{bmatrix}, \quad t \in [k_\ell, k_{\ell+1}), \quad r \in \{0, 1, \ldots, d-1\},$$

where

$$\left[\bar{\mathcal{B}}^{(r)}_{d,d-1}(t)\right] = \frac{d}{d-r} \begin{bmatrix} -\frac{t-k_{\delta+d}}{\Delta k_\delta} & 0 & 0 & \ldots & 0 & 0 & 0 \\ \frac{t-k_\delta}{\Delta k_\delta} & -\frac{t-k_{\delta+d+1}}{\Delta k_{\delta+1}} & 0 & \ldots & 0 & 0 & 0 \\ 0 & \frac{t-k_{\delta+1}}{\Delta k_{\delta+1}} & -\frac{t-k_{\delta+d+2}}{\Delta k_{\delta+2}} & \ldots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & \frac{t-k_{\delta+d-3}}{\Delta k_{\delta+d-3}} & -\frac{t-k_{\delta+2d-2}}{\Delta k_{\delta+d-2}} & 0 \\ 0 & 0 & 0 & \ldots & 0 & \frac{t-k_{\delta+d-2}}{\Delta k_{\delta+d-2}} & -\frac{t-k_{\delta+2d-1}}{\Delta k_{\delta+d-1}} \\ 0 & 0 & 0 & \ldots & 0 & 0 & \frac{t-k_{\delta+d-1}}{\Delta k_{\delta+d-1}} \end{bmatrix},$$

with $\Delta k_\delta = k_{\delta+d} - k_\delta,\ \delta = \ell - (d-1)$. In general, we express the recursive numerical relation as follows:

$$\begin{bmatrix} \mathcal{B}^{(r)}_{\delta-1,d}(t) \\ \vdots \\ \mathcal{B}^{(r)}_{\delta+d-1,d}(t) \end{bmatrix} = \bar{\mathcal{B}}^{(r)}_{d,d-1}(t)\ \bar{\mathcal{B}}^{(r)}_{d-1,d-2}(t) \ldots \bar{\mathcal{B}}^{(r)}_{1,0}(t)\ \mathcal{B}^{(r)}_{\delta+d-1,0}(t), \quad r \in \{0, 1, \ldots, d-1\}.$$

Moreover, the active basis for $t \in [k_\ell, k_{\ell+1})$ and their dependence on lower degree active basis functions with decreasing rth derivative give rise to the following expansion:

$$\begin{array}{ccccccccc}
\mathcal{B}^{(r)}_{\ell-d,d}(t) & \mathcal{B}^{(r)}_{\ell-d+1,d}(t) & \mathcal{B}^{(r)}_{\ell-d+2,d}(t) & \mathcal{B}^{(r)}_{\ell-d+3,d}(t) & \dots & \mathcal{B}^{(r)}_{\ell-2,d}(t) & \mathcal{B}^{(r)}_{\ell-1,d}(t) & \mathcal{B}^{(r)}_{\ell,d}(t) \\
& \mathcal{B}^{(r-1)}_{\ell-d+1,d-1}(t) & \mathcal{B}^{(r-1)}_{\ell-d+2,d-1}(t) & \mathcal{B}^{(r-1)}_{\ell-d+3,d-1}(t) & \dots & \mathcal{B}^{(r-1)}_{\ell-2,d-1}(t) & \mathcal{B}^{(r-1)}_{\ell-1,d-1}(t) & \mathcal{B}^{(r-1)}_{\ell,d-1}(t) \\
& & \mathcal{B}^{(r-2)}_{\ell-d+2,d-2}(t) & \mathcal{B}^{(r-2)}_{\ell-d+3,d-2}(t) & \dots & \mathcal{B}^{(r-2)}_{\ell-2,d-2}(t) & \mathcal{B}^{(r-2)}_{\ell-1,d-2}(t) & \mathcal{B}^{(r-2)}_{\ell,d-2}(t) \\
& & & & \ddots & \vdots & \vdots & \vdots
\end{array}$$

Replacing the B-Spline basis definition, we are able to express their actual numerical dependence. Let us first determine the dependence between the basis functions in the first and second rows:

$$\begin{bmatrix} \mathcal{B}^{(r)}_{\delta-1,d}(t) \\ \mathcal{B}^{(r)}_{\delta,d}(t) \\ \mathcal{B}^{(r)}_{\delta+1,d}(t) \\ \vdots \\ \mathcal{B}^{(r)}_{\delta+d-3,d}(t) \\ \mathcal{B}^{(r)}_{\delta+d-2,d}(t) \\ \mathcal{B}^{(r)}_{\delta+d-1,d}(t) \end{bmatrix} = \left[\bar{\mathcal{B}}^{(r,r-1)}_{d,d-1}(t)\right] \begin{bmatrix} \mathcal{B}^{(r-1)}_{\delta,d-1}(t) \\ \mathcal{B}^{(r-1)}_{\delta+1,d-1}(t) \\ \mathcal{B}^{(r-1)}_{\delta+2,d-1}(t) \\ \vdots \\ \mathcal{B}^{(r-1)}_{\delta+d-3,d-1}(t) \\ \mathcal{B}^{(r-1)}_{\delta+d-2,d-1}(t) \\ \mathcal{B}^{(r-1)}_{\delta+d-1,d-1}(t) \end{bmatrix}, \quad t \in [k_\ell, k_{\ell+1}),$$

where

$$\left[\bar{\mathcal{B}}^{(r,r-1)}_{d,d-1}(t)\right] = d \begin{bmatrix} -\frac{1}{\Delta k_\delta} & 0 & 0 & \dots & 0 & 0 & 0 \\ \frac{1}{\Delta k_\delta} & -\frac{1}{\Delta k_{\delta+1}} & 0 & \dots & 0 & 0 & 0 \\ 0 & \frac{1}{\Delta k_{\delta+1}} & -\frac{1}{\Delta k_{\delta+2}} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \frac{1}{\Delta k_{\delta+d-3}} & -\frac{1}{\Delta k_{\delta+d-2}} & 0 \\ 0 & 0 & 0 & \dots & 0 & \frac{1}{\Delta k_{\delta+d-2}} & -\frac{1}{\Delta k_{\delta+d-1}} \\ 0 & 0 & 0 & \dots & 0 & 0 & \frac{1}{\Delta k_{\delta+d-1}} \end{bmatrix}$$

with $\Delta k_\delta = k_{\delta+d} - k_\delta,\ \delta = \ell - (d-1)$ as before. In general, we express the recursive numerical

relation as follows:

$$\begin{bmatrix} \mathcal{B}^{(r)}_{\delta-1,d}(t) \\ \vdots \\ \mathcal{B}^{(r)}_{\delta+d-1,d}(t) \end{bmatrix} = \bar{\mathcal{B}}^{(r,r-1)}_{d,d-1}(t)\ \bar{\mathcal{B}}^{(r-1,r-2)}_{d-1,d-2}(t)\dots\bar{\mathcal{B}}^{(1,0)}_{d-r+1,d-r}(t) \begin{bmatrix} \mathcal{B}^{(0)}_{\delta+r-1,d-r}(t) \\ \vdots \\ \mathcal{B}^{(0)}_{\delta+d-1,d-r}(t) \end{bmatrix}, \quad r \in \{1,2,\dots,d-1\}.$$

In summary, the active basis functions for any desired rth derivative are obtained as follows:

$$\begin{bmatrix} \mathcal{B}^{(r)}_{\delta-1,d}(t) \\ \vdots \\ \mathcal{B}^{(r)}_{\delta+d-1,d}(t) \end{bmatrix} = \prod_{j=0}^{r-1} \bar{\mathcal{B}}^{(r-j,r-(j+1))}_{d-j,d-(j+1)}(t) \prod_{j=r}^{d-1} \bar{\mathcal{B}}^{(0)}_{d-j,d-(j+1)}(t), \quad r \in \{1,2,\dots,d-1\},\ t \in [k_\ell, k_{\ell+1}).$$

Example 6.1. Consider $d = 3$, $r = 2$. Then, $\delta = \ell - (d-1) = \ell - 2$, and the active basis functions are computed as follows:

$$\begin{bmatrix} \mathcal{B}^{(2)}_{\ell-3,3}(t) \\ \mathcal{B}^{(2)}_{\ell-2,3}(t) \\ \mathcal{B}^{(2)}_{\ell-1,3}(t) \\ \mathcal{B}^{(2)}_{\ell,3}(t) \end{bmatrix} = \prod_{j=0}^{1} \bar{\mathcal{B}}^{(2-j,2-(j+1))}_{3-j,3-(j+1)}(t) \prod_{j=2}^{2} \bar{\mathcal{B}}^{(0)}_{3-j,3-(j+1)}(t) = \bar{\mathcal{B}}^{(2,1)}_{3,2}(t)\ \bar{\mathcal{B}}^{(1,0)}_{2,1}(t)\ \bar{\mathcal{B}}^{(0)}_{1,0}(t), \quad t \in [k_\ell, k_{\ell+1}),$$

where

$$\bar{\mathcal{B}}^{(2,1)}_{3,2}(t) = 3 \begin{bmatrix} -\frac{1}{\Delta k_{\ell-2}} & 0 & 0 \\ \frac{1}{\Delta k_{\ell-2}} & -\frac{1}{\Delta k_{\ell-1}} & 0 \\ 0 & \frac{1}{\Delta k_{\ell-1}} & -\frac{1}{\Delta k_\ell} \\ 0 & 0 & \frac{1}{\Delta k_\ell} \end{bmatrix}, \ \bar{\mathcal{B}}^{(1,0)}_{2,1}(t) = 2 \begin{bmatrix} -\frac{1}{\Delta k_{\ell-1}} & 0 \\ \frac{1}{\Delta k_{\ell-1}} & -\frac{1}{\Delta k_\ell} \\ 0 & \frac{1}{\Delta k_\ell} \end{bmatrix}, \ \bar{\mathcal{B}}^{(0)}_{1,0}(t) = 1 \begin{bmatrix} -\frac{t-k_{\ell+1}}{\Delta k_\ell} \\ \\ \frac{t-k_\ell}{\Delta k_\ell} \end{bmatrix}$$

$$\begin{bmatrix} \mathcal{B}^{(2)}_{\ell-3,3}(t) \\ \mathcal{B}^{(2)}_{\ell-2,3}(t) \\ \mathcal{B}^{(2)}_{\ell-1,3}(t) \\ \mathcal{B}^{(2)}_{\ell,3}(t) \end{bmatrix} = 3! \begin{bmatrix} -\frac{1}{\Delta k_{\ell-2}} & 0 & 0 \\ \frac{1}{\Delta k_{\ell-2}} & -\frac{1}{\Delta k_{\ell-1}} & 0 \\ 0 & \frac{1}{\Delta k_{\ell-1}} & -\frac{1}{\Delta k_\ell} \\ 0 & 0 & \frac{1}{\Delta k_\ell} \end{bmatrix} \begin{bmatrix} -\frac{1}{\Delta k_{\ell-1}} & 0 \\ \frac{1}{\Delta k_{\ell-1}} & -\frac{1}{\Delta k_\ell} \\ 0 & \frac{1}{\Delta k_\ell} \end{bmatrix} \begin{bmatrix} -\frac{t-k_{\ell+1}}{\Delta k_\ell} \\ \\ \frac{t-k_\ell}{\Delta k_\ell} \end{bmatrix}, \quad t \in [k_\ell, k_{\ell+1}).$$

If we exploit the local property of NURBS basis functions, then we are able to write equation (4.0.5) more succinctly as follows:

$$\tilde{\mathbf{z}}_i(\tau, \tilde{\mathbf{w}}_i, \tilde{\mathbf{p}}_i) = \begin{bmatrix} 0 & \dots & \mathcal{R}^{(0)}_{\ell_i-d_i,d_i}(\tau, \tilde{\mathbf{w}}_i) & \dots & \mathcal{R}^{(0)}_{\ell_i,d_i}(\tau, \tilde{\mathbf{w}}_i) & \dots & 0 \\ 0 & \dots & \mathcal{R}^{(1)}_{\ell_i-d_i,d_i}(\tau, \tilde{\mathbf{w}}_i) & \dots & \mathcal{R}^{(1)}_{\ell_i,d_i}(\tau, \tilde{\mathbf{w}}_i) & \dots & 0 \\ \vdots & \dots & \vdots & \dots & \vdots & \dots & \vdots \\ 0 & \dots & \mathcal{R}^{(D_i)}_{\ell_i-d_i,d_i}(\tau, \tilde{\mathbf{w}}_i) & \dots & \mathcal{R}^{(D_i)}_{\ell_i,d_i}(\tau, \tilde{\mathbf{w}}_i) & \dots & 0 \end{bmatrix} \begin{bmatrix} p^i_0 \\ \vdots \\ p^i_{\ell_i-d_i} \\ \vdots \\ p^i_{\ell_i} \\ \vdots \\ p^i_{N^c_i-1} \end{bmatrix},$$

with $\ell_i \in \{d_i, d_i + m_i, \dots, d_i + (N^i_p - 1)m_i\}$. That is, the local property of NURBS basis functions gives rise to sparse matrices.

6.2 Strong Convex Hull Property

Next, we will expand on the strong convex hull property of NURBS curves. This property states that a NURBS curve is contained in the convex hull of its control polygon. We will be mostly interested in exploiting this property in the case of d-dimensional space where $d \in \{2, 3\}$. We begin by stating precisely the notion of a path or trajectory.

Definition 6.1. A *path* in $\mathbb{R}^d$ is a map $\boldsymbol{\sigma} : [t_0, t_f] \to \mathbb{R}^d$. The points $\boldsymbol{\sigma}(t_0)$ and $\boldsymbol{\sigma}(t_f)$ are called the *endpoints* of the path. If $\boldsymbol{\sigma}$ is of class C^r, we say $\boldsymbol{\sigma}$ is an r-th differentiable path.

For these cases, we express a NURBS path in the form:

$$\begin{aligned} \mathbf{c}(t) = \begin{bmatrix} x_1(t) \\ \vdots \\ x_d(t) \end{bmatrix} &= \begin{bmatrix} \sum_{j=\ell-d}^{\ell} \mathcal{R}^{(0)}_{j,d}(t, \mathbf{w})\ p^{x_1}_j \\ \vdots \\ \sum_{j=\ell-d}^{\ell} \mathcal{R}^{(0)}_{j,d}(t, \mathbf{w})\ p^{x_d}_j \end{bmatrix} \\ &= \mathcal{R}^{(0)}_{\ell-d,d}(t, \mathbf{w}) \begin{bmatrix} p^{x_1}_{\ell-d} \\ \vdots \\ p^{x_d}_{\ell-d} \end{bmatrix} + \cdots + \mathcal{R}^{(0)}_{\ell,d}(t, \mathbf{w}) \begin{bmatrix} p^{x_1}_{\ell} \\ \vdots \\ p^{x_d}_{\ell} \end{bmatrix} \\ &= \begin{bmatrix} \mathcal{R}^{(0)}_{\ell-d,d}(t, \mathbf{w}) & \dots & \mathcal{R}^{(0)}_{\ell,d}(t, \mathbf{w}) \end{bmatrix} \begin{bmatrix} \mathbf{p}_{\ell-d} \\ \vdots \\ \mathbf{p}_{\ell} \end{bmatrix}, \end{aligned}$$

where $t \in [k_\ell, k_{\ell+1})$. Let us recast the NURBS path emphasizing the polynomial pieces:

$$\mathbf{c}_n(t) \;=\; \left[\begin{array}{ccc} \mathcal{R}^{(0)}_{\ell-d,d}(t,\mathbf{w}) & \ldots & \mathcal{R}^{(0)}_{\ell,d}(t,\mathbf{w}) \end{array}\right] \left[\begin{array}{c} \mathbf{p}_{\ell-d} \\ \vdots \\ \mathbf{p}_{\ell} \end{array}\right], \quad t \in [b_n, b_{n+1}),$$

with $n = 0, \ldots, N_p - 1,\ \ell = d + mn$. According to the strong convex hull property, the polynomial piece $\mathbf{c}_n(t)$ will remain inside the smallest convex set containing the control points $\mathbf{p}_{\ell-d}, \ldots, \mathbf{p}_{\ell}$; that is, $\mathbf{c}_n(t) \subset \text{conv}\left(\{\mathbf{p}_{\ell-d}, \ldots, \mathbf{p}_{\ell}\}\right)$. Moreover, due to the smoothness conditions imposed on the resulting NURBS curve, adjacent polytopes will share a fixed set of control points. If the NURBS curve is required to be continuous ($s = 0$), then the adjacent polytopes will share a single control point. If we require the NURBS curve to be continuously differentiable ($s = 1$), then adjacent polytopes will share two control points. In general, if we require the NURBS curve to be sth continuously differentiable (e.g., belongs to C^s), then adjacent polytopes will share $s + 1$ control points. This set of overlapping polytopes gives rise to a connected region (in general non-convex) which bounds the path $\mathbf{c}(t)$. It is this feature of ensuring that the NURBS path $\mathbf{c}(t)$ never leaves the delineated region trace by the union of adjacently overlapping polytopes which is most useful for trajectory generation purposes. In later chapters, this feature will be exploited to reduce the number of trajectory constraints involved in the optimal control problem.

The number of polytopes involved in bounding the NURBS path $\mathbf{c}(t)$ is equal in number to the polynomial pieces, N_p. These polytopes are obtained by computing the convex hull of $d+1$ control points at the time. Finally, smoothness conditions impose that at least $(s + 1)$ control points be shared between consecutive polytopes, giving rise to a non-empty intersection between them. The order of the sequence of control points, then, becomes important, and it is computed as follows:

$$\{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{N_c-1}\} = \bigcup_{i=1}^{N_p} \{\mathbf{p}_{(i-1)(d-s)}, \ldots, \mathbf{p}_{i(d-s)+s}\}. \tag{6.2.1}$$

The overlapping polytopes are then obtained using:

$$\mathcal{P}_i = \text{conv}\left(\{\mathbf{p}_{(i-1)(d-s)}, \ldots, \mathbf{p}_{i(d-s)+s}\}\right),\ \mathbf{p}_i \in \mathbb{R}^d,\ d = \{2, 3\},\ i = 1, \ldots, N_p. \tag{6.2.2}$$

In addition, we define the intersection of adjacent polytopes by

$$\mathcal{Q}_j = \mathcal{P}_j \cap \mathcal{P}_{j+1}, \quad j = 1, \ldots N_p - 1. \tag{6.2.3}$$

The sharing control points must lie in these intersection sets. That is,

$$\{\mathbf{p}_{j(d-s)}, \ldots, \mathbf{p}_{j(d-s)+s}\} \in \mathcal{Q}_j, \quad j = 1, \ldots N_p - 1. \tag{6.2.4}$$

Example 6.2. Let $N_p = 4$, $d+1 = 7$, $s+1 = 5$; then, the total number of control points is $N_c = 13$, and they are ordered as follows: $\{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{12}\} = \bigcup_{i=1}^{4} \{\mathbf{p}_{2(i-1)}, \ldots, \mathbf{p}_{2(i+2)}\}$, with the following overlapping pattern:

$$\begin{array}{ccccccccccccc} \mathbf{p}_0 & \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 & \mathbf{p}_4 & \mathbf{p}_5 & \mathbf{p}_6 & & & & & & \\ & & \mathbf{p}_2 & \mathbf{p}_3 & \mathbf{p}_4 & \mathbf{p}_5 & \mathbf{p}_6 & \mathbf{p}_7 & \mathbf{p}_8 & & & & \\ & & & & \mathbf{p}_4 & \mathbf{p}_5 & \mathbf{p}_6 & \mathbf{p}_7 & \mathbf{p}_8 & \mathbf{p}_9 & \mathbf{p}_{10} & & \\ & & & & & & \mathbf{p}_6 & \mathbf{p}_7 & \mathbf{p}_8 & \mathbf{p}_9 & \mathbf{p}_{10} & \mathbf{p}_{11} & \mathbf{p}_{12} \end{array}, \tag{6.2.5}$$

with each set of control points used to construct the polytopes $\mathcal{P}_i, i = 1, \ldots, N_p$. Assume that the 13 control points have the following coordinates:

$$\begin{bmatrix} 17.7505 & 19.1365 & 27.8785 & 50.0533 & 57.3028 & 60.1812 & 60.2878 & 73.2942 & 80.6503 & 83.5288 & 90.7783 & 90.3518 & 89.0725 \\ 21.7011 & 35.0556 & 18.0445 & 22.9730 & 44.2766 & 34.8967 & 55.8824 & 60.8108 & 45.3895 & 65.7393 & 68.4420 & 80.0477 & 87.9968 \end{bmatrix};$$

then, the union of overlapping polytopes may be graphically drawn as follows:

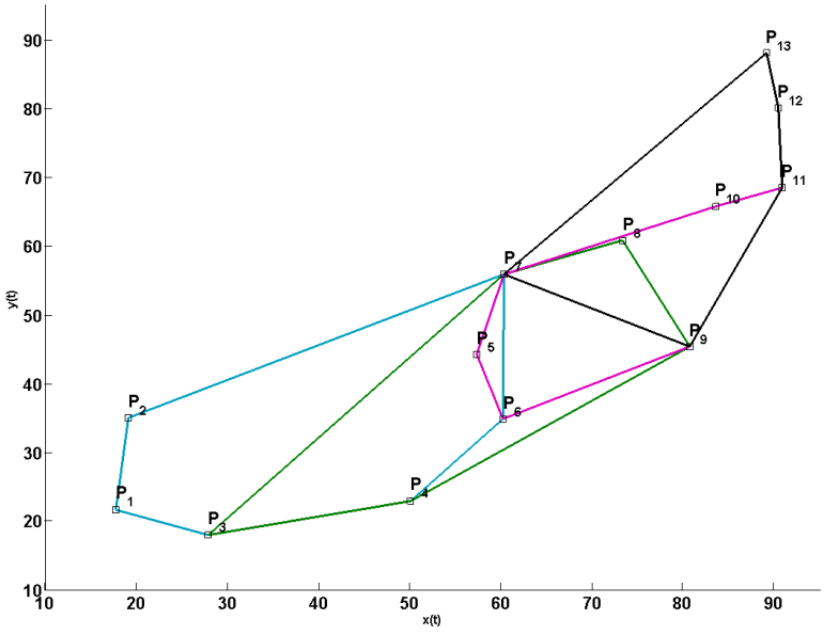

Figure 6.1: Union of overlapping polytopes

Note that each of the polytopes is constructed by the convex hull of $d+1$ points. In particular, we will be interested only on polytopes that have nonzero volume. That is, for $d = 2$ one starts with triangles, and for $d = 3$ one starts with tetrahedrons.

Chapter 7

Solving Optimal Control Problems with Active Weights and Control Points as Decision Variables

In this chapter, we will focus on the computation of minimizers for optimal control problems that model the effort required for differentially flat systems to evolve across state configurations while satisfying trajectory and actuator constraints. In order to relieve the burden often associated with direct methods, we will exploit the properties of both differentially flat systems and NURBS curves to induce a transformation of the original optimal control problem into a simpler, more favorable numerical computational form. This is accomplished by effectively removing the dynamic, trajectory, and/or actuator constraints from the original optimal control problem.

As previously determined by other authors Fliess et al. [1995], for differentially flat systems there exists a set of *flat outputs* (equal in number to the inputs) such that all states and inputs can be determined from these outputs without integration. Consequently, one can rewrite the optimal control problem in terms of the flat outputs and then find a minimizer in the flat-output space. Since the flat outputs implicitly contain all the information about the dynamics of the system by introducing this transformation, no explicit dynamic constraints remain in the transformed optimal control problem (removal of dynamic constraints).

If we further parameterize these flat outputs by piecewise polynomial functions in terms of a linear combination of NURBS basis functions, then we are able to exploit the fact that NURBS curves are the result of combining two interrelated structures — NURBS basis functions and a union of overlapping polytopes constructed from the coefficients of the linear combination. After discretization, the NURBS basis functions depend only on the weights, and the union of overlapping polytopes depends only on the control points. If, in particular, d of these flat outputs are related

parametrically and their respective parametric NURBS path lies in the vector space $\mathcal{P}_{\mathbf{b},o,s}$, then we are able to fix the coordinates of the control points in such a way that their control polygon delineates a region of space that avoids all obstructions in the path space and contains the initial and final path conditions. Since we are guaranteed that the resulting NURBS path will be bounded by the control polygon regardless of the values of the weights and independent of discretization, then we are able to remove these path constraints from the optimal control problem (removal of constraints).

After these two transformations, the original optimal control problem has been converted into a modified optimal control problem without dynamic, trajectory, and/or actuator constraints. We proceed, as in a conventional direct method, by transcribing the modified optimal control problem to a nonlinear programming problem with the remaining parameters (active weights and control points) as the decision variables. It is important to note that, in general, only a subset of the flat outputs is written in parametric form and, therefore, only the corresponding control points will be fixed in order to remove constraints. The control points of the remaining flat outputs (active control points) will still be allowed to vary.

In summary, the approach used in this chapter consists of the following steps:

a) Rewrite the original optimal control problem, using the fact that the dynamical system is differentially flat, in terms of the *flat outputs* and their derivatives (removal of dynamic constraints).

b) Parameterize the flat outputs by piecewise polynomial functions using a linear combination of *NURBS basis* functions.

c) Fix the *control points* of parametric NURBS paths in such a way that they delineate regions free from obstructions in their respective path spaces and contain their own initial and final path conditions (removal of constraints).

d) Transcribe the modified optimal control problem to a nonlinear programming problem with the *active weights and control points* as the decision variables.

Starting from the optimal control problem (4.0.1)-(4.0.3), we partition the constraints into two groups — removable constraints and non-removable constraints. *Removable constraints* are those sets of path constraints that can be mutually satisfied by a union of overlapping polytopes. *Non-removable constraints* are simply those sets which are not removable.

$$\min_{\mathbf{x}(t),\mathbf{u}(t)} \quad \mathcal{F}_0(\mathbf{x}(t_0),\mathbf{u}(t_0)) + \int_{t_0}^{t_f} \mathcal{F}_t(\mathbf{x}(t),\mathbf{u}(t))\ dt + \ \mathcal{F}_f(\mathbf{x}(t_f),\mathbf{u}(t_f)) \tag{7.0.1}$$

subject to

$$\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t),\mathbf{u}(t)), \qquad t \in [t_0, t_f] \tag{7.0.2}$$

$$\begin{aligned}
\bar{\boldsymbol{\ell}}_0 \leq \quad & \bar{\mathbf{A}}_0\ \bar{\mathbf{x}}(t_0) + \bar{\mathbf{B}}_0\bar{\mathbf{u}}(t_0) && \leq \bar{\mathbf{u}}_0 && \\
\bar{\boldsymbol{\ell}}_t \leq \quad & \bar{\mathbf{A}}_t\ \bar{\mathbf{x}}(t) + \bar{\mathbf{B}}_t\bar{\mathbf{u}}(t) && \leq \bar{\mathbf{u}}_t, && t \in [t_0, t_f] \\
\bar{\boldsymbol{\ell}}_f \leq \quad & \bar{\mathbf{A}}_f\ \bar{\mathbf{x}}(t_f) + \bar{\mathbf{B}}_f\bar{\mathbf{u}}(t_f) && \leq \bar{\mathbf{u}}_f && \\
\bar{\mathbf{L}}_0 \leq \quad & \bar{\mathbf{c}}_0(\bar{\mathbf{x}}(t_0),\bar{\mathbf{u}}(t_0)) && \leq \bar{\mathbf{U}}_0 && \\
\bar{\mathbf{L}}_t \leq \quad & \bar{\mathbf{c}}_t(\bar{\mathbf{x}}(t),\bar{\mathbf{u}}(t)) && \leq \bar{\mathbf{U}}_t, && t \in [t_0, t_f] \\
\bar{\mathbf{L}}_f \leq \quad & \bar{\mathbf{c}}_f(\bar{\mathbf{x}}(t_f),\bar{\mathbf{u}}(t_f)) && \leq \bar{\mathbf{U}}_f &&
\end{aligned} \tag{7.0.3}$$

$$\begin{aligned}
\tilde{\boldsymbol{\ell}}_0 \leq \quad & \tilde{\mathbf{A}}_0\mathbf{x}(t_0) + \tilde{\mathbf{B}}_0\mathbf{u}(t_0) && \leq \tilde{\mathbf{u}}_0 && \\
\tilde{\boldsymbol{\ell}}_t \leq \quad & \tilde{\mathbf{A}}_t\mathbf{x}(t) + \tilde{\mathbf{B}}_t\mathbf{u}(t) && \leq \tilde{\mathbf{u}}_t, && t \in [t_0, t_f] \\
\tilde{\boldsymbol{\ell}}_f \leq \quad & \tilde{\mathbf{A}}_f\mathbf{x}(t_f) + \tilde{\mathbf{B}}_f\mathbf{u}(t_f) && \leq \tilde{\mathbf{u}}_f && \\
\tilde{\mathbf{L}}_0 \leq \quad & \tilde{\mathbf{c}}_0(\mathbf{x}(t_0),\mathbf{u}(t_0)) && \leq \tilde{\mathbf{U}}_0 && \\
\tilde{\mathbf{L}}_t \leq \quad & \tilde{\mathbf{c}}_t(\mathbf{x}(t),\mathbf{u}(t)) && \leq \tilde{\mathbf{U}}_t, && t \in [t_0, t_f] \\
\tilde{\mathbf{L}}_f \leq \quad & \tilde{\mathbf{c}}_f(\mathbf{x}(t_f),\mathbf{u}(t_f)) && \leq \tilde{\mathbf{U}}_f &&
\end{aligned} \tag{7.0.4}$$

Let the state and input evolutions be described by the mappings $\mathbf{x} : [t_0, t_f] \to \mathcal{X} \subset \mathbb{R}^n$ and $\mathbf{u} : [t_0, t_f] \to \mathcal{U} \subset \mathbb{R}^m$. In addition, assume the cost functional (7.0.1), dynamic constraints (7.0.2), and trajectory and actuator constraints (7.0.3)–(7.0.4) to be sufficiently smooth. In addition, the cost functional is expressed as a sum of three terms (i.e., initial, trajectory and final). Each function $\mathcal{F}_\ell$, $\ell \in \{0, t, f\}$ in the cost functional is a scalar-valued function $\mathcal{F}_\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$. The linear and nonlinear constraints are also divided into three terms (i.e., initial, trajectory, and final), and they are vector-valued functions $\bar{\mathcal{L}}_\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{\bar{N}^l_\ell}$, $\tilde{\mathcal{L}}_\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{N^l_\ell - \bar{N}^l_\ell}$, $\bar{c}_\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{\bar{N}^n_\ell}$ and $\tilde{c}_\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{N^n_\ell - \bar{N}^n_\ell}$, $\ell \in \{0, t, f\}$. Each of these sets of constraints is allowed to be set to equality, one-sided bounded, two-sided bounded, or unbounded by manipulating the values of the lower and upper bounds.

7.1 Removal of Dynamic Constraints

As mentioned in the introduction, we are interested in a real-time implementation of trajectory generation. In particular, restricting to differentially flat systems has the computational advantage of simplifying the optimal control problem by removing the dynamic constraints from the problem. In addition, as determined by Fliess et al. [1995], for differentially flat systems there exists a set of *flat outputs* (equal in number to the inputs) such that all states and inputs can be determined from these outputs without integration.

Definition 7.1 (Differentially Flat Systems)**.** A system is said to be differentially flat if one can find a set of variables, called the flat outputs, such that the system is algebraic over the differential field generated by the set of flat outputs. That is, if the system has states $\mathbf{x} \in \mathbb{R}^n$ and inputs $\mathbf{u} \in \mathbb{R}^m$, then the system is differentially flat if and only if we can find outputs $\mathbf{y} \in \mathbb{R}^m$ of the form

$$\mathbf{y} = \mathcal{H}(\mathbf{x}, \mathbf{u}^{(0)}, \mathbf{u}^{(1)}, \ldots, , \mathbf{u}^{(r)}) \tag{7.1.1}$$

$$\text{such that} \tag{7.1.2}$$

$$\mathbf{x} = \mathcal{F}(\mathbf{y}^{(0)}, \mathbf{y}^{(1)}, \ldots, , \mathbf{y}^{(q)}) \tag{7.1.3}$$

$$\mathbf{u} = \mathcal{G}(\mathbf{y}^{(0)}, \mathbf{y}^{(1)}, \ldots, , \mathbf{y}^{(q)}). \tag{7.1.4}$$

That is, one is able to express the flat outputs in terms of the states, inputs, and at most rth time derivatives of the inputs in such a way that there is a bijection such that the states and inputs can also be expressed in terms of at most qth time derivatives of the flat outputs.

Although, in general, we do not have a methodology to determine which systems are differentially flat or a systematic way of computing the flat outputs, there exist results in the literature that let us answer these questions for some specific cases. In particular, differentially flat systems include controllable linear systems and nonlinear systems which are feedback linearizable either by static or dynamic feedback, Fliess et al. [1995]. In addition, other specific systems such as aircraft in forward flight and some classes of vertical take-off and landing (VTOL) aircraft have been determined to be differentially flat. In the end, even if the system is not differentially flat, many times one is able to approximate it by control-relevant models that are differentially flat.

Since the behavior of flat systems is determined by the flat outputs, we can plan trajectories in flat space and then map these to appropriate inputs. With this transformation, we then rewrite the original optimal control problem (7.0.1) – (7.0.4) and exclude the dynamic constraints. At this point, one is justified to inquire if there are any disadvantages in mapping the original set of states and inputs to the flat outputs. In general, one can expect that any singularities in the original space will

be mapped to the flat space, but since the mappings (7.1.1)–(7.1.4) are obtained through algebraic manipulation of the dynamic constraints there might be new singularities or parameter restrictions that will need to be addressed. Also, it becomes more difficult to reason about the original dynamics and constraints in flat space, and the degree of the polynomial pieces parameterizing the flat outputs are usually larger to accommodate the required smoothness. Thankfully, all of these issues can be addressed off-line.

7.2 Removal of Trajectory Constraints

In the previous section, we were able to express the states and inputs in terms of a set of flat outputs $\mathbf{y} : [t_0, t_f] \rightarrow \mathcal{R}^m$. This produced a significant reduction in the number of unknown variables (i.e., from $n + m$ to m). We restrict our search for minimizers to the modified optimal control problem (dynamic constraints have been removed) in the space of piecewise polynomial functions with a prescribed number of polynomial pieces, order, and smoothness. This is a finite-dimensional space and, as a consequence, we are able to express each member in the vector space in terms of a set of basis functions. In particular, we express each flat output in terms of a linear combination of NURBS basis functions. The dimension of the vector space of any curve so parameterized is determined by $N_c = \dim(\mathcal{P}_{\mathbf{b},o,s}) = N_p(o - (s + 1)) + (s + 1)$. We will make use of the local support property of NURBS curves, which ensures us that the computation of any value of the curve only requires the computation of at most $d + 1$ NURBS basis functions, to efficiently compute the value of the curve at any given point. This is a significant savings in computational time because in general $d + 1$ is a much lesser quantity that N_c.

We are now in a position to explore the removal of trajectory and/or actuator constraints from the optimal control problem (7.0.1) – (7.0.4). As mentioned before, in the definition of a parametric NURBS curve lying in $\mathbb{R}^d$, $d = \{2, 3\}$, there are two important structures being combined to generate it — NURBS basis functions and a union of overlapping polytopes obtained from the coefficients of the linear combination. According to the strong convex hull property of NURBS curves, each polynomial piece is guaranteed to be bounded by a corresponding polytope (i.e., the smallest convex set containing a subset of the coefficients of the linear combination or control points). This idea extends to the whole NURBS curve in which case the parametric curve is guaranteed to remain inside the union of a set of overlapping polytopes. Therefore, if the control points defining such polytopes happen to be fixed in space, and we manipulate the NURBS basis functions (after discretization they depend only on the weights), then we are able to obtain an infinite number of curves all lying inside the space delineated by the polytopes, regardless of the values of the weights. As a consequence,

if the polytopes delineate a section of space that is feasible with respect to some set of trajectory constraints, then the resulting parametric curve will also be feasible with respect to the same set of trajectory constraints.

We will make use of this property to separately treat the guidance and obstacle-avoidance problems as follows. We first design a section of space (in general, non-convex) that is feasible with respect to trajectory constraints and that contains the initial and final path conditions by judiciously placing the control points (e.g., to form a union of overlapping polytopes avoiding all obstacles). Then, we generate paths that minimize the original cost functional and satisfy the remaining constraints (i.e., omit the trajectory constraints), using the active weights and control points as decision variables (some of the control points having been fixed in the previous step). The key observation is that paths generated by solving the latter problem will be guaranteed to be contained inside the feasible region generated in the former phase (i.e., will be feasible with respect to trajectory constraints) without the need to explicit write the trajectory constraints into the optimal control problem and independent of the discretization. Since the resulting trajectory, if it exists, is guaranteed to remain inside the designed region, we effectively can remove these constraints from the optimal control problem.

One of the main challenges in the application of the above procedure resides on the construction of a feasible section of space with respect to trajectory constraints starting only from obstacle information and the initial and final path conditions. In general, the obstructed space is described by the union of the obstacles, $\mathcal{O}_i$. That is,

$$\mathcal{T} = \bigcup_{i=1}^{M} \mathcal{O}_i = \mathcal{O}_1 \vee \cdots \vee \mathcal{O}_M.$$

Using de Morgan's Law, we are able to express free space (the feasible set) as follows:

$$\mathcal{F} = \neg \bigcup_{i=1}^{M} \mathcal{O}_i = \bigcap_{i=1}^{M} \neg\mathcal{O}_i = \neg\mathcal{O}_1 \wedge \cdots \wedge \neg\mathcal{O}_M,$$

where the logic operators $\wedge$, $\vee$, and $\neg$ stand for "and", "or", and "not", respectively. For practical reasons, we will only consider obstacles that are basic closed semi-algebraic sets; in particular, those that are compact convex with nonempty interiors.

Definition 7.2 (Basic closed semi-algebraic set)**.** A *basic closed semi-algebraic* subset of $\mathbb{R}^n$ is a set of the form

$$\{\mathbf{x} \in \mathbb{R}^n \mid f_1(\mathbf{x}) \leq 0 \wedge \cdots \wedge f_m(\mathbf{x}) \leq 0\}, \tag{7.2.1}$$

where $f_1, \ldots, f_m \in \mathbb{R}[x_1, \ldots, x_n]$, where $\mathbb{R}[x_1, \ldots, x_n]$ denotes the collection of all polynomials in $x_1, \ldots, x_n$ with coefficients in $\mathbb{R}$.

Since we are restricting ourselves to basic closed semi-algebraic sets, we describe the ith obstacle in the form:

$$\mathcal{O}_i = \{\mathbf{x} \in \mathbb{R}^d \mid f_1^i(\mathbf{x}) \leq 0 \wedge \cdots \wedge f_{n_i}^i(\mathbf{x}) \leq 0\}.$$

The feasible set then becomes

$$\begin{aligned}
\mathcal{F} &= \bigcap_{i=1}^{M} \neg \mathcal{O}_i \\
&= \bigcap_{i=1}^{M} \neg \{\mathbf{x} \in \mathbb{R}^d \mid f_1^i(\mathbf{x}) \leq 0 \wedge \cdots \wedge f_{n_i}^i(\mathbf{x}) \leq 0\} \\
&= \bigcap_{i=1}^{M} \{\mathbf{x} \in \mathbb{R}^d \mid f_1^i(\mathbf{x}) > 0 \vee \cdots \vee f_{n_i}^i(\mathbf{x}) > 0\} \\
&= \bigcap_{i=1}^{M} \{\mathbf{x} \in \mathbb{R}^d \mid \bigcup_{j=1}^{n_i} f_j^i(\mathbf{x}) > 0\}.
\end{aligned}$$

That is, the feasible set, in general, is a conjunction of disjunctions. This also elucidates the wide spread use of ellipses for approximating obstacles in the literature. Since in optimization-based approaches the feasible set must be specified in terms of intersection of sets, then it is desirable to approximate the set $\bigcup_{j=1}^{n_i} f_j^i(\mathbf{x}) > 0$ by a single polynomial inequality, rendering the feasible set to be a conjunction. However, the outcome of such approximations is unpredictable conservatism. In addition, one still has to include a large number of nonlinear constraints into the optimal control problem. Since in obstacle avoidance problems the bulk of the constraints are used to describe the feasible set, the inclusion of these many nonlinear constraint inequalities can render the problem real-time intractable. Alternatively, one can use boolean variables to enforce disjunctions. That is,

$$\left(\bigcap_{j=1}^{n_i} f_j^i(\mathbf{x}) > M_i \; b_j\right) \bigcap \left(\sum_{j=1}^{n_i} b_j \leq n_i - 1, b_j \in \{0,1\}\right) \implies \bigcup_{j=1}^{n_i} f_j^i(\mathbf{x}) > 0. \tag{7.2.2}$$

Note that if at least one of the boolean variables is 0, then at least one of the original inequalities is true. In this case, a boolean variable is required per inequality and, depending on how many inequalities are involved in describing an obstacle, this could be a very large number of extra decision variables that one must determine. In this case, the optimal control problem after transcription becomes a mixed-integer nonlinear programming problem with a large set of mixed-integer nonlinear constraints and a large set of extra boolean variables. As a result, the problem is very computationally expensive and real-time intractable.

Example 7.1. Consider an obstructed set which is made up of a single obstacle, $\mathcal{O}$. In particular, let the obstacle be defined by the following non-empty compact convex semi-algebraic set:

$$\mathcal{O} = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 10 \wedge 0.5\ x_1^2 + 0.25\ x_2^4 \leq 10 \wedge x_1 + x_2 \leq 3\}.$$

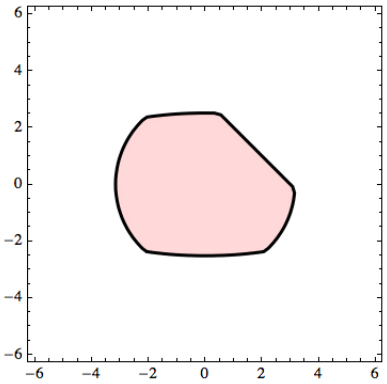

Figure 7.1: Obstructed Space

The feasible set is correspondingly described as its complement using de Morgan's Law.

$$\begin{aligned}
\mathcal{F} &= \{\mathbf{x} \in \mathbb{R}^2 \mid \neg\left(x_1^2 + x_2^2 \leq 10 \wedge 0.5\ x_1^2 + 0.25\ x_2^4 \leq 10 \wedge x_1 + x_2 \leq 3\right)\} \\
&= \{\mathbf{x} \in \mathbb{R}^2 \mid \neg\left(x_1^2 + x_2^2 \leq 10\right) \vee \neg\left(0.5\ x_1^2 + 0.25\ x_2^4 \leq 10\right) \vee \neg\left(x_1 + x_2 \leq 3\right)\} \\
&= \{\mathbf{x} \in \mathbb{R}^2 \mid x_1^2 + x_2^2 > 10 \vee 0.5\ x_1^2 + 0.25\ x_2^4 > 10 \vee x_1 + x_2 > 3\}.
\end{aligned}$$

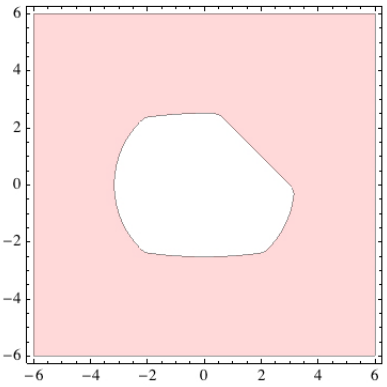

Figure 7.2: Free Space

In particular, note the the feasible set is described in terms of a disjunction.

Now that we have exposed the structure of the feasible set, we are in a position to appreciate the difficulties involved in constructing a feasible region of space by control point manipulation. In general, the choice of the vector space $\mathcal{P}_{\mathbf{b},o,s}$ containing the NURBS path will dictate the number of control points available for the construction of a section of space that is feasible with respect to trajectory constraints. In this specification, as noted in Chapter 6, is encoded the ordering of the control points, their subset partition (i.e., selection of $d+1$ points at a time) for putting together N_p polytopes in such a way that they share adjacently $s+1$ points. In addition, we require that the first and last control points correspond to the coordinates of the initial and final path conditions, respectively, and that these controls be vertices of their respective polytopes.

In particular, the construction of such a region without any further restrictions is complicated by the following issues: a) lack of uniqueness of a feasible region unless it is a singleton (in general, there exist many feasible regions connecting the initial and final path conditions, see Figure 7.3), b) must guarantee that the smallest convex sets containing the relevant control points per polynomial piece are feasible with respect to the constraints (the smallest convex set of a set of points is, by definition, the convex hull of them, and there is no analytical way of expressing this structure ahead of time), c) the feasible set is described by a conjunction of disjunctions, and d) it is not clear what objective function to use to drive a desired structure. Consequently, we are forced to determine a pragmatic set of conditions under which it is possible to construct feasible regions reliably.

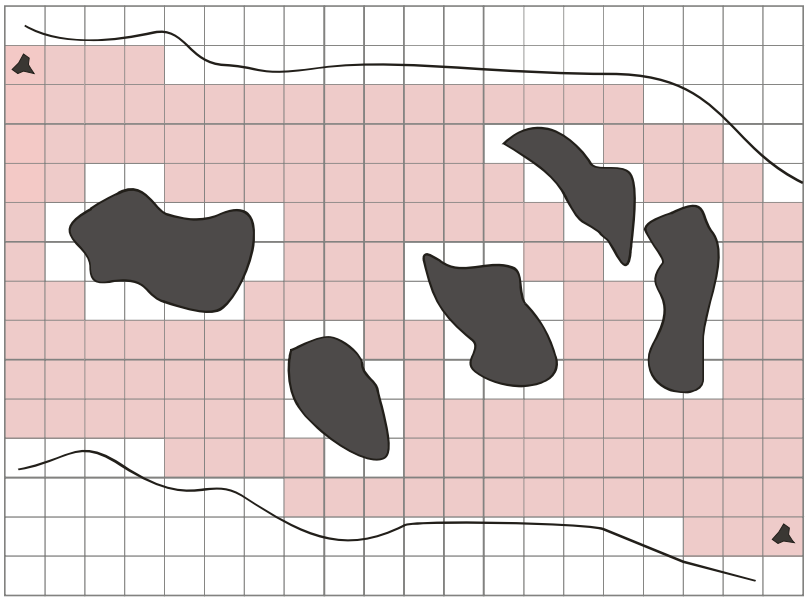

Figure 7.3: Cell Decomposition of a UAV trajectory space

We address such problems by requiring that a section of space, feasible with respect to trajectory constraints, be given in the form of an ordered union of pairwise adjacently overlapping non-empty convex compact sets, where the first set contains the initial path condition, and the last set contains the final path condition. These requirements remove the uniqueness problem because a specific direction is chosen before hand. This is, in practical terms, the way it occurs in many autonomous systems applications where a single safe corridor may be constructed from sensor data in the direction of the moving system. In the case of an autonomous system traversing an urban environment, one can exploit the structure of the streets and buildings to build safe regions. More structure than this is required, however, to build the overlapping set of polytopes corresponding to the NURBS parameterization. In particular, as mentioned above, we require that the original union of safe regions overlap only pairwise adjacently to avoid ill conditions, that they be ordered, and that they contain the initial and final path conditions. Since we will proceed this design by solving for trajectories which must satisfy the modified optimal control problem, it makes sense that we would like to design the largest region of space contained inside the given ordered union of pairwise adjacently overlapping safe regions to increase the likelihood of finding such a trajectory. One way in which this can be accomplished is by maximizing the sum of all the d-volumes of the approximating polytopes. This is equivalent to maximizing the volume of the ordered union of pairwise adjacently overlapping polytopes and their pairwise intersection, Theorem 2.30. In general, only non-empty convex compact sets have non-zero volume, and they are closed with respect to a finite number of intersections. Moreover, since these sets are convex, then we can also make use of Theorem 2.1 to ensure that all the computed polytopes lie strictly in the given union of overlapping safe regions without requiring an analytical description of the convex hull. As a final remark, one can exploit the fact that there exists more than one feasible region connecting the initial and final path conditions for those cases where one fails to acquire a trajectory satisfying the remaining constraints in the modified optimal control problem. In that occurrence, one can attempt to build a polytopal corridor that inner approximates on of the remaining feasible regions.

7.2.1 Inner Approximation of Feasible Regions

Mathematically, our objective is to develop an optimal inner approximation of a region $\mathcal{R}$ that is feasible with respect to trajectory and/or actuator constraints. The inner approximation $\mathcal{I}$ of $\mathcal{R}$ will be defined as the ordered union of pairwise adjacently overlapping polytopes, $\mathcal{P}_i$, $i = 1, \ldots, N_p$

$$\mathcal{I} = \left(\bigcup_{i=1}^{N_p} \mathcal{P}_i \right) \subseteq \mathcal{R}. \tag{7.2.3}$$

The purpose is then to place the control points making up each of the polytopes $\mathcal{P}_i, i = 1, \ldots, N_p$ in

such a way that the resulting union lies entirely in the feasible region $\mathcal{R}$, and it is the one with largest volume. Figure 7.4 illustrates the specification of a region $\mathcal{R}$ and a possible inner approximation of this region using five polytopes. Each of the polytopes are the convex hull of eight control points, and adjacent polytopes share three of these control points to ensure that the resulting curve (not shown) is at least C^2. In general, the choice of vector space $\mathcal{P}_{\mathbf{b},o,s}$ containing the desired NURBS curve will dictate the specific approximation. Consequently, the set $\mathcal{I}$ is not unique, and the same region $\mathcal{R}$ may be inner approximated in many different ways.

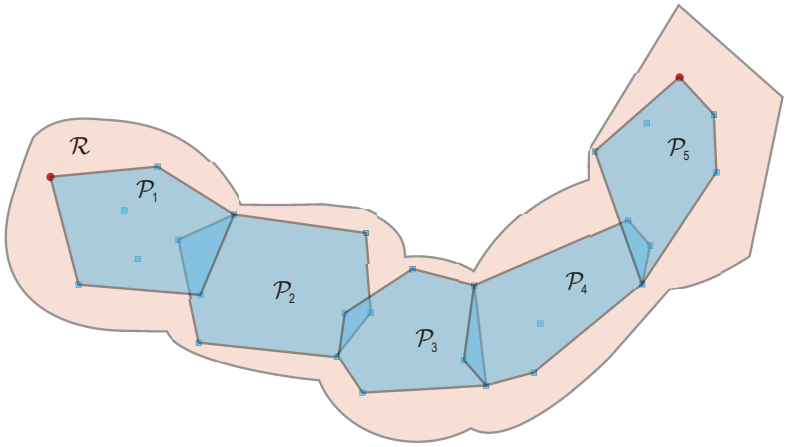

Figure 7.4: Inner approximation of region $\mathcal{R}$ by an ordered union of pairwise adjacently overlapping polytopes.

Theorem 7.1. Let $\mathcal{R}$ be a feasible region with respect to a set of trajectory constraints in $\mathbb{R}^d$, $d \in \{2, 3\}$. Assume that this region is defined by an ordered union of pairwise adjacently overlapping non-empty compact convex sets $\mathcal{S}_i$, $i = 1, \ldots, \mathcal{N}$ with non-zero volume overlap. That is,

$$\mathcal{R} = \bigcup_{i=1}^{\mathcal{N}} \mathcal{S}_i \text{ such that } v_d(\mathcal{S}_j \cap \mathcal{S}_{j+1}) \neq 0 \text{ for } j = 1, \ldots, \mathcal{N} - 1.$$

In addition, assume that the initial path condition, $\mathbf{c}(t_0)$, is contained in the interior of the set $\mathcal{S}_1$, and the final path condition, $\mathbf{c}(t_f)$, is contained in the interior of the set $\mathcal{S}_{\mathcal{N}}$. Furthermore, define a parametric NURBS curve, $\mathbf{c}(t)$ $t \in [t_0, t_f]$, lying in $\mathbb{R}^d$, $d \in \{2, 3\}$, respectively, and made up of $\mathcal{N}$ polynomial pieces with order o and sth continuously differentiable (s is not arbitrary; it has a lower and upper bound, $1 < s < d$). If we restrict the ordered set of control points arising from the specification of the parametric NURBS curve

$$\{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{N_c-1}\} = \bigcup_{i=1}^{\mathcal{N}} \{\mathbf{p}_{(i-1)(d-s)}, \ldots, \mathbf{p}_{i(d-s)+s}\},\ \mathbf{p}_k \in \mathbb{R}^d\ \forall\ k,\ N_c = \mathcal{N}(o - (s+1)) + (s+1)$$

such that $\{\mathbf{p}_{(i-1)(d-s)}, \ldots, \mathbf{p}_{i(d-s)+s}\} \in \mathcal{S}_i,\ i = 1, \ldots, \mathcal{N}$ and $\{\mathbf{p}_{(i-1)(d-s)}, \ldots, \mathbf{p}_{i(d-s)+s}\} \in \mathcal{S}_{j+1} \cap \mathcal{S}_j, j = 1, \ldots, \mathcal{N}-1$, and if the nonlinear programming problem $NLP(\mathcal{P}_1, \ldots, \mathcal{P}_\mathcal{N})$,

$$\begin{aligned}
\min -v_d(\mathcal{P}_1 \cup \cdots \cup \mathcal{P}_\mathcal{N}) - v_d(\mathcal{P}_1 \cap \mathcal{P}_2) - \quad & \ldots \quad -v_d(\mathcal{P}_{\mathcal{N}-1} \cap \mathcal{P}_\mathcal{N}) \\
\text{subject to} & \\
\mathbf{p}_0 &= \mathbf{c}(t_0) \in \mathcal{S}_i,\ i = 1 \\
\{\mathbf{p}_{(i-1)(d-s)+1}, \ldots, \mathbf{p}_{i(d-s)+s}\} &\in \tilde{\mathcal{S}}_i \subset \mathcal{S}_i,\ i = 1 \\
\{\mathbf{p}_{(i-1)(d-s)}, \ldots, \mathbf{p}_{i(d-s)+s}\} &\in \mathcal{S}_i,\ i = 2, \ldots, \mathcal{N}-1 \\
\{\mathbf{p}_{(i-1)(d-s)}, \ldots, \mathbf{p}_{i(d-s)+s-1}\} &\in \tilde{\mathcal{S}}_i \subset \mathcal{S}_i,\ i = \mathcal{N} \\
\{\mathbf{p}_{j(d-s)}, \ldots, \mathbf{p}_{j(d-s)+s}\} &\in \mathcal{S}_j \cap \mathcal{S}_{j+1},\quad j = 1, \ldots \mathcal{N}-1 \\
\mathbf{p}_{N_c-1} &= \mathbf{c}(t_f) \in \mathcal{S}_i,\ i = \mathcal{N} \\
v_d(\mathcal{P}_i) &\neq 0,\ i = 1, \ldots, \mathcal{N}
\end{aligned}$$

has a solution, then we can determine an inner approximation $\mathcal{I}$ of region $\mathcal{R}$ in the form of an ordered union of pairwise adjacently overlapping polytopes $\mathcal{I} = \left(\bigcup_{i=1}^{\mathcal{N}} \mathcal{P}_i\right) \subseteq \mathcal{R}$, where

$$\mathcal{P}_i = \text{conv}\left(\{\mathbf{p}_{(i-1)(d-s)}, \ldots, \mathbf{p}_{i(d-s)+s}\}\right), i = 1, \ldots, \mathcal{N}.$$

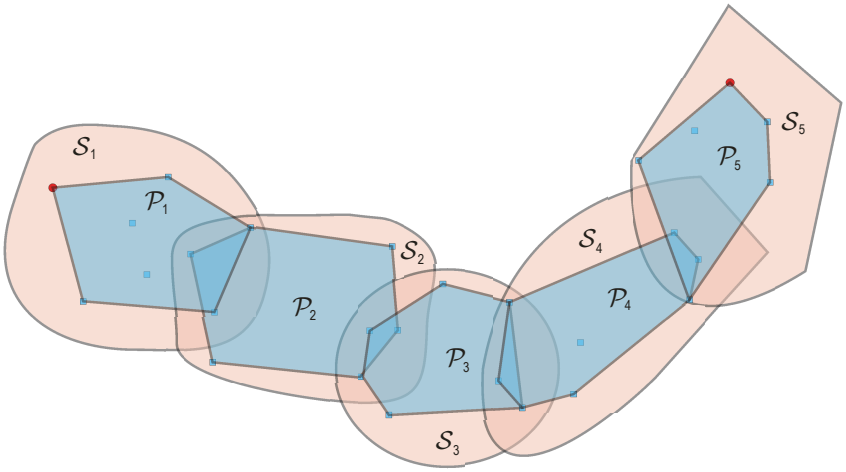

Figure 7.5: Specification of an ordered union of pairwise adjacently overlapping non-empty compact convex sets $\mathcal{S}_i$ and the resulting inner approximation by an ordered union of pairwise adjacently overlapping polytopes $\mathcal{P}_i$.

Proof. Assume that the nonlinear programming problem $NLP(\mathcal{P}_1,\ldots,\mathcal{P}_{\mathcal{N}})$ has a solution. Then, we have that a set of ordered control points $\{\mathbf{p}_0,\mathbf{p}_1,\ldots,\mathbf{p}_{N_c-1}\}$ exists and that they satisfy the condition $\{\mathbf{p}_{(i-1)(d-s)},\ldots,\mathbf{p}_{i(d-s)+s}\}\in\mathcal{S}_i,\ i=1,\ldots,\mathcal{N}$. Since the sets $\mathcal{S}_i,\ \ i=1,\ldots,\mathcal{N}$ are all convex, then we have that the polytopes $\mathcal{P}_i=\text{conv}\left(\{\mathbf{p}_{(i-1)(d-s)},\ldots,\mathbf{p}_{i(d-s)+s}\}\right),\ i=1,\ldots,\mathcal{N}$ lie, correspondingly, in $\mathcal{S}_i,\ i=1,\ldots,\mathcal{N}$ according to Theorem 2.1. Moreover, since the control points also satisfy $\{\mathbf{p}_{(i-1)(d-s)},\ldots,\mathbf{p}_{i(d-s)+s}\}\in\mathcal{S}_{j+1}\cap\mathcal{S}_j, j=1,\ldots,\mathcal{N}-1$, then, pairwise adjacently, the polytopes must overlap (according to the smoothness conditions on NURBS curves, they must share at least $s+1$ control points), although not necessarily with a non-zero volume. Since $\mathcal{P}_i\subseteq\mathcal{S}_i$, then if pairwise adjacent polytopes overlap they must do so in the intersection of the sets $\mathcal{S}_i$; that is, $\mathcal{P}_{j+1}\cap\mathcal{P}_j\in\mathcal{S}_{j+1}\cap\mathcal{S}_j, j=1,\ldots,\mathcal{N}_{-1}$. Consequently, $\left(\bigcup_{i=1}^{\mathcal{N}}\mathcal{P}_i\right)\subseteq\left(\bigcup_{i=1}^{\mathcal{N}}\mathcal{S}_i\right)$. Figure 7.5 illustrates the general idea behind the specification of region $\mathcal{R}$ and the resulting inner approximation by the set $\mathcal{I}$.

Finally, since the sets $\mathcal{S}_i$ are non-empty compact convex sets, they have non-zero volumes, and it makes sense to expand the ordered union of the pairwise adjacently overlapping polytopes in order to inner approximate the feasible region with the largest volume possible. Repeatedly applying Theorem 2.30, we are able to inductively prove that in general the sum of the d-volumes of $\mathcal{N}$ polytopes is determined as follows:

$$\begin{aligned} v_d(\mathcal{P}_1)+\cdots+v_d(\mathcal{P}_{\mathcal{N}}) &= v_d(\mathcal{P}_1\cup\cdots\cup\mathcal{P}_{\mathcal{N}})+ \\ &\quad v_d\left(\bigcup_{j=2}^{\mathcal{N}}(\mathcal{P}_1\cap\mathcal{P}_j)\right)+v_d\left(\bigcup_{j=3}^{\mathcal{N}}(\mathcal{P}_2\cap\mathcal{P}_j)\right)+\cdots+v_d(\mathcal{P}_{\mathcal{N}-1}\cap\mathcal{P}_{\mathcal{N}}). \end{aligned}$$

In particular, if the sets only overlap pairwise adjacently, we have that the sum of the d-volumes of the polytopes can be simplified to

$$v_d(\mathcal{P}_1)+\cdots+v_d(\mathcal{P}_{\mathcal{N}}) = v_d(\mathcal{P}_1\cup\cdots\cup\mathcal{P}_{\mathcal{N}})+v_d(\mathcal{P}_1\cap\mathcal{P}_2)+\cdots+v_d(\mathcal{P}_{\mathcal{N}-1}\cap\mathcal{P}_{\mathcal{N}}).$$

Therefore, using the above result and Theorem 2.47, we have that the cost function of the nonlinear programming problem $NLP(\mathcal{P}_1,\ldots,\mathcal{P}_{\mathcal{N}})$ that maximizes the volumes of the whole union of the polytopes and their pairwise adjacent intersections is equivalent to

$$\begin{aligned} v_d(\mathcal{P}_1\cup\cdots\cup\mathcal{P}_{\mathcal{N}})+v_d(\mathcal{P}_1\cap\mathcal{P}_2)+\cdots+v_d(\mathcal{P}_{\mathcal{N}-1}\cap\mathcal{P}_{\mathcal{N}}) &= v_d(\mathcal{P}_1)+\cdots+v_d(\mathcal{P}_{\mathcal{N}}) \\ &= \sum_{i_1=1}^{\mathcal{N}}\cdots\sum_{i_d=1}^{\mathcal{N}}\mathcal{V}(\mathcal{P}_{i_1},\ldots,\mathcal{P}_{i_d}), \end{aligned}$$

where $\mathcal{V}(\mathcal{P}_{i_1},\ldots,\mathcal{P}_{i_d})$ are the mixed volumes of polytopes $\mathcal{P}_{i_1},\ldots,\mathcal{P}_{i_d}$. In particular, according

to properties of mixed volumes (see Definition 2.25), the mixed volume of the convex polytopes $\mathcal{P}_1, \ldots, \mathcal{P}_d$ in $\mathbb{R}^d$, $\mathcal{V}(\mathcal{P}_1, \ldots, \mathcal{P}_d)$, is a non-negative continuous function in d variables on the set of convex polytopes and symmetric in the variables $\mathcal{P}_i$. As a consequence, it suffices to discard the case when any of the polytope volumes is zero to obtain an ordered union of pairwise adjacently overlapping polytopes that approximates $\mathcal{R}$ with volume $v_d(\mathcal{P}_1 \cup \cdots \cup \mathcal{P}_\mathcal{N}) + v_d(\mathcal{P}_1 \cap \mathcal{P}_2) + \cdots + v_d(\mathcal{P}_{\mathcal{N}-1} \cap \mathcal{P}_\mathcal{N})$. □

In order to be efficient, we will restrict the search for the coordinates of the control points to the boxes that circumscribe the sets $\mathcal{S}_i$, $i = 1, \ldots, \mathcal{N}$ and $\mathcal{S}_{j+1} \cap \mathcal{S}_j$, $j = 1, \ldots, \mathcal{N}-1$. We will label these boxes by $\mathcal{B}_i$, $i = 1, \ldots, \mathcal{N}$ and $\tilde{\mathcal{B}}_j$, $j = 1, \ldots, \mathcal{N}-1$, respectively. Let the coordinates $x_j^i \in \left[(x_j^i)^\ell, (x_j^i)^u\right]$, $j = 1, \ldots, d$. In general, we define $p_i^\ell = \left((x_1^i)^\ell, \ldots, (x_d^i)^\ell\right)$ and $p_i^u = \left((x_1^i)^u, \ldots, (x_d^i)^u\right)$, $i = 1, \ldots, \mathcal{N}$. That is, we can add the following set of constraints to the the nonlinear programming problem $NLP(\mathcal{P}_1, \ldots, \mathcal{P}_\mathcal{N})$.

$$
\begin{aligned}
p_i^\ell \leq \quad & \{\mathbf{p}_{(i-1)(d-s)+1}, \ldots, \mathbf{p}_{i(d-s)+s}\} && \leq p_i^u, \quad i = 1 \\
p_i^\ell \leq \quad & \{\mathbf{p}_{(i-1)(d-s)}, \ldots, \mathbf{p}_{i(d-s)+s}\} && \leq p_i^u, \quad i = 2, \ldots, N_p - 1 \\
p_i^\ell \leq \quad & \{\mathbf{p}_{(i-1)(d-s)}, \ldots, \mathbf{p}_{i(d-s)+s-1}\} && \leq p_i^u, \quad i = N_p.
\end{aligned}
$$

In addition, we have deliberately left the initial and final path conditions outside of the set $\tilde{\mathcal{S}}_i \subset \mathcal{S}_i$ to ensure that they become vertices of the resulting union of polytopes. In general, one can use a previously generated version of the union of polytopes and modified coordinates of control points $\mathbf{p}_0$ and $\mathbf{p}_{\mathcal{N}-1}$ to accommodate other initial and final path conditions, so long as the control points remain vertices of their respective polytopes and in their corresponding sets $\mathcal{S}_i$.

For practical implementations, one can restrict the individual sets $\mathcal{S}_i$ to be only those that are compact convex semi-algebraic sets with nonempty interiors (e.g., the simplest sets being polytopes). The construction of such regions can be obtained using a cell decomposition method or constructed online using sensor information followed by some type of pairwise adjacent overlapping procedure. In particular, the simplest sets meeting the above criteria are polytopes, and we will restrict to these because they give rise to optimization problems that are computationally less expensive (linear constraints) than their nonlinear counterparts.

Consider the following application of Theorem 7.1. Assume that the we are given the sets shown in Figure 7.6 that meet the conditions specified in Theorem 7.1, with all the sets $\mathcal{S}_i$ being polytopes. Then, by implementing the optimization problem $NLP(\mathcal{P}_1, \ldots, \mathcal{P}_\mathcal{N})$ and solving it, we obtain the ordered union of pairwise adjacently overlapping polytopes as shown in Figure 7.6.

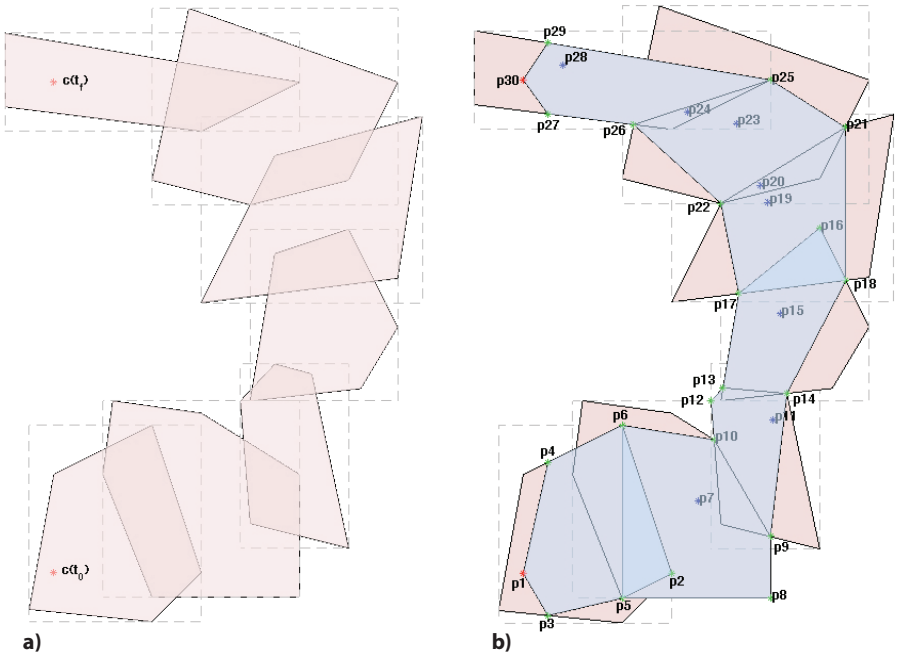

Figure 7.6: a) Specification of a feasible region in terms of a ordered union of adjacently overlapping non-empty compact convex sets and b) Inner approximation of feasible region by an ordered union of adjacently overlapping polytopes

After the removal of dynamic constraints, trajectory, and/or actuator constraints we have the following discretized optimal control problem:

$$\min_{\tilde{\mathbf{z}}} J[\tilde{\mathbf{z}}] \quad = \mathcal{G}_0(\tilde{\mathbf{z}}(\tau_0, \tilde{\mathbf{w}}, \tilde{\mathbf{p}})) \quad + \sum_{i=0}^{N_\tau - 2} \left[\sum_{j=0}^{N} \gamma_j^i \; \mathcal{G}_t(\tilde{\mathbf{z}}(r_j, \tilde{\mathbf{w}}, \tilde{\mathbf{p}})) \right] + \mathcal{G}_f(\tilde{\mathbf{z}}(\tau_{N_\tau - 1}, \tilde{\mathbf{w}}, \tilde{\mathbf{p}}))$$

subject to

$$\begin{aligned}
\tilde{\boldsymbol{\ell}}_0 \leq \quad & \tilde{\mathcal{A}}_0 \; \tilde{\mathbf{z}}(\tau_0, \tilde{\mathbf{w}}, \tilde{\mathbf{p}}) && \leq \tilde{\mathbf{u}}_0 \\
\tilde{\boldsymbol{\ell}}_t \leq \quad & \tilde{\mathcal{A}}_t \; \tilde{\mathbf{z}}(\tau_i, \tilde{\mathbf{w}}, \tilde{\mathbf{p}}) && \leq \tilde{\mathbf{u}}_t, \qquad i = 0, \ldots, N_\tau - 1 \\
\tilde{\boldsymbol{\ell}}_f \leq \quad & \tilde{\mathcal{A}}_f \; \tilde{\mathbf{z}}(\tau_{N_\tau - 1}, \tilde{\mathbf{w}}, \tilde{\mathbf{p}}) && \leq \tilde{\mathbf{u}}_f \\
\tilde{\mathbf{L}}_0 \leq \quad & \tilde{\mathcal{C}}_0(\tilde{\mathbf{z}}(\tau_0, \tilde{\mathbf{w}}, \tilde{\mathbf{p}})) && \leq \tilde{\mathbf{U}}_0 \\
\tilde{\mathbf{L}}_t \leq \quad & \tilde{\mathcal{C}}_t(\tilde{\mathbf{z}}(\tau_i, \tilde{\mathbf{w}}, \tilde{\mathbf{p}})) && \leq \tilde{\mathbf{U}}_t, \qquad i = 0, \ldots, N_\tau - 1 \\
\tilde{\mathbf{L}}_f \leq \quad & \tilde{\mathcal{C}}_f(\tilde{\mathbf{z}}(\tau_{N_\tau - 1}, \tilde{\mathbf{w}}, \tilde{\mathbf{p}})) && \leq \tilde{\mathbf{U}}_f.
\end{aligned}$$

Note that we have removed the dynamic constraints (7.0.2) and trajectory and/or actuator constraints (7.0.3) from the original optimal control problem.

7.3 Nonlinear Programming Problem

The result of the previous parameterization and discretization is a nonlinear programming problem where the unknowns are the active weights and control points of all the NURBS curves. The structure of a general nonlinear programming problem is as follows:

$$\begin{aligned} &\min_{\mathbf{y}} f(\mathbf{y}) \\ &\text{subject to} \\ &\mathbf{L} \leq \begin{bmatrix} \mathbf{y} \\ \mathbf{A}\mathbf{y} \\ \mathbf{c}(\mathbf{y}) \end{bmatrix} \leq \mathbf{U}. \end{aligned}$$

For our current setting, $\mathbf{y} \in \mathbb{R}^{\mathcal{N}_T^c}$, $\mathcal{N}_T^c$ is the total number of active decision variables, $f : \mathbb{R}^{\mathcal{N}_T^c} \to \mathbb{R}$ is a smooth real-valued function, $\mathbf{A} : \mathbb{R}^{\mathcal{N}_T^c} \to \mathbb{R}^{\tilde{N}_T^l = \tilde{N}_i^l + \tilde{N}_\tau \tilde{N}_t^l + \tilde{N}_f^l}$ is a linear operator, $\mathbf{c} : \mathbb{R}^{\mathcal{N}_T^c} \to \mathbb{R}^{\tilde{N}_T^n = \tilde{N}_i^n + \tilde{N}_\tau \tilde{N}_t^n + \tilde{N}_f^n}$ is a smooth vector-valued function, and the lower bounds and upper bounds are vectors $\mathbf{L}, \mathbf{U} \in \mathbb{R}^{\mathcal{N}_T^c + \tilde{N}_T^l + \tilde{N}_T^n}$, where $\tilde{N}_T^l$ and $\tilde{N}_T^n$ are the total number of linear and nonlinear constraints, respectively. We construct the objective function f, the linear operator $\mathbf{A}$, the constraint functions $\mathbf{c}$, and lower and upper bounds as follows:

$$f(\mathbf{y}) = \mathcal{G}_0(\tilde{\mathbf{z}}(\tau_0, \mathbf{y})) + \sum_{i=0}^{N_\tau - 2} \left[\sum_{j=0}^{N} \gamma_j^i \; \mathcal{G}_t(\tilde{\mathbf{z}}(r_j, \mathbf{y})) \right] + \mathcal{G}_f(\tilde{\mathbf{z}}(\tau_{N_\tau - 1}, \mathbf{y})) \tag{7.3.1}$$

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_c \\ \tilde{\boldsymbol{\ell}}_i \\ (\tilde{\boldsymbol{\ell}}_t)^0 \\ \vdots \\ (\tilde{\boldsymbol{\ell}}_t)_{N_c - 1} \\ \tilde{\boldsymbol{\ell}}_f \\ \tilde{\mathbf{L}}_i \\ (\tilde{\mathbf{L}}_t)_0 \\ \vdots \\ (\tilde{\mathbf{L}}_t)_{N_c - 1} \\ \tilde{\mathbf{L}}_f \end{bmatrix}, \quad \mathbf{c}(\mathbf{y}) = \begin{bmatrix} \tilde{\mathcal{A}}_0 \; \tilde{\mathbf{z}}(\tau_0, \mathbf{y}) \\ \tilde{\mathcal{A}}_t \; \tilde{\mathbf{z}}(\tau_0, \mathbf{y}) \\ \vdots \\ \tilde{\mathcal{A}}_t \; \tilde{\mathbf{z}}(\tau_{N_c - 1}, \mathbf{y}) \\ \tilde{\mathcal{A}}_f \; \tilde{\mathbf{z}}(\tau_{N_c - 1}, \mathbf{y}) \\ \tilde{\mathcal{C}}_0(\tilde{\mathbf{z}}(\tau_0, \mathbf{y})) \\ \tilde{\mathcal{C}}_t(\tilde{\mathbf{z}}(\tau_0, \mathbf{y})) \\ \vdots \\ \tilde{\mathcal{C}}_t(\tilde{\mathbf{z}}(\tau_{N_c - 1}, \mathbf{y})) \\ \tilde{\mathcal{C}}_f(\tilde{\mathbf{z}}(\tau_{N_c - 1}, \mathbf{y})) \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} \mathbf{U}_c \\ \tilde{\mathbf{u}}_i \\ (\tilde{\mathbf{u}}_t)_0 \\ \vdots \\ (\tilde{\mathbf{u}}_t)_{N_c - 1} \\ \tilde{\mathbf{u}}_f \\ \tilde{\mathbf{U}}_i \\ (\tilde{\mathbf{U}}_t)_0 \\ \vdots \\ (\tilde{\mathbf{U}}_t)_{N_c - 1} \\ \tilde{\mathbf{U}}_f \end{bmatrix}, \tag{7.3.2}$$

where $\mathbf{L}_c$ and $\mathbf{U}_c$ are the lower and upper bounds for the active decision variables. In general, we set the bounds for the active weights to be in the range $(0, \infty)$ and the active control points to be in the range $(-\infty, \infty)$. Note that it is no longer true that the linear constraints in the optimal

control problem map to linear constraints in the nonlinear programming problem. This is due to the nonlinear nature of the NURBS basis functions with respect to the weight parameters. The Jacobians of the objective function and constraint function are computed as follows:

$$\begin{aligned}\mathbf{D_y}f(\tilde{\mathbf{z}}(\mathbf{y})) &= \mathbf{D}_{\tilde{\mathbf{z}}}\mathcal{G}_0(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_0,\mathbf{y}) + \sum_{i=0}^{N_\tau-2}\left[\sum_{j=0}^{N}\gamma_j^i\ \mathbf{D}_{\tilde{\mathbf{z}}}\mathcal{G}_t(\tilde{\mathbf{z}}(r_j,\mathbf{y}))\ \mathbf{D_y}\tilde{\mathbf{z}}(r_j,\mathbf{y})\right] \\ &\quad +\mathbf{D}_{\tilde{\mathbf{z}}}\mathcal{G}_f(\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y}))\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y})\end{aligned}$$

$$\mathbf{D_y c}(\tilde{\mathbf{z}}(\mathbf{y})) = \begin{bmatrix} \tilde{\mathcal{A}}_0\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_0,\mathbf{y}) \\ \tilde{\mathcal{A}}_t\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_0,\mathbf{y}) \\ \vdots \\ \tilde{\mathcal{A}}_t\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}) \\ \tilde{\mathcal{A}}_f\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}) \\ \mathbf{D}_{\tilde{\mathbf{z}}}\tilde{\mathcal{C}}_0(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_0,\mathbf{y}) \\ \mathbf{D}_{\tilde{\mathbf{z}}}\tilde{\mathcal{C}}_t(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_0,\mathbf{y}) \\ \vdots \\ \mathbf{D}_{\tilde{\mathbf{z}}}\tilde{\mathcal{C}}_t(\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}))\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}) \\ \mathbf{D}_{\tilde{\mathbf{z}}}\tilde{\mathcal{C}}_f(\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}))\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}) \end{bmatrix}.$$

Likewise, the Hessian of the objective function and constraint function are computed as follows:

$$\begin{aligned}\mathbf{D_{yy}}f(\tilde{\mathbf{z}}(\mathbf{y})) &= \mathbf{D_y}\tilde{\mathbf{z}}(\tau_0,\mathbf{y})^T\mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\mathcal{G}_0(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_0,\mathbf{y}) + \sum_{i=1}^{\mathcal{N}_T^v}\frac{\partial\mathcal{G}_0(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))}{\partial\hat{z}_i}\ \mathbf{D_{yy}}\hat{z}_i \\ &\quad + \sum_{i=0}^{N_\tau-2}\left[\sum_{j=0}^{N}\gamma_j^i\ \mathbf{D_y}\tilde{\mathbf{z}}(r_j,\mathbf{y})^T\mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\mathcal{G}_t(\tilde{\mathbf{z}}(r_j,\mathbf{y}))\ \mathbf{D_y}\tilde{\mathbf{z}}(r_j,\mathbf{y})\right] \\ &\quad + \sum_{i=0}^{N_\tau-2}\left[\sum_{j=0}^{N}\gamma_j^i\ \left\{\sum_{k=1}^{\mathcal{N}_T^v}\frac{\partial\mathcal{G}_t(\tilde{\mathbf{z}}(r_j,\mathbf{y}))}{\partial\hat{z}_k}\ \mathbf{D_{yy}}\hat{z}_k\right\}\right] \\ &\quad +\mathbf{D_y}\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y})^T\mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\mathcal{G}_f(\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y}))\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y}) + \sum_{i=1}^{\mathcal{N}_T^v}\frac{\partial\mathcal{G}_f(\tilde{\mathbf{z}}(\tau_{N_\tau-1},\mathbf{y}))}{\partial\hat{z}_i}\ \mathbf{D_{yy}}\hat{z}_i\end{aligned}$$

$$\begin{aligned}\mathbf{D_{yy}}\left[\tilde{\mathcal{A}}_0^1\ \tilde{\mathbf{z}}(\tau_0,\mathbf{y})\right] &= \mathbf{D_y}\tilde{\mathbf{z}}(\tau_0,\mathbf{y})^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\left[\tilde{\mathcal{A}}_0^1\ \tilde{\mathbf{z}}(\tau_0,\mathbf{y})\right]\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_0,\mathbf{y}) + \sum_{i=1}^{\mathcal{N}_T^v}\frac{\partial\left[\tilde{\mathcal{A}}_0^1\ \tilde{\mathbf{z}}(\tau_0,\mathbf{y})\right]}{\partial\hat{z}_i}\ \mathbf{D_{yy}}\hat{z}_i \\ &\vdots \\ \mathbf{D_{yy}}\left[\tilde{\mathcal{A}}_0^{\tilde{N}_i^l}\ \tilde{\mathbf{z}}(\tau_0,\mathbf{y})\right] &= \mathbf{D_y}\tilde{\mathbf{z}}(\tau_0,\mathbf{y})^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\left[\tilde{\mathcal{A}}_0^{\tilde{N}_i^l}\ \tilde{\mathbf{z}}(\tau_0,\mathbf{y})\right]\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_0,\mathbf{y}) + \sum_{i=1}^{\mathcal{N}_T^v}\frac{\partial\left[\tilde{\mathcal{A}}_0^{\tilde{N}_i^l}\ \tilde{\mathbf{z}}(\tau_0,\mathbf{y})\right]}{\partial\hat{z}_i}\ \mathbf{D_{yy}}\hat{z}_i \\ \mathbf{D_{yy}}\left[\tilde{\mathcal{A}}_t^1\ \tilde{\mathbf{z}}(\tau_0,\mathbf{y})\right] &= \mathbf{D_y}\tilde{\mathbf{z}}(\tau_0,\mathbf{y})^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\left[\tilde{\mathcal{A}}_t^1\ \tilde{\mathbf{z}}(\tau_0,\mathbf{y})\right]\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_0,\mathbf{y}) + \sum_{i=1}^{\mathcal{N}_T^v}\frac{\partial\left[\tilde{\mathcal{A}}_t^1\ \tilde{\mathbf{z}}(\tau_0,\mathbf{y})\right]}{\partial\hat{z}_i}\ \mathbf{D_{yy}}\hat{z}_i \\ &\vdots \\ \mathbf{D_{yy}}\left[\tilde{\mathcal{A}}_t^{\tilde{N}_t^l}\ \tilde{\mathbf{z}}(\tau_0,\mathbf{y})\right] &= \mathbf{D_y}\tilde{\mathbf{z}}(\tau_0,\mathbf{y})^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\left[\tilde{\mathcal{A}}_t^{\tilde{N}_t^l}\ \tilde{\mathbf{z}}(\tau_0,\mathbf{y})\right]\ \mathbf{D_y}\tilde{\mathbf{z}}(\tau_0,\mathbf{y}) + \sum_{i=1}^{\mathcal{N}_T^v}\frac{\partial\left[\tilde{\mathcal{A}}_t^{\tilde{N}_t^l}\ \tilde{\mathbf{z}}(\tau_0,\mathbf{y})\right]}{\partial\hat{z}_i}\ \mathbf{D_{yy}}\hat{z}_i\end{aligned}$$

$$
\begin{aligned}
&\vdots \\
\mathbf{D}_{\mathbf{yy}}\left[\tilde{\mathcal{A}}_t^1\ \tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})\right] &= \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\left[\tilde{\mathcal{A}}_t^1\ \tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})\right]\ \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}) \\
&\quad + \sum_{i=1}^{\mathcal{N}_T^v} \frac{\partial\left[\tilde{\mathcal{A}}_t^1\ \tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})\right]}{\partial\hat{z}_i}\ \mathbf{D}_{\mathbf{yy}}\hat{z}_i \\
&\vdots \\
\mathbf{D}_{\mathbf{yy}}\left[\tilde{\mathcal{A}}_t^{\tilde{N}_t^l}\ \tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})\right] &= \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\left[\tilde{\mathcal{A}}_t^{\tilde{N}_t^l}\ \tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})\right]\ \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}) \\
&\quad + \sum_{i=1}^{\mathcal{N}_T^v} \frac{\partial\left[\tilde{\mathcal{A}}_t^{\tilde{N}_t^l}\ \tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})\right]}{\partial\hat{z}_i}\ \mathbf{D}_{\mathbf{yy}}\hat{z}_i \\
\mathbf{D}_{\mathbf{yy}}\left[\tilde{\mathcal{A}}_f^1\ \tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})\right] &= \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\left[\tilde{\mathcal{A}}_f^1\ \tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})\right]\ \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}) \\
&\quad + \sum_{i=1}^{\mathcal{N}_T^v} \frac{\partial\left[\tilde{\mathcal{A}}_f^1\ \tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})\right]}{\partial\hat{z}_i}\ \mathbf{D}_{\mathbf{yy}}\hat{z}_i \\
&\vdots \\
\mathbf{D}_{\mathbf{yy}}\left[\tilde{\mathcal{A}}_f^{\tilde{N}_f^l}\ \tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})\right] &= \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\left[\tilde{\mathcal{A}}_f^{\tilde{N}_f^l}\ \tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}))\right]\ \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}) \\
&\quad + \sum_{i=1}^{\mathcal{N}_T^v} \frac{\partial\left[\tilde{\mathcal{A}}_t^{\tilde{N}_f^l}\ \tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})\right]}{\partial\hat{z}_i}\ \mathbf{D}_{\mathbf{yy}}\hat{z}_i \\
\mathbf{D}_{\mathbf{yy}}\tilde{\mathcal{C}}_0^1(\tilde{\mathbf{z}}(\tau_0,\mathbf{y})) &= \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_0,\mathbf{y})^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\tilde{\mathcal{C}}_0^1(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))\ \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_0,\mathbf{y}) + \sum_{i=1}^{\mathcal{N}_T^v} \frac{\partial\tilde{\mathcal{C}}_0^1(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))}{\partial\hat{z}_i}\ \mathbf{D}_{\mathbf{yy}}\hat{z}_i \\
&\vdots \\
\mathbf{D}_{\mathbf{yy}}\tilde{\mathcal{C}}_0^{\tilde{N}_i^n}(\tilde{\mathbf{z}}(\tau_0,\mathbf{y})) &= \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_0,\mathbf{y})^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\tilde{\mathcal{C}}_0^{\tilde{N}_i^n}(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))\ \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_0,\mathbf{y}) + \sum_{i=1}^{\mathcal{N}_T^v} \frac{\partial\tilde{\mathcal{C}}_0^{\tilde{N}_i^n}(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))}{\partial\hat{z}_i}\ \mathbf{D}_{\mathbf{yy}}\hat{z}_i \\
\mathbf{D}_{\mathbf{yy}}\tilde{\mathcal{C}}_t^1(\tilde{\mathbf{z}}(\tau_0,\mathbf{y})) &= \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_0,\mathbf{y})^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\tilde{\mathcal{C}}_t^1(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))\ \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_0,\mathbf{y}) + \sum_{i=1}^{\mathcal{N}_T^v} \frac{\partial\tilde{\mathcal{C}}_t^1(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))}{\partial\hat{z}_i}\ \mathbf{D}_{\mathbf{yy}}\hat{z}_i \\
&\vdots \\
\mathbf{D}_{\mathbf{yy}}\tilde{\mathcal{C}}_t^{\tilde{N}_t^n}(\tilde{\mathbf{z}}(\tau_0,\mathbf{y})) &= \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_0,\mathbf{y})^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\tilde{\mathcal{C}}_t^{\tilde{N}_t^n}(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))\ \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_0,\mathbf{y}) + \sum_{i=1}^{\mathcal{N}_T^v} \frac{\partial\tilde{\mathcal{C}}_t^{\tilde{N}_t^n}(\tilde{\mathbf{z}}(\tau_0,\mathbf{y}))}{\partial\hat{z}_i}\ \mathbf{D}_{\mathbf{yy}}\hat{z}_i \\
&\vdots \\
\mathbf{D}_{\mathbf{yy}}\tilde{\mathcal{C}}_t^1(\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})) &= \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\tilde{\mathcal{C}}_t^1(\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}))\ \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}) \\
&\quad + \sum_{i=1}^{\mathcal{N}_T^v} \frac{\partial\tilde{\mathcal{C}}_t^1(\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}))}{\partial\hat{z}_i}\ \mathbf{D}_{\mathbf{yy}}\hat{z}_i \\
&\vdots \\
\mathbf{D}_{\mathbf{yy}}\tilde{\mathcal{C}}_t^{\tilde{N}_t^n}(\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})) &= \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\tilde{\mathcal{C}}_t^{\tilde{N}_t^n}(\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}))\ \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}) \\
&\quad + \sum_{i=1}^{\mathcal{N}_T^v} \frac{\partial\tilde{\mathcal{C}}_t^{\tilde{N}_t^n}(\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}))}{\partial\hat{z}_i}\ \mathbf{D}_{\mathbf{yy}}\hat{z}_i
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{D}_{\mathbf{yy}}\tilde{\mathcal{C}}_f^1(\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})) &= \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\tilde{\mathcal{C}}_f^1(\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}))\ \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}) \\
&\quad + \sum_{i=1}^{\mathcal{N}_T^v} \frac{\partial\tilde{\mathcal{C}}_f^1(\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}))}{\partial\hat{z}_i}\ \mathbf{D}_{\mathbf{yy}}\hat{z}_i \\
&\vdots \\
\mathbf{D}_{\mathbf{yy}}\tilde{\mathcal{C}}_f^{\tilde{N}_f^n}(\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})) &= \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y})^T\ \mathbf{D}_{\tilde{\mathbf{z}}\tilde{\mathbf{z}}}\tilde{\mathcal{C}}_f^{\tilde{N}_f^n}(\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}))\ \mathbf{D}_{\mathbf{y}}\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}) \\
&\quad + \sum_{i=1}^{\mathcal{N}_T^v} \frac{\partial\tilde{\mathcal{C}}_t^{\tilde{N}_f^n}(\tilde{\mathbf{z}}(\tau_{N_c-1},\mathbf{y}))}{\partial\hat{z}_i}\ \mathbf{D}_{\mathbf{yy}}\hat{z}_i.
\end{aligned}
$$

The Jacobian and Hessian of the z variables with respect to the active weights and control points are obtained by using Proposition 3.2 and by ordering both the z variables and the active decision variables as described in Chapter 4. Before delving into some examples demonstrating the methodology described in this chapter, let us summarize the steps described above. Figure 7.7 illustrates the steps required after expressing the optimal control problem in terms of the flat outputs.

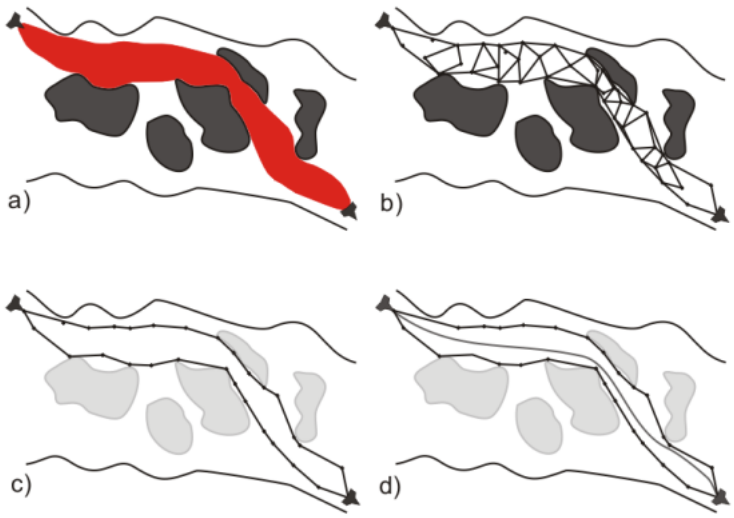

Figure 7.7: a) Feasible region specification, b) inner approximation of feasible region by an ordered union of adjacently overlapping polytopes, c) removal of trajectory constraints, d) computation of trajectory

We begin with a specification of a feasible region meeting the requirements spelled out in Theorem 7.1, and then we solve the nonlinear programming problem $NLP(\mathcal{P}_1,\ldots,\mathcal{P}_{\mathcal{N}})$ arising from the specification of a parametric NURBS curve. The result is an ordered union of adjacently overlapping

polytopes. We then fixed these control points and prepare to solve a modified version of the original optimal control problem which, in addition to omitting dynamic constraints, also omits the trajectory constraints that are satisfied by the union of polytopes. The result is a minimizer to the original optimal control problem (7.0.1)–(7.0.4).

Example 7.2 (Analytical - Revisited using active weights and control points as decision variables)**.** At this point, we will revisit Example 5.1. Previously, we had shown how to use NURBS with control points only to compute optimal trajectories. Here, we will begin by comparing those results with the methodology of this chapter. That is, we will consider NURBS with active weights and control points.

$$x_i \in \mathcal{P}_{\mathbf{b}_i,o_i,s_i}, \quad i=1,\dots 3, \text{ with } \mathbf{b}_i = \begin{bmatrix} 0 & 3.3333 & 6.6667 & 10.0000 \end{bmatrix},\ o_i = 6,\ s_i = 2$$
$$u_j \in \mathcal{P}_{\mathbf{b}_j,o_j,s_j}, \quad j=1,\dots 2, \text{ with } \mathbf{b}_j = \begin{bmatrix} 0 & 2.5000 & 5.0000 & 7.5000 & 10.0000 \end{bmatrix}, o_j = 5,\ s_j = 0$$

Based on Proposition 3.2, a NURBS curve and its time derivatives are expressed recursively as follows:

$$c_i^{(r_i)}(t,\mathbf{w}_i,\mathbf{p}_i) = \frac{\mathbf{w}_i^T \text{diag}(\{\mathcal{B}_{d_i}^{(r_i)}(t)\})\mathbf{p}_i}{\left[\mathcal{B}_{d_i}^{(0)}(t)\right]^T \mathbf{w}_i} - \sum_{k=1}^{r_i} \begin{pmatrix} r_i \\ k \end{pmatrix} \frac{\left[\mathcal{B}_{d_i}^{(k)}(t)\right]^T \mathbf{w}_i}{\left[\mathcal{B}_{d_i}^{(0)}(t)\right]^T \mathbf{w}_i}\ c_i^{(r_i-k)}(t,\mathbf{w}_i,\mathbf{p}_i), \quad t\in[t_0,t_f]$$

where $\left[\mathcal{B}_{d_i}^{(r_i)}(t)\right] = \begin{bmatrix} \mathcal{B}_{0,d_i}^{(r_i)}(t) \\ \vdots \\ \mathcal{B}_{N_i^c-1,d_i}^{(r_i)}(t) \end{bmatrix}$, $\text{diag}(\{\mathcal{B}_{d_i}^{(r_i)}(t)\}) = \begin{bmatrix} \mathcal{B}_{0,d_i}^{(r_i)}(t) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \mathcal{B}_{N_i^c-1,d_i}^{(r_i)}(t) \end{bmatrix}$,

$\mathbf{w}_i = \begin{bmatrix} w_i^0 \\ \vdots \\ w_i^{N_i^c-1} \end{bmatrix}$, and $\mathbf{p}_i = \begin{bmatrix} p_i^0 \\ \vdots \\ p_i^{N_i^c-1} \end{bmatrix}$. More specifically, the states and inputs in the current example take the following form at a collocation point τ_k:

$$x_i^{(0)}(\tau_k,\mathbf{w}_i,\mathbf{p}_i) = \frac{\mathbf{w}_i^T \text{diag}(\{\mathcal{B}_{d_i}^{(0)}(\tau_k)\})\mathbf{p}_i}{\left[\mathcal{B}_{d_i}^{(0)}(\tau_k)\right]^T \mathbf{w}_i},\ i\in\{1,2,3\}$$

$$x_i^{(1)}(\tau_k,\mathbf{w}_i,\mathbf{p}_i) = \frac{\mathbf{w}_i^T \text{diag}(\{\mathcal{B}_{d_i}^{(1)}(\tau_k)\})\mathbf{p}_i}{\left[\mathcal{B}_{d_i}^{(0)}(\tau_k)\right]^T \mathbf{w}_i} - \frac{\left[\mathcal{B}_{d_i}^{(1)}(\tau_k)\right]^T \mathbf{w}_i}{\left(\left[\mathcal{B}_{d_i}^{(0)}(\tau_k)\right]^T \mathbf{w}_i\right)^2}\ \mathbf{w}_i^T \text{diag}(\{\mathcal{B}_{d_i}^{(0)}(\tau_k)\})\mathbf{p}_i,\ i\in\{1,2,3\}$$

$$u_j^{(0)}(\tau_k,\mathbf{w}_j,\mathbf{p}_j) = \frac{\mathbf{w}_j^T \text{diag}(\{\mathcal{B}_{d_j}^{(0)}(\tau_k)\})\mathbf{p}_j}{\left[\mathcal{B}_{d_j}^{(0)}(\tau_k)\right]^T \mathbf{w}_j},\ j\in\{1,2\}$$

Note that in this instance the curves depend nonlinearly on the parameters of the piecewise polynomial functions (i.e., active weights and control points). Figure 7.8 and Figure 7.9 compare the minimizer found using this methodology against the one found in Example 5.1.

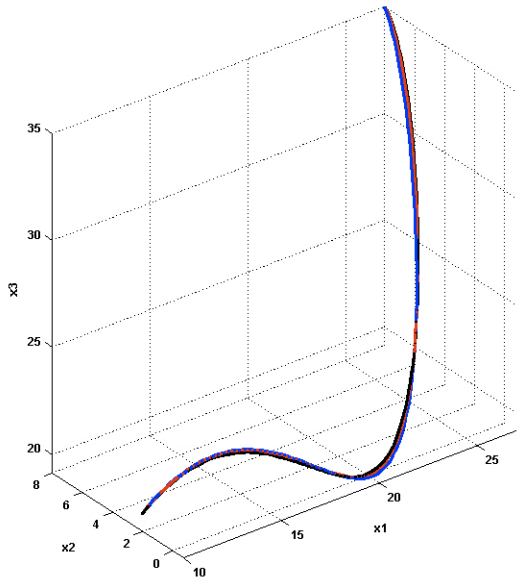

Figure 7.8: Numerical optimal state trajectory with control points only (black), Numerical optimal state trajectory with active weights and control points (red) and Analytical optimal state trajectory (blue)

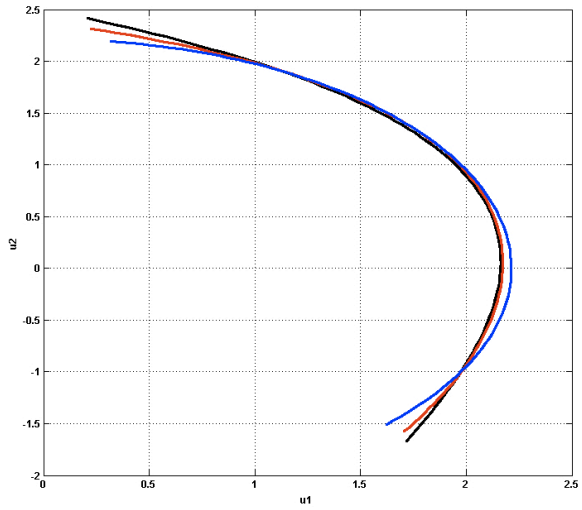

Figure 7.9: Numerical optimal input trajectory with control points only (black), Numerical optimal input trajectory with active weights and control points (red) and Analytical optimal input trajectory (blue)

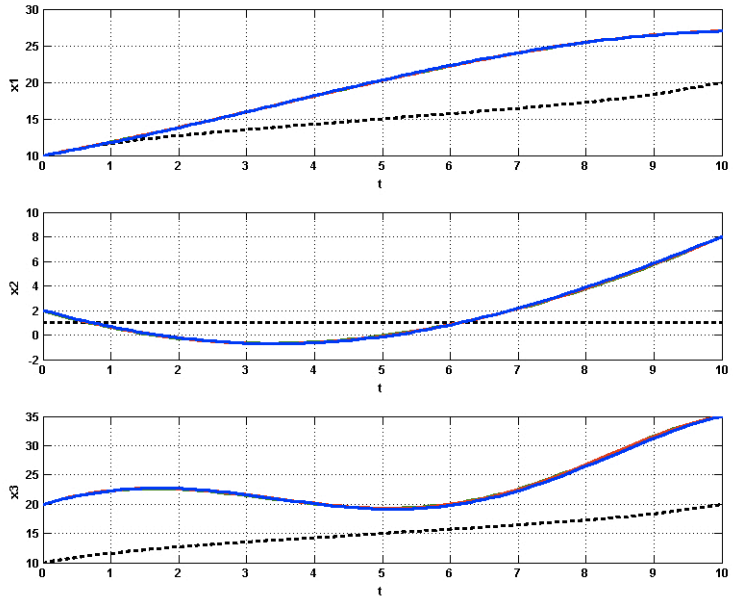

Figure 7.10: Initial guess for states (black), Numerical optimal states with control points only (green), Numerical optimal states with active weights and control points (red) and Analytical optimal states (blue)

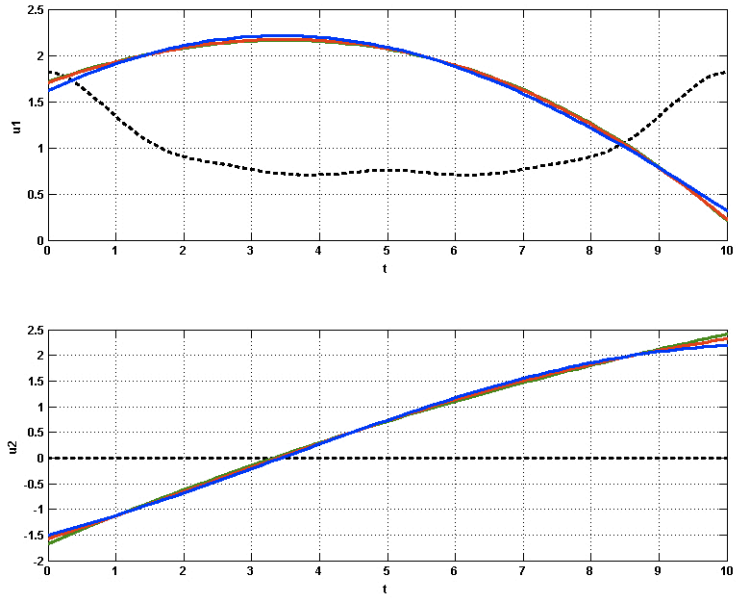

Figure 7.11: Initial guess for inputs (black), Numerical optimal inputs with control points only (green), Numerical optimal inputs with active weights and control points (red) and Analytical optimal inputs (blue)

Example 7.3 (Analytical - Revisited exploiting differential flatness and using active weights and control points)**.** It turns out, that this system is differentially flat. Consequently, we are able to map the states and inputs to a lower dimensional space, reducing the number of decision variables. Consider the following flat parameterization of this dynamical systems by letting $z_1 = x_1$ and $z_2 = x_3$ serve as flat outputs. In order for this parameterization to be flat we must be able to write the states and inputs in terms of the flat outputs and some of their derivatives. That is,

$$\begin{aligned} x_1 &= z_1 \\ x_2 &= \frac{\dot{z}_2}{\dot{z}_1} \\ x_3 &= z_2 \\ u_1 &= \dot{z}_1 \\ u_2 &= \frac{\dot{z}_1\ddot{z}_2 - \ddot{z}_1\dot{z}_2}{\dot{z}_1^2} \end{aligned}$$

Moreover, we should be able to write the flat output variables and their derivatives in terms of the states, the inputs and some derivatives of the inputs.

$$\begin{aligned} z_1 &= x_1 \\ \dot{z}_1 &= u_1 \\ \ddot{z}_1 &= \dot{u}_1 \\ z_2 &= x_3 \\ \dot{z}_2 &= u_1 x_2 \\ \ddot{z}_2 &= u_1 u_2 + \dot{u}_1 x_2 \end{aligned}$$

More succinctly, we are able to write,

$$(z_1, \ldots, \ddot{z}_1, z_2, \ldots, \ddot{z}_2) = \Psi(x_1, x_2, x_3, u_1, \dot{u}_1, u_2) = \Psi(\xi)$$

In particular, note that this mapping is locally invertible, with the exception of the following point:

$$\det\left(\mathbf{D}_\xi \Psi(\xi)\right) = -u_1^2$$

Consequently, we will need to avoid the singularity occurring at $u_1 = \dot{z}_1 = 0$. Specifically, we will constraint this input to be positive.

In terms of the flat outputs, the optimal control problem takes the form:

$$\min_{z_1,z_2} \quad \frac{1}{2}\int_{t_0=0}^{t_f=10} \left[\begin{array}{cc} \dot{z}_1 & \frac{\dot{z}_1\ddot{z}_2-\ddot{z}_1\dot{z}_2}{\dot{z}_1^2} \end{array}\right] \mathbf{R} \left[\begin{array}{c} \dot{z}_1 \\ \frac{\dot{z}_1\ddot{z}_2-\ddot{z}_1\dot{z}_2}{\dot{z}_1^2} \end{array}\right] dt$$

subject to:

$$\begin{aligned} z_1(t_0) &= 10 \\ \frac{\dot{z}_2(t_0)}{\dot{z}_1(t_0)} &= 2 \\ z_2(t_0) &= 20 \\ \dot{z}_1(t) &\in (0.25, \infty) \\ z_1(t_f) &= 27 \\ \frac{\dot{z}_2(t_f)}{\dot{z}_1(t_f)} &= 8 \\ z_2(t_f) &= 35 \end{aligned}$$

As before, we proceed to parameterize the flat outputs in terms of piecewise polynomial functions as a linear combination of NURBS basis functions. The specific parameterization proceed as follows:

$$z_i \in \mathcal{P}_{\mathbf{b}_i,o_i,s_i}, \quad i = 1,\ldots 2, \text{ with } \mathbf{b}_i = \left[\begin{array}{ccccc} 0 & 2.5000 & 5 & 7.5000 & 10 \end{array}\right], \; o_i = 6, \; s_i = 3$$

Using Proposition 3.2, a NURBS curve and its time derivatives are expressed in the current example in the following form at a collocation point τ_k:

$$\begin{aligned} z_i^{(0)}(\tau_k, \mathbf{w}_i, \mathbf{p}_i) &= \frac{\mathbf{w}_i^T \text{diag}(\{\mathcal{B}_{d_i}^{(0)}(\tau_k)\})\mathbf{p}_i}{\left[\mathcal{B}_{d_i}^{(0)}(\tau_k)\right]^T \mathbf{w}_i}, \; i \in \{1,2\} \\ z_i^{(1)}(\tau_k, \mathbf{w}_i, \mathbf{p}_i) &= \frac{\mathbf{w}_i^T \text{diag}(\{\mathcal{B}_{d_i}^{(1)}(\tau_k)\})\mathbf{p}_i}{\left[\mathcal{B}_{d_i}^{(0)}(\tau_k)\right]^T \mathbf{w}_i} - \frac{\left[\mathcal{B}_{d_i}^{(1)}(\tau_k)\right]^T \mathbf{w}_i}{\left(\left[\mathcal{B}_{d_i}^{(0)}(\tau_k)\right]^T \mathbf{w}_i\right)^2} \mathbf{w}_i^T \text{diag}(\{\mathcal{B}_{d_i}^{(0)}(\tau_k)\})\mathbf{p}_i, \; i \in \{1,2\} \\ z_i^{(2)}(\tau_k, \mathbf{w}_i, \mathbf{p}_i) &= \frac{\mathbf{w}_i^T \text{diag}(\{\mathcal{B}_{d_i}^{(2)}(\tau_k)\})\mathbf{p}_i}{\left[\mathcal{B}_{d_i}^{(0)}(\tau_k)\right]^T \mathbf{w}_i} - 2\frac{\left[\mathcal{B}_{d_i}^{(1)}(\tau_k)\right]^T \mathbf{w}_i}{\left(\left[\mathcal{B}_{d_i}^{(0)}(\tau_k)\right]^T \mathbf{w}_i\right)^2} \mathbf{w}_i^T \text{diag}(\{\mathcal{B}_{d_i}^{(1)}(\tau_k)\})\mathbf{p}_i + \\ &\quad \left(2\frac{\left(\left[\mathcal{B}_{d_i}^{(1)}(\tau_k)\right]^T \mathbf{w}_i\right)^2}{\left(\left[\mathcal{B}_{d_i}^{(0)}(\tau_k)\right]^T \mathbf{w}_i\right)^3} - \frac{\left[\mathcal{B}_{d_i}^{(2)}(\tau_k)\right]^T \mathbf{w}_i}{\left(\left[\mathcal{B}_{d_i}^{(0)}(\tau_k)\right]^T \mathbf{w}_i\right)^2}\right) \mathbf{w}_i^T \text{diag}(\{\mathcal{B}_{d_i}^{(0)}(\tau_k)\})\mathbf{p}_i, \; i \in \{1,2\} \end{aligned}$$

Figure 7.12 and Figure 7.13 compare the minimizer found by exploiting differential flatness to the

analytical optimal.

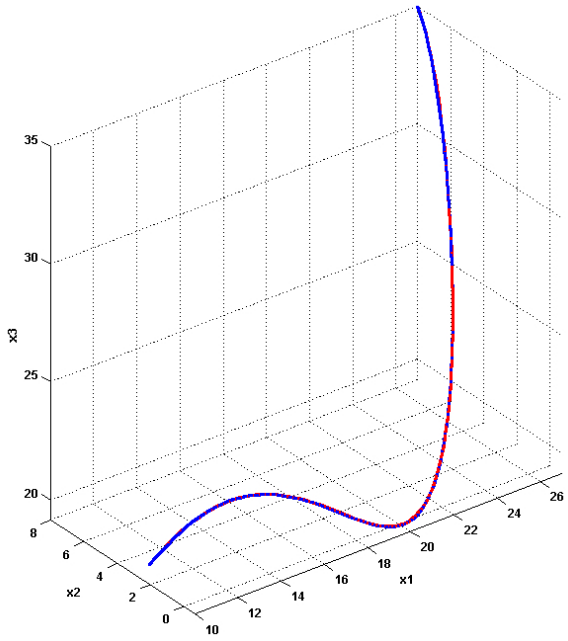

Figure 7.12: Numerical optimal state trajectory by exploiting differential flatness (red) and Analytical optimal state trajectory (blue)

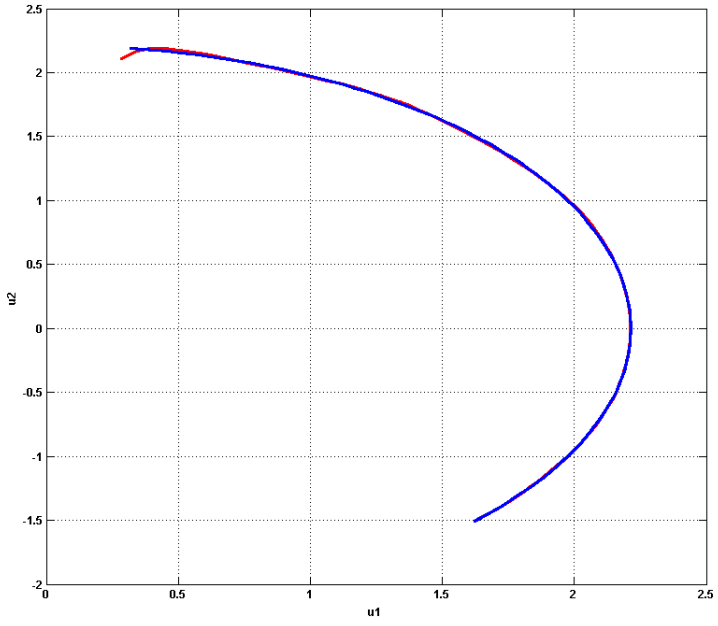

Figure 7.13: Numerical optimal input trajectory by exploiting differential flatness (red) and Analytical optimal input trajectory (blue)

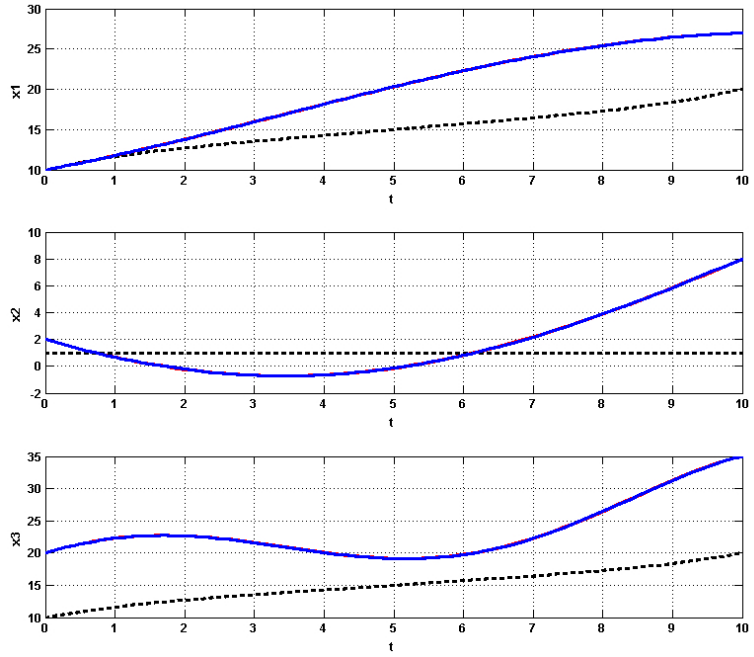

Figure 7.14: Initial guess for states (black), Numerical optimal states by exploiting differential flatness (red) and Analytical optimal states (blue)

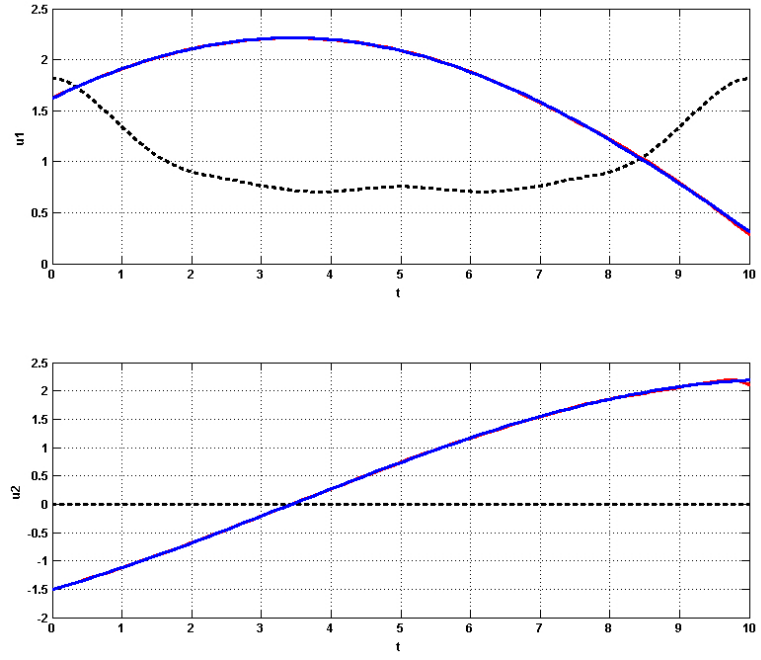

Figure 7.15: Initial guess for inputs (black), Numerical optimal inputs by exploiting differential flatness (red) and Analytical optimal inputs (blue)

Example 7.4 (Analytical-Revisited exploiting differential flatness and using active weights and control points subject to trajectory constraints)**.** Consider adding the following trajectory constraint: $(x_1-25)^2+(x_2-3)^2+(x_1-26)^2-25>0$. In terms of the flat variables, this trajectory constraint takes the following form: $(z_1-25)^2+(\frac{\dot{z}_2}{\dot{z}_1}-3)^2+(z_2-26)^2-25>0$. Note that this constraint is not in flat space because it also requires the derivatives of the flat outputs.

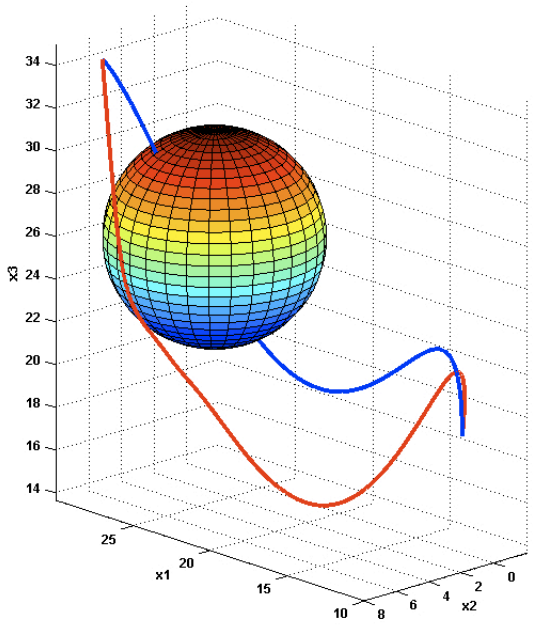

Figure 7.16: Constrained state trajectory (red) and Analytical optimal state trajectory (blue)

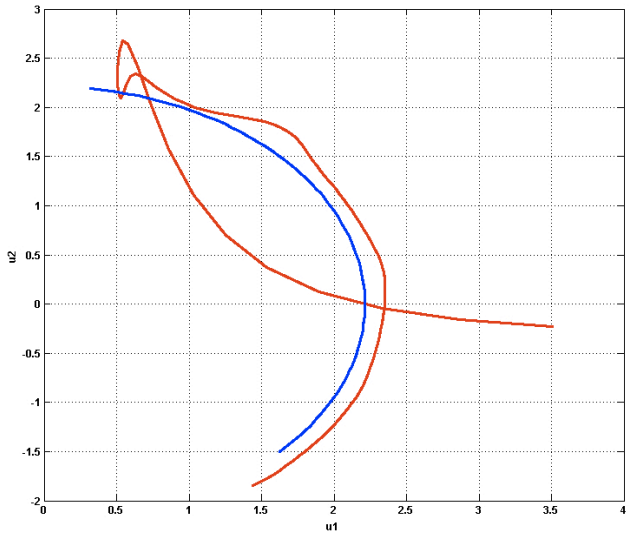

Figure 7.17: Constrained input trajectory (red) and Analytical optimal input trajectory (blue)

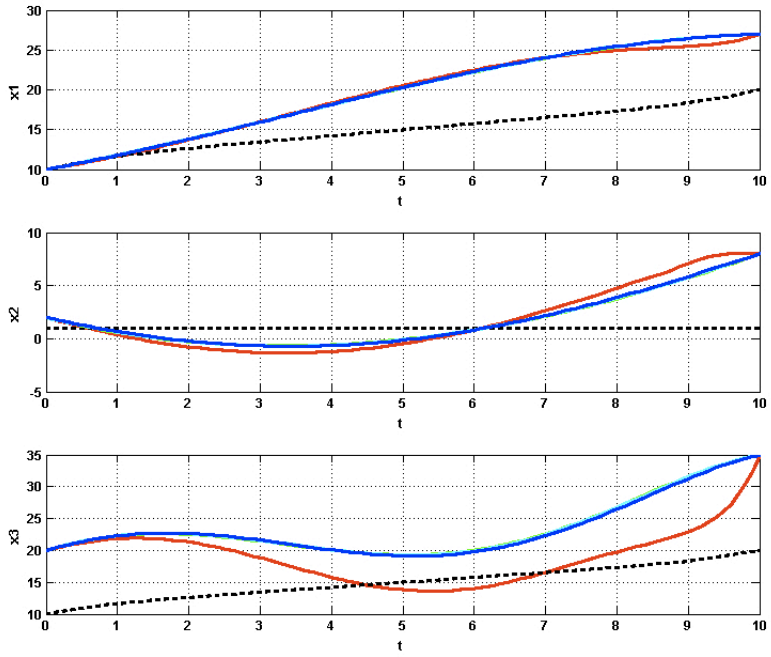

Figure 7.18: Initial guess for states (black), Constrained states by exploiting differential flatness (red) and Analytical optimal states (blue)

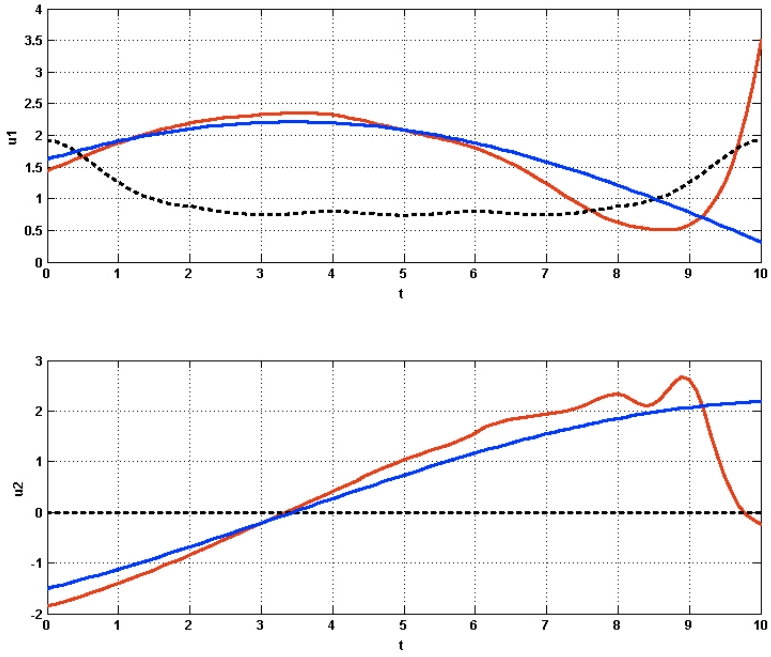

Figure 7.19: Initial guess for inputs (black), Constrained inputs by exploiting differential flatness (red) and Analytical optimal inputs (blue)

Example 7.5 (VTOL UAV)**.** In this example, we will use our approach to plan trajectories for a small four-propeller VTOL UAV.

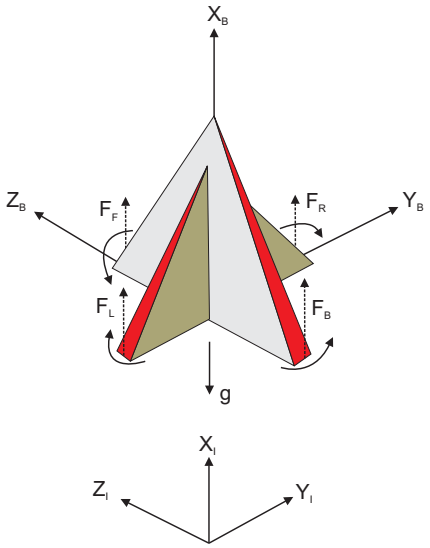

Figure 7.20: VTOL UAV Coordinate frames. Arrows around forces depict assumed direction of propeller rotation.

The translational and attitudinal equations describing the VTOL UAV motion are expressed as follows:

$$\begin{bmatrix} \ddot{x}+g \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \frac{F_L+F_R+F_F+F_B}{m^B} \begin{bmatrix} \cos\theta\ \cos\psi \\ \cos\theta\ \sin\psi \\ -\sin\theta \end{bmatrix} \tag{7.3.3}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{k_w}{I_x}(F_L+F_R-F_B-F_F)+\frac{(I_y-I_z)}{I_x}\ q\ r \\ \frac{\ell_{BF}}{I_y}(F_F-F_B)+\frac{(I_z-I_x)}{I_y}\ p\ r \\ \frac{\ell_{LR}}{I_z}(F_L-F_R)+\frac{(I_x-I_y)}{I_z}\ p\ q \end{bmatrix}, \tag{7.3.4}$$

where (x, y, z) denotes the position of the center of mass of the UAV. The magnitude of applied force along the X_B body axis is denoted F, and all others are zero. The mass of the UAV is $m^B = 2.0$ kg, g is gravity, and p, q, and r are the components of the angular velocity in body coordinates. Moreover, we have assumed that there are no products of inertia. The principle moments of inertia are denoted as follows: $I_x = 0.25\ kg\ m^2$, $I_y = 0.125\ kg\ m^2$, and $I_z = 0.125\ kg\ m^2$. The torque constant of the propeller is $k_w = 0.5$, $\ell_{FB} = 1.0$ m is half the distance between the applied force F_F and F_B, and $\ell_{LR} = 1.0$ m is half the distance between the applied force F_L and F_R as seen in Figure 7.20.

In order to determine the final form of the equations of motion, we take time derivatives of the components of the angular velocity and obtain:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} -\cos\theta\ \dot{\psi}\ \dot{\theta} + \ddot{\phi} - \sin\theta\ \ddot{\psi} \\ \cos\phi\ \dot{\theta} + \cos\theta\ \sin\phi\ \dot{\psi} \\ -\dot{\psi}\left(\cos\phi\ \sin\phi\ \dot{\theta} + \cos\theta\ \sin\phi\ \dot{\phi}\right) + \cos\phi\left(-\dot{\theta}\ \dot{\phi} + \cos\theta\ \ddot{\psi}\right) \end{bmatrix} \tag{7.3.5}$$

and replace these expressions in the attitudinal dynamics (7.3.4). Let us consider the minimum time optimal control problem of taking the VTOL UAV from an initial equilibrium solution to a final equilibrium solution subject to obstacle constraints and input constraints.

$$\min_{\mathbf{x},\mathbf{u}}\ t_f \tag{7.3.6}$$

$$\dot{\boldsymbol{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{7.3.7}$$

$$(\mathbf{x}(t_0), \mathbf{x}(t_f)) \in \mathcal{B} \tag{7.3.8}$$

$$(x(t), y(t), z(t)) \in \mathcal{I} \subseteq \mathcal{R} \tag{7.3.9}$$

$$\mathbf{u}(t) \in \mathcal{U}, \tag{7.3.10}$$

where the states $\mathbf{x} = (x, \dot{x}, y, \dot{y}, z, \dot{z}, \phi, \dot{\phi}, \theta, \dot{\theta}, \psi, \dot{\psi})$, and the inputs $\mathbf{u} = (F_L, F_R, F_F, F_B)$. In all, there are 12 states and 4 inputs. The region $\mathcal{I}$ specified in the trajectory constraints (7.3.9) is defined by the ordered union of adjacently overlapping polytopes by using the control points of the (x, y, z) NURBS curves. This system happens to be differentially flat. Therefore, we will exploit this fact to obtain a reduce set of variables to describe the full set of dynamics. The flat outputs, in this case, happen to be $z_1 = x,\ z_2 = y,\ z_3 = z,\ z_4 = \phi$. We can write the flat outputs in terms of the states, inputs, and their derivatives as follows:

$$\begin{aligned}
z_1^{(1)} &= \dot{x} \qquad z_1^{(2)} = F(\cos\theta\cos\psi) - g \\
z_2^{(1)} &= \dot{y} \qquad z_2^{(2)} = F(\cos\theta\sin\psi) \\
z_3^{(1)} &= \dot{z} \qquad z_3^{(2)} = -F\sin\theta \\
z_4^{(1)} &= \dot{\phi} \qquad z_4^{(2)} = f_1(\psi, \phi, \theta, \dot{\psi}, \dot{\theta}, \dot{\phi}, M_{X_b}, M_{Y_b}, M_{Z_b}) \\
z_1^{(3)} &= f_2(F, \dot{F}, \psi, \phi, \theta, \dot{\psi}, \dot{\theta}, \dot{\phi}) \\
z_2^{(3)} &= f_3(F, \dot{F}, \psi, \phi, \theta, \dot{\psi}, \dot{\theta}, \dot{\phi}) \\
z_3^{(3)} &= f_4(F, \dot{F}, \psi, \phi, \theta, \dot{\psi}, \dot{\theta}, \dot{\phi}) \\
z_1^{(4)} &= f_5(F, \dot{F}, \ddot{F}, \psi, \phi, \theta, \dot{\psi}, \dot{\theta}, \dot{\phi}, M_{Y_b}, M_{Z_b}) \\
z_2^{(4)} &= f_6(F, \dot{F}, \ddot{F}, \psi, \phi, \theta, \dot{\psi}, \dot{\theta}, \dot{\phi}, M_{Y_b}, M_{Z_b}) \\
z_3^{(4)} &= f_7(F, \dot{F}_1, \ddot{F}, \psi, \phi, \theta, \dot{\psi}, \dot{\theta}, \dot{\phi}, M_{Y_b}, M_{Z_b}).
\end{aligned}$$

In short,

$$(z_1, \ldots, z_2^{(4)}, z_2, \ldots, z_2^{(4)}, z_3, \ldots, z_3^{(4)}, z_4, \ldots, z^{(2)}) = \Psi(\xi),$$

where

$$\xi = (x, y, z, \theta, \phi, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\theta}, \dot{\phi}, \dot{\psi}, M_{X_b}, M_{Y_b}, M_{Z_b}, F, \dot{F}, \ddot{F}). \tag{7.3.11}$$

The above relation is locally invertible, with the exception of a few points, since

$$\det(\frac{\partial \Psi}{\partial \xi}) = \frac{-F^6 \cos\theta}{I_x I_y I_z} \tag{7.3.12}$$

is nonzero. For implementation purposes we need to explicitly write the states and inputs in terms of the flat outputs. From the flat output definitions, we obtain: $x = z_1,\ y = z_2,\ z = z_3,\ \phi = z_4$ and $\dot{x} = \dot{z}_1,\ \dot{y} = \dot{z}_2,\ \dot{z} = \dot{z}_3,\ \dot{\phi} = \dot{z}_4$. Using the equations of motion, we are now required to write ψ, θ, F, M_{B_X}, M_{B_Y}, and M_{B_Z} in terms of the flat outputs. Using the translational dynamics (7.3.3), in flat-output space we can solve for ψ, θ, and F analytically,

$$\psi = \tan^{-1}\left(\frac{\ddot{z}_2}{\ddot{z}_1 + g}\right) \tag{7.3.13}$$

$$\theta = \tan^{-1}\left(\frac{-\ddot{z}_3}{(\ddot{z}_1 + g)\ \cos\psi + \ddot{z}_2\ \sin\psi}\right) \tag{7.3.14}$$

$$F = m\sqrt{(\ddot{z}_1 + g)^2 + \ddot{z}_2^2 + \ddot{z}_3^2}. \tag{7.3.15}$$

Likewise, using the attitudinal dynamics (7.3.4) in flat-output space we can solve for the propulsive moments: M_{B_X}, M_{B_Y}, and M_{B_Z}

$$M_{B_X} = I_x\ \dot{p} + (I_z - I_y)\ q\ r \tag{7.3.16}$$

$$M_{B_Y} = I_y\ \dot{q} + (I_x - I_z)\ p\ r \tag{7.3.17}$$

$$M_{B_Z} = I_z\ \dot{r} + (I_y - I_x)\ p\ q. \tag{7.3.18}$$

To write the moments in flat-output space, we write the components of the angular velocity in terms of Euler angles and their derivatives (7.3.5), and substituting in (7.3.16)-(7.3.18) we obtain

$$\begin{aligned} M_{B_X} &= f_{B_X}(I_x, I_y, I_z, \theta, \phi, \dot{\theta}, \dot{\phi}, \dot{\psi}, \ddot{\phi}, \ddot{\psi}) \\ M_{B_Y} &= f_{B_Y}(I_x, I_y, I_z, \theta, \phi, \dot{\theta}, \dot{\phi}, \dot{\psi}, \ddot{\theta}, \ddot{\psi}) \\ M_{B_Z} &= f_{B_Z}(I_x, I_y, I_z, \theta, \phi, \dot{\theta}, \dot{\phi}, \dot{\psi}, \ddot{\psi}). \end{aligned}$$

We will require the following variables $\phi, \dot{\phi}, \ddot{\phi}, \psi, \dot{\psi}, \ddot{\psi}, \theta, \dot{\theta}$, and $\ddot{\theta}$. Since ϕ is a flat output we have

that $\phi = z_4,\ \dot{\phi} = \dot{z}_4,\ \ddot{\phi} = \ddot{z}_4$. The other derivatives can be obtained by taking time derivatives of (7.3.13) and (7.3.14). After these substitutions, the moments will depend only on the flat outputs and their time derivatives. Moreover, we are able to obtain the individual fan forces F_L, F_R, F_F and F_B by inverting

$$\begin{bmatrix} M_{XB} \\ M_{YB} \\ M_{ZB} \\ F \end{bmatrix} = \begin{bmatrix} k_w & k_w & -k_w & -k_w \\ 0 & 0 & \ell_{FB} & -\ell_{FB} \\ \ell_{LR} & -\ell_{LR} & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} F_L \\ F_R \\ F_F \\ F_B \end{bmatrix}.$$

The applied thrust forces are fixed in the body, as shown in Figure 7.20.

After this transformation, we are able to remove (7.3.7) from the minimum optimal control problem. We then proceed to parameterize the flat outputs by NURBS basis functions. In our case, we choose piecewise polynomial functions $\mathcal{P}_{\mathbf{b},o,s}$ for each flat output. More specifically $z_1, \ldots, z_3$ are $\mathcal{P}_{\mathbf{b},7,4}$, with 16 polynomial pieces being pasted at the breakpoints $\mathbf{b}$, where $\mathbf{b}$ is constructed by evenly positioning the end points of the polynomials in the interval $[0, 1]$. The flat output z_4 is also constructed with 16 polynomials and the same breakpoints $\mathbf{b}$ except for a different order and curve smoothness: $\mathcal{P}_{\mathbf{b},6,3}$. The next step is to construct a corridor in $\mathcal{S}$ which avoids all obstacles. This will allow us to remove (7.3.9) from the optimal control problem. Figure 7.21 shows the corridor that has been designed for this problem.

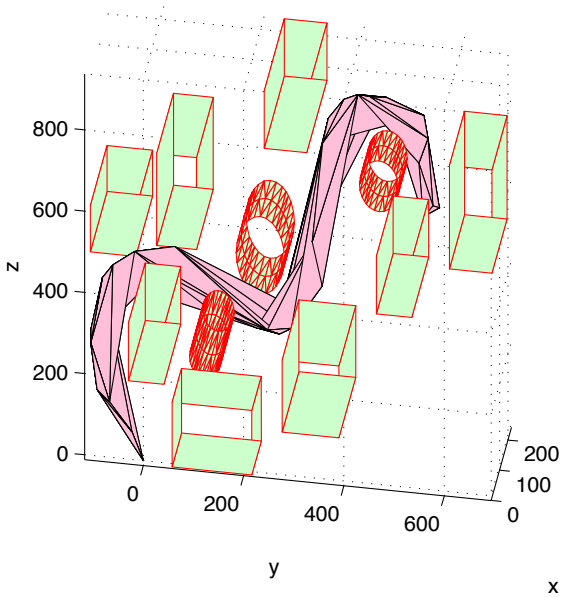

Figure 7.21: Feasible corridor with respect to obstacles in trajectory space.

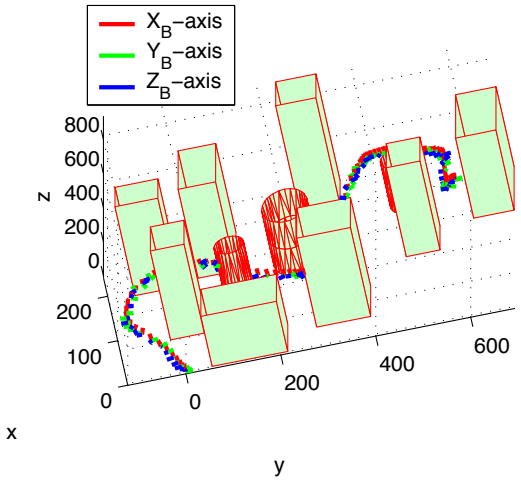

Figure 7.22: Front View: VTOL UAV trajectory shown with obstacles. Attitude is depicted by the rotation of the body axis.

The only constraints remaining in our optimal control problem are (7.3.8) and (7.3.10). The body forces F_F, F_R, F_L, and F_B are constrained to be in the interval $[0, 0.3\ m\ g]$. We further constrained the angle θ to the interval $(-\pi/2, \pi/2)$ to avoid the singularity described in (7.3.12). The optimal trajectory was found starting from a random initial guess as shown in Figure 7.22 (front view). The 125.6 sec trajectory was solved in real time on a 3 GHz Pentium IV PC. The body forces are shown in Figure 7.23.

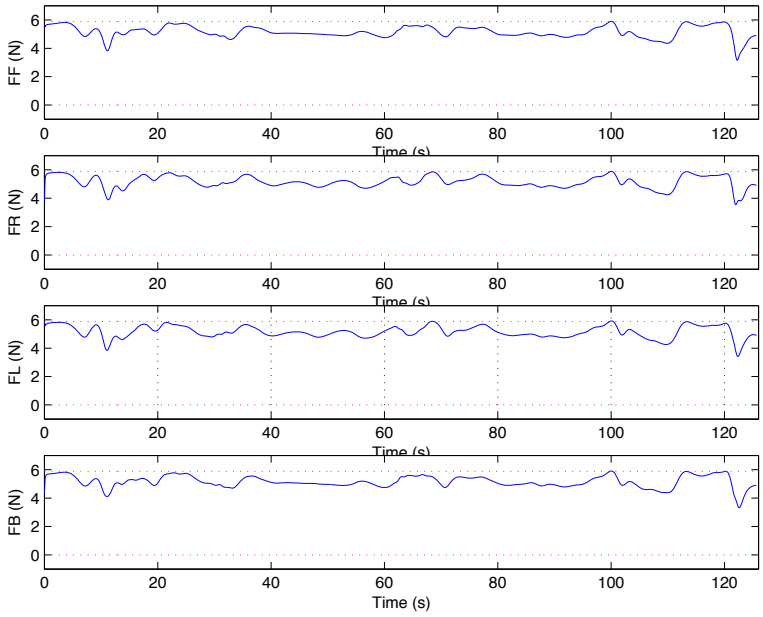

Figure 7.23: Body Forces: F_F, F_R, F_L and F_B.

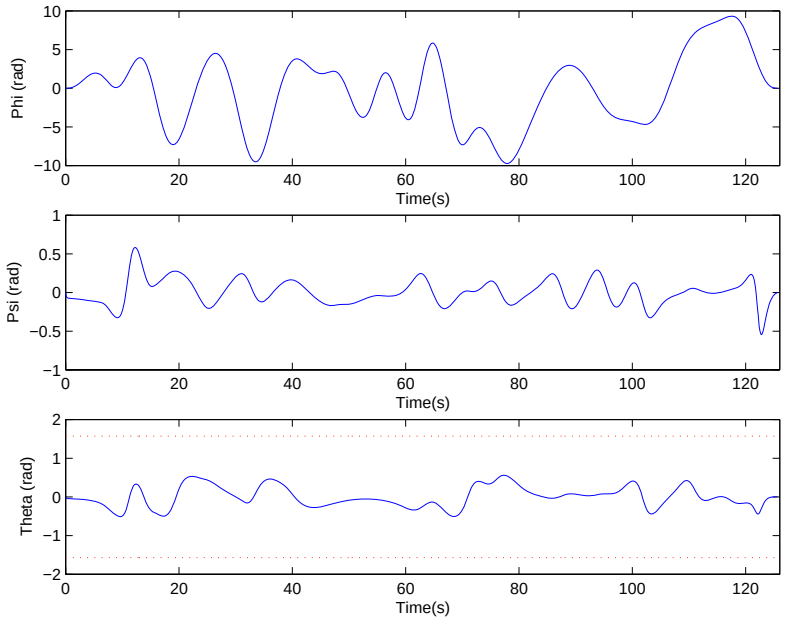

Figure 7.24: Euler angles: ϕ, ψ and θ.

Example 7.6 (Alice - Caltech's autonomous system)**.** We are going to revisit Example 5.3. In particular, this system is differentially flat, and we will take advantage of this fact. Let $z_1 = x$ and $z_2 = y$ be flat variables. Then, we use the dimensionalized equations of motion to solve for $u, \theta, \mathcal{A},$ and ϕ. The states and inputs are expressed in terms of the flat variables as follows:

$$\begin{aligned}
x(\tilde{\mathbf{z}}(\tau)) &= z_1(\tau) \\
y(\tilde{\mathbf{z}}(\tau)) &= z_2(\tau) \\
u(\tilde{\mathbf{z}}(\tau)) &= \sqrt{\dot{z}_1^2(\tau) + \dot{z}_2^2(\tau)} \\
\theta(\tilde{\mathbf{z}}(\tau)) &= \tan^{-1}\left[\frac{\dot{z}_2(\tau)}{\dot{z}_1(\tau)}\right] \\
\mathcal{A}(\tilde{\mathbf{z}}(\tau)) &= \dot{u}(\tilde{\mathbf{z}}(\tau)) \\
\phi(\tilde{\mathbf{z}}(\tau)) &= \tan^{-1}\left[\frac{\dot{\theta}(\tilde{\mathbf{z}}(\tau))\ L}{u(\tilde{\mathbf{z}}(\tau))}\right],
\end{aligned}$$

where

$$\begin{aligned}
\dot{u}(\tilde{\mathbf{z}}(\tau)) &= \frac{\Psi_0(\tilde{\mathbf{z}}(\tau))}{u(\tilde{\mathbf{z}}(\tau))} \\
\dot{\theta}(\tilde{\mathbf{z}}(\tau)) &= \frac{\Psi_1(\tilde{\mathbf{z}}(\tau))}{u^2(\tilde{\mathbf{z}}(\tau))} \\
\dot{\phi}(\tilde{\mathbf{z}}(\tau)) &= \frac{L\ u(\tilde{\mathbf{z}}(\tau))^2\ \left(3\ \Psi_1(\tilde{\mathbf{z}}(\tau))\ \dot{u}(\tilde{\mathbf{z}}(\tau)) - u(\tilde{\mathbf{z}}(\tau))\ \dot{\Psi}_1(\tilde{\mathbf{z}}(\tau))\right)}{u(\tilde{\mathbf{z}}(\tau))^6 + L^2\ \Psi_1^2(\tilde{\mathbf{z}}(\tau))} \\
\dot{\Psi}_1(\tilde{\mathbf{z}}(\tau)) &= \ddot{z}_2(\tau)\ \ddot{z}_1(\tau) - \dot{z}_2(\tau)\ \dddot{z}_1(\tau) - \ddot{z}_1(\tau)\ \ddot{z}_2(\tau) + \dot{z}_1(\tau)\ \dddot{z}_2(\tau)
\end{aligned}$$

with

$$\begin{aligned}
\Psi_0(\tilde{\mathbf{z}}(\tau)) &= \dot{z}_1(\tau)\ \ddot{z}_1(\tau) + \dot{z}_2(\tau)\ \ddot{z}_2(\tau) \\
\Psi_1(\tilde{\mathbf{z}}(\tau)) &= \dot{z}_1(\tau)\ \ddot{z}_2(\tau) - \dot{z}_2(\tau)\ \ddot{z}_1(\tau);
\end{aligned}$$

and, letting $z_3 = \xi$ we define $\tilde{\mathbf{z}}(\tau) = (z_1(\tau), \dot{z}_1(\tau), \ddot{z}_1(\tau), \dddot{z}_1(\tau), z_2(\tau), \dot{z}_2(\tau), \ddot{z}_2(\tau), \dddot{z}_2(\tau), z_3(\tau))$. In addition, we will add an extra term to our current cost functional to allowed obstacle and terrain information to be included in a form of cost. For this purpose, we will construct a NURBS surface, $\mathcal{S}(x(t), y(t))$, and construct the cost functional:

$$\int_C \mathcal{S}(x(t), y(t))\ ds = \int_{t_0}^{t_f} \mathcal{S}(x(t), y(t)) \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2}\ dt. \tag{7.3.19}$$

Using the flat parameterization, we rewrite the cost functional as follows:

$$\int_0^1 \mathcal{J}(\tilde{\mathbf{z}}(\tau)\ d\tau = \int_0^1 \mathcal{S}(z_1(\tau), z_2(\tau))\ \sqrt{\dot{z}_1(\tau)^2 + \dot{z}_2(\tau)^2}\ d\tau. \tag{7.3.20}$$

We rewrite the optimal control problem with this parameterization as follows:

$$\min \qquad \gamma_0\ z_3(1) + \gamma_1 \int_0^1 \left[\frac{\mathcal{A}(\tilde{\mathbf{z}}(\tau))^2}{z_3^3(\tau)} + z_3(\tau)\ \phi(\tilde{\mathbf{z}}(\tau))^2 + \frac{1}{z_3(\tau)}\ \dot{\phi}(\tilde{\mathbf{z}}(\tau))^2 \right]\ d\tau + \gamma_2 \int_0^1 \mathcal{J}(\tilde{\mathbf{z}}(\tau)\ d\tau$$

subject to

Trajectory Constraints:

$$\begin{aligned}
0.01 \leq \quad & \tfrac{u(\tilde{\mathbf{z}}(\tau))}{z_3(\tau)} && \leq v_{\max} \\
a_{\min} \leq \quad & \tfrac{\mathcal{A}(\tilde{\mathbf{z}}(\tau))}{z_3^2} && \leq a_{\max} \\
\phi_{\min} \leq \quad & \phi(\tilde{\mathbf{z}}(\tau)) && \leq \phi_{\max} \\
\dot{\phi}_{\min} \leq \quad & \tfrac{\dot{\phi}(\tilde{\mathbf{z}}(\tau))}{z_3(\tau)} && \leq \dot{\phi}_{\max} \\
-\frac{gW}{2h_{cg}} \leq \quad & \tfrac{\Psi_1(\tilde{\mathbf{z}}(\tau))}{u(\tilde{\mathbf{z}}(\tau))z_3^2(\tau)} && \leq \frac{gW}{2h_{cg}}
\end{aligned}$$

Initial and Final Boundary Conditions:

$$\begin{aligned}
z_1(0) &= x_0, & z_1(1) &= x_f \\
z_2(0) &= y_0, & z_2(1) &= y_f \\
\tan^{-1}(\frac{\dot{z}_1}{\dot{z}_2}) &= \theta_0, & \tan^{-1}(\frac{\dot{z}_1}{\dot{z}_2}) &= \theta_f
\end{aligned}$$

where γ_j, $j = \{0, 1, 2\}$ is 0 if not enforce and a nonzero real value otherwise. The cost for Alice traversing an urban environment can be set by pricing sections of space according to whether or not the vehicle is to allowed to traverse it, as illustrated in Figure 7.25. For purposes of trajectory generation using a direct method, we are restricted to cost functions that are at least differentiable. For this purpose, we will use a NURBS surface approximation of the cost map in the form of a smooth function (see Figure 7.26) with smooth partial derivatives (see Figure 7.27 and Figure 7.28). Fortunately, many of the properties of NURBS curves extend to NURBS surfaces such as the local support property. As in the case of NURBS curves, this property will make it possible to perform numerical computations on the NURBS surface by using only a subset of the basis functions.

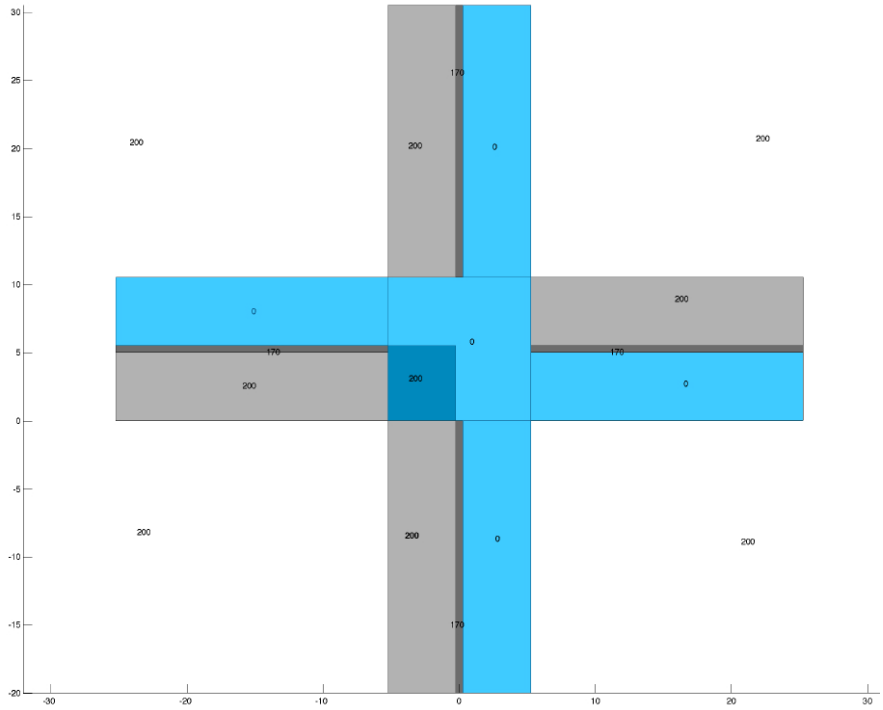

Figure 7.25: Cost of motion through an urban environment.

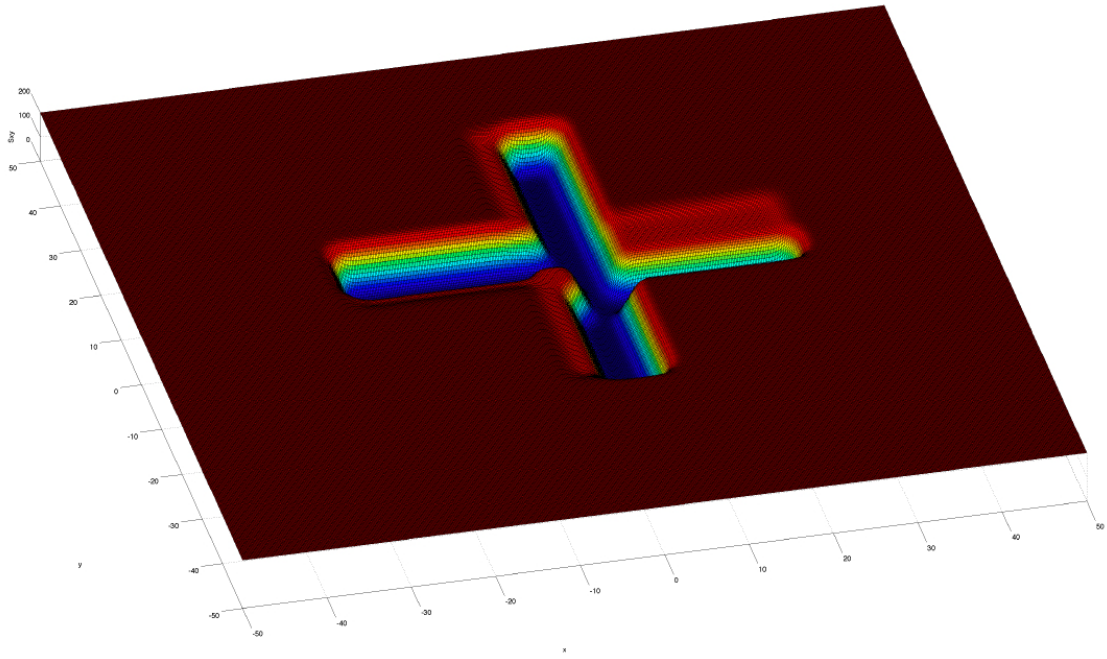

Figure 7.26: NURBS smoothing of the cost map.

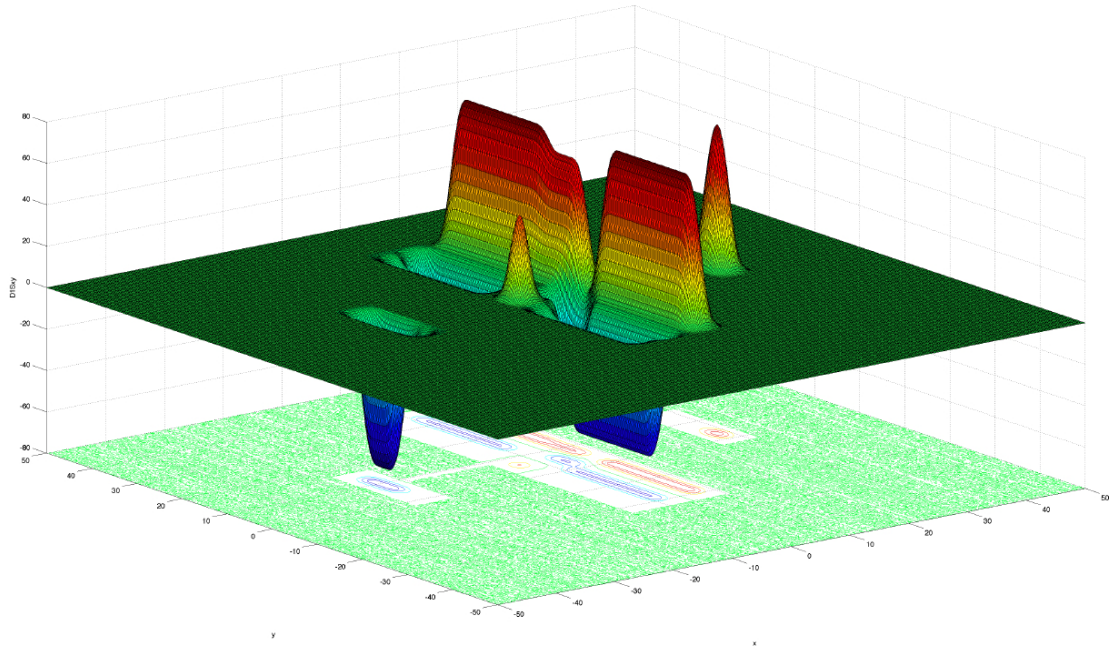

Figure 7.27: Partial derivative of cost map with respect to x.

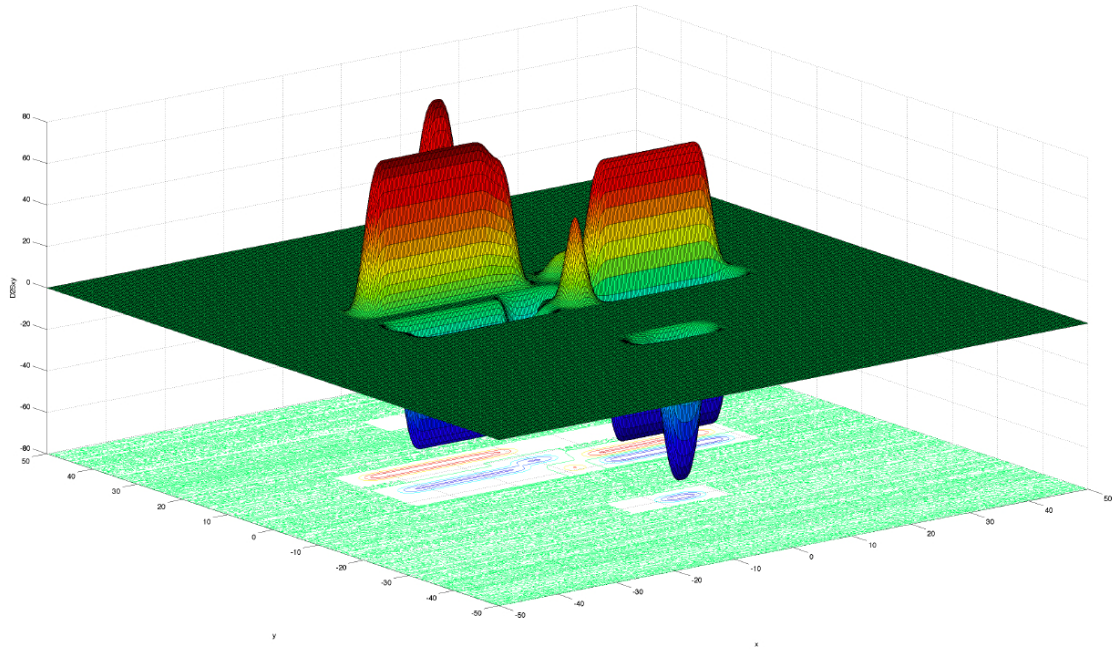

Figure 7.28: Partial derivative of cost map with respect to y

With the NURBS cost function in place, we are ready to determine trajectories for the vehicle that perform according to the cost of the motion. Figure 7.29 and Figure 7.30 show a left and right

maneuver, respectively. In addition to embedding safe driving regions, one can also embed obstacle cost as illustrated in Figure 7.31.

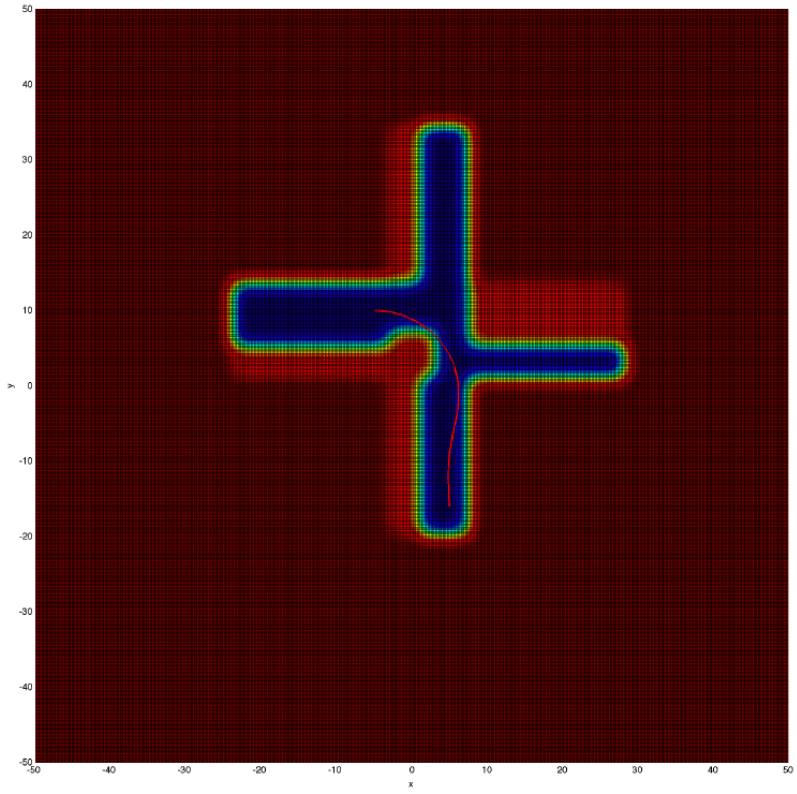

Figure 7.29: Minimum time left-turn maneuver respecting cost function.

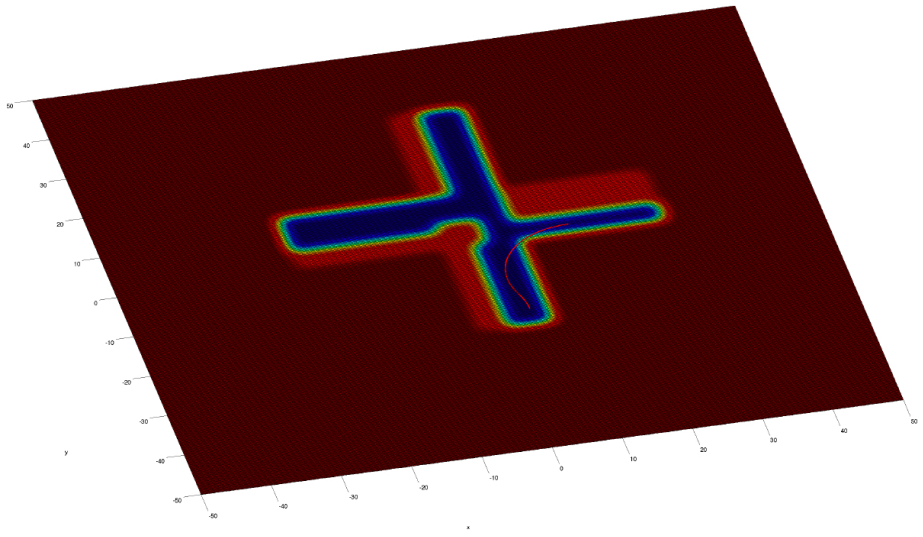

Figure 7.30: Minimum time right-turn maneuver respecting cost function.

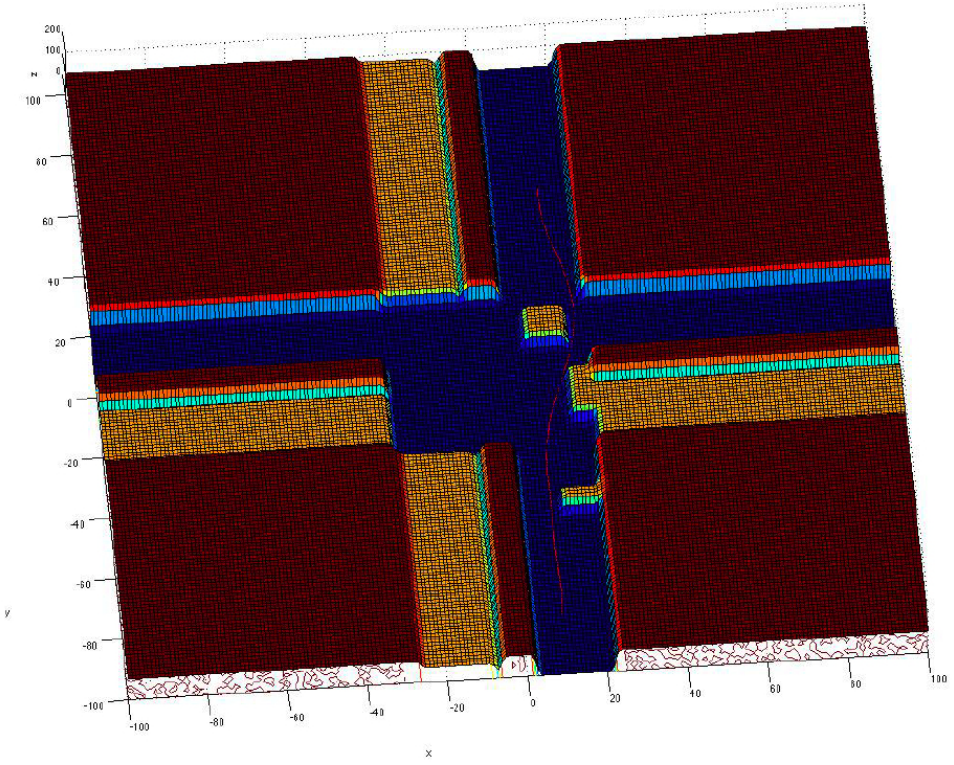

Figure 7.31: Minimum time obstacle avoidance maneuver.

Moreover, we are interested in using Theorem 7.1. There are two way that this was implemented in Alice: a) using DARPA's route data definition file (RDDF) and b) using corridor gates. In particular, Figure 7.32 illustrates the application of Theorem 7.1 starting form the RDDF specification. Although the RDDF specification meets the criteria of Theorem 7.1, the use of the specification leads to nonlinear constraints. We remedy this by approximating such regions by very low cost polytopal approximations (octagons). The result is an ordered union of pairwise adjacently overlapping polytopes with nonempty interiors, see Figure 7.32b. We proceed by truncating the feasible region of the first and last polytopes in such a way that the first and last control points are ensured to be vertices of the resulting approximating polytopes. This is done by finding the hyperplanes perpendicular to the direction vectors of both the first and last polytopes and then moving them towards the center of each by some small amount, see Figure 7.32c. Finally, we solve the nonlinear programming problem $NLP(\mathcal{P}_1, \ldots, \mathcal{P}_\mathcal{N})$ in order to obtain an inner polytopal approximation of the feasible region that satisfies NURBS parameterization. Lastly, we fixed the coordinates of the control points according to the optimal solution found in $NLP(\mathcal{P}_1, \ldots, \mathcal{P}_\mathcal{N})$, and we proceed to solve the modified optimal control problem with the active weights and control points as decision variables. The results are illustrated in Figure 7.33.

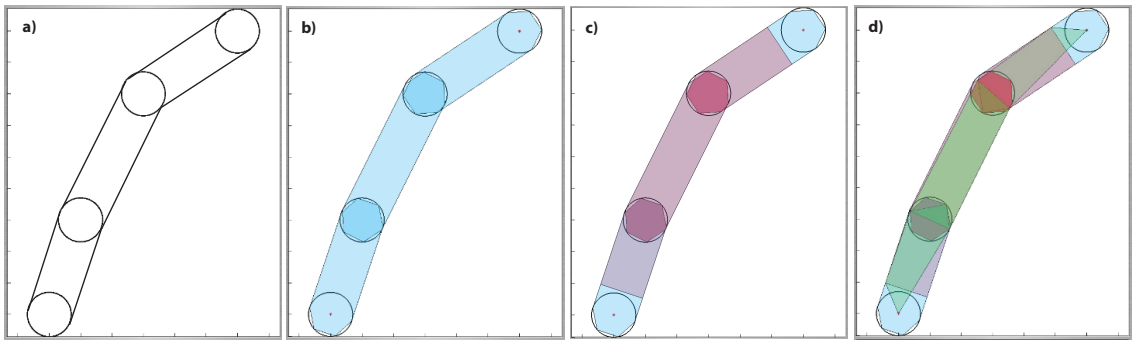

Figure 7.32: a) RDDF specification, b) Polytopal feasible region with pairwise adjacent, c) Ensure that initial and final control points are vertices by restricting and d) Approximation of feasible region by an ordered union of pairwise adjacently overlapping polytopes

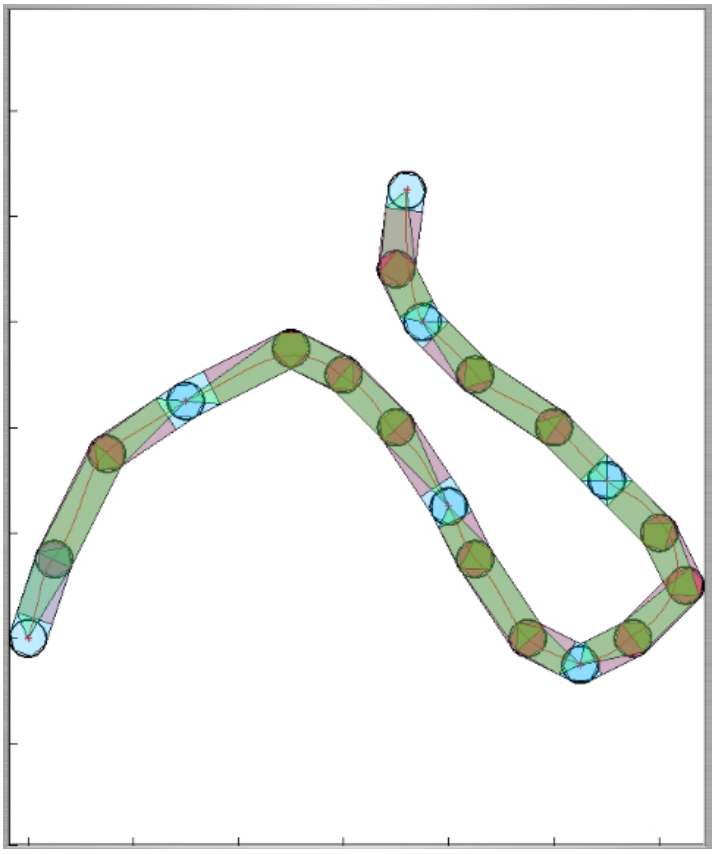

Figure 7.33: Consecutive trajectory generation for Alice

Chapter 8

Final Remarks

8.1 Summary

In this thesis, we have described a new method for numerically computing minimizers for optimal control problems that model the effort required for nonlinear dynamical systems to evolve across state configurations while satisfying trajectory and actuator constraints. In particular, we implemented a direct method (i.e., transcribed the original optimal control problem to a nonlinear programming problem). One of the main distinctions in this implementation from all others previously published resides in the choice of basis functions used to describe each member of the restricted search space. Specifically, the restricted search space consisted of the space of all piecewise polynomial functions with a prescribed number of polynomial pieces, order, and smoothness. This space happened to be finite dimensional and, consequently, each member in the space was expressed in terms of some set of basis functions. We chose NURBS basis functions for this purpose and showed how to judiciously exploit the properties of the resulting NURBS curve to improve the computational effort often associated with solving optimal control problems for constrained dynamical systems.

More specifically, we showed in detail how to exploit the structure of both the NURBS basis functions and the associated union of overlapping polytopes resulting from the placement in d-space ($d \in \{2, 3\}$) of the coefficients of the linear combination of the NURBS curve. One of the main realizations uncovered through the careful study of these structures was the fact that one can use the strong convex hull property of NURBS curves to separately treat the guidance and obstacle-avoidance problems, making the original optimal control problem tractable. This separation was accomplished by manipulating both the weights and control points; one set of parameters used at a time to solve one of the problems. That is, we showed how one can first design a connected section of space (in general, non-convex) by judiciously placing the control points in such a way that the union

of the associated polytopes was feasible with respect to trajectory constraints and contained the initial and final path conditions. Then, generate paths which minimized the original cost functional and satisfied the dynamic and actuator constraints (omitting the trajectory constraints), using the active weights as decision variables. The key observation was that paths generated by solving the latter problem were guaranteed to be contained inside the feasible region generated in the former phase, without the need of explicitly writing the trajectory constraints into the optimal control problem, and independent of the discretization.

In addition, we showed how one can construct systematically a feasible region that corresponds to a NURBS parameterization starting from an ordered union of pairwise adjacently overlapping nonempty compact convex sets. Specifically, we showed how to setup a nonlinear programming problem to solve for the feasible region in terms of an ordered union of pairwise adjacently overlapping polytopes with nonempty interiors by maximizing the sum of their volumes and starting from a feasible region described by an ordered union of pairwise adjacently overlapping nonempty convex compact semi-algebraic sets. In addition, we showed in simulation how this strategy could be implemented practically for an autonomous system traversing an urban environment.

Finally, this work culminated in the filing of patent US20070179685 on behalf of Northrop Grumman for the Space Technology sector and in the development of the NURBSbasedOTG software package. This C++ package contains the theoretical results of this thesis in the form of an object-oriented implementation optimized for real-time trajectory generation.

8.2 Conclusions

8.2.1 Towards a Real-Time implementation

At the current time, more than any other thus far, research on autonomous system is at its apogee. In fact, even as I write this, the Defense Advance Research Projects Agency (DARPA) is conducting the final selection of those teams that will be allowed to participate in the DARPA URBAN CHALLENGE 2007, taking place at the Southern California Logistics Airport in Victorville, California. All top universities and companies in the country have been working hard during the past 18 months to produce entries to this competition in the form of terrestrial vehicles guided purely by control algorithms written in a hand-full of different languages and running on board the fastest computers. The goal? To autonomously traverse an urban environment under time constraints while obeying traffic rules. The prize? Two million dollars to cover some of the expenses and priceless bragging rights.

Not surprising, the design of an autonomous system is a multidisciplinary endeavor, requiring technological advances from the many fields of science and engineering. In designing these types of systems, it is highly desirable to endow the units with a capability of determining, analyzing, and posing relevant problems mathematically on-line and solving them on time to implement the required corrections. The objective to be accomplished by the autonomous unit could be programmed by humans or could be self imposed. While trying to meet the objective, the autonomous unit will collect information using monitoring subsystems that sense the environment and its internal subsystems. This information will be varying constantly (obstacles are in different places, energy reserves are at different levels, faults might occur and so on). This information will be translated into new sets of state trajectory and input constraints, new model dynamics and new objectives; consequently, new problems will have to be assembled on-line using the newly gathered information. Moreover, the autonomous unit would seek for solutions that are in some way optimal in order to properly manage the limited resources at its disposal, and it would require that such solutions be obtainable in real-time. In summary, it is highly desirable to develop a feedback methodology that deals with online modification of the optimal control problem and that guarantees stability and performance of the closed-loop system under the added burden of reconfiguration. In line with this thesis' theme, the author suggests theoretical extensions to Receding Horizon Control (RHC). The biggest challenge in theoretically extending the RHC results to the setting of autonomous systems lies on determining a strategy that precludes the system from loosing feasibility after a model and/or constraint reconfiguration.

A first attempt to practically implement a reconfigurable strategy using the results of this thesis on an autonomous system was performed during the period of September 2006 - July 2007. The goal was to provide Alice (Caltech's autonomous vehicle) with an alternative trajectory generation strategy for use during the DARPA URBAN CHALLENGE 2007 race. Example 5.3 and Example 7.6 show some of the specifics of the implementation that was tried. In particular, in order to deal with reconfiguration, an object-oriented (inheritance, polymorphism, etc.) software approach was chosen. Careful attention was placed on trade-offs between flexibility and time computation, always favoring speed whenever possible. For the computations required to implement Theorem 7.1; that is, conversion between $\mathcal{H}$- to $\mathcal{V}$-polytope representations and volume computation, we used the *Double Description* method implemented by Fukuda and Prodon [1995] and the VINCI software package implemented by Büeler et al. [2000], respectively.

Unfortunately, three months short of the race, we realized that the time table for our approach to

be included in Alice had run its course. There still remained many milestones that needed to be accomplished before a reliable version could be used on board Alice. One of the major issues was reliability. Although we could solve many of the posed problems fast enough for real-time purposes, we could not consistently solve all problems posed, or at least with a reasonable rate of failure. Some of the reasons for this inconsistency have to do with the inherit nature of the methodology. That is,

a) the rate of convergence and success in finding a solution of a posed nonlinear programming problems tend to be sensitive to initial guesses — this translates into guessing both the weights and control points, as supposed to actual points on the NURBS curve. In addition, under constraint reconfiguration, one cannot reuse previous solutions as guesses to new problems, which is a very useful strategy in traditional RHC.

b) computing the exact volume of a polyhedron is NP hard (acceptable only for regions made up of a few sets) — sensing the environment sometimes leads naturally to representing the feasible space with a large number of overlapping sets and this tends to be a hard problem to solve.

c) not all problems posed have solutions and this cannot be detected ahead of time

d) conversion between $\mathcal{H}$- to $\mathcal{V}$-polytope representations are highly dependent on scaling, translation from the origin and rounding errors

e) SQP problems are only guaranteed to be feasible when an optimal solution is found — if feasibility was guaranteed throughout, one could stop computation after some time interval and use the latest feasible solution to guide the system

f) the rate of convergence of an SQP method is sped up when using the exact hessian but this could be a very expensive computation — using parallel algorithms and exploiting multi-core processors to split the computation can make these computation less burdensome.

Many of these issues can be overcome given enough time. In general, issues e) and f) could be overcome by the use of an alternative nonlinear programming solver such as the KNITRO (SLQP solver). This solver uses an interior point method to determine solutions of the posed problems, makes use of second-order data and it is able to maintain feasibility throughout. c) could be overcome by posing problems that are convex in the nonlinear programming problem sense even though they are not necessarily convex in the optimal control problem sense. b) could be solved by replacing the exact volume computation of polytopes with an approximate less expensive measure that gives the same intended result.

8.3 Future Work

8.3.1 From Cell Decomposition to Pairwise Adjacently Overlapping Cells

In this thesis, we mentioned on passing that one way in which the region $\mathcal{R}$, alluded on Theorem 7.1, could be constructed was by performing a cell decomposition of the trajectory space followed by a pairwise adjacently overlapping procedure. How this actually is done practically still remains an open question. However, it seems to the author that this should be possible for those cases where the trajectory space is decomposed using polytopal cells by constructing the smallest set containing the centroids of pairwise adjacent cells. Consider the examples shown in Figure 8.1. In both cases, one begins with a polytopal cell decomposition. Is it possible to produce an overlapping procedure such that one can move to adjacent cells starting from the pink cell shown in the figure on the left? The answer seems to be yes. The green polytope constructed using the centroids of the cells produces an overlapping of the pink cells. Starting from any pink cell and using the green cell to achieve an overlap, we are able to move to any other pink cell satisfying the conditions of Theorem 7.1. One of the challenges to overcome is to achieve an overlapping procedure that is complete (i.e., if it fails no such overlapping is possible) when obstacles are present in the trajectory space.

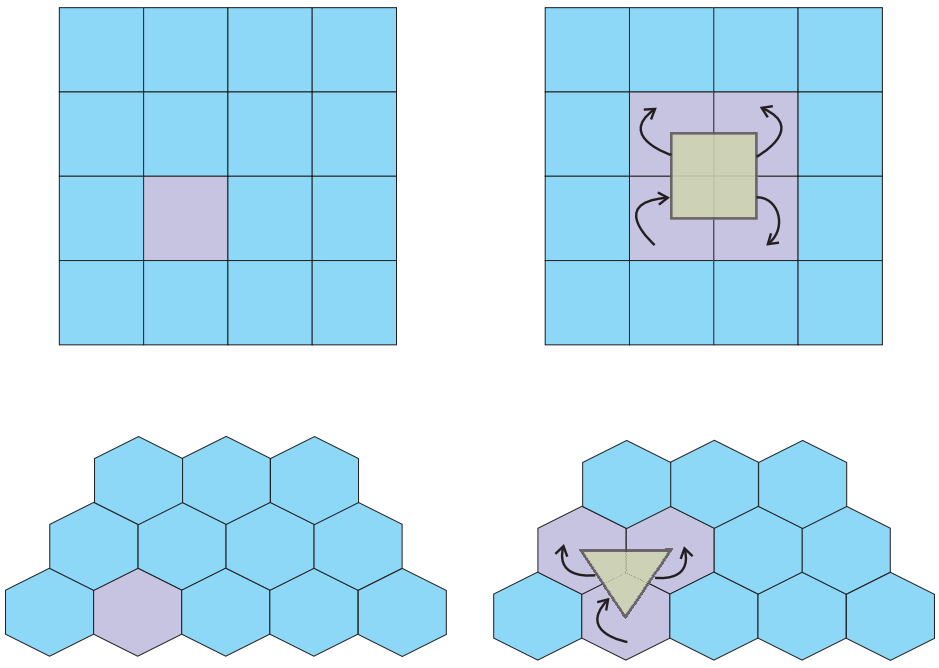

Figure 8.1: Two examples of polytopal cell decomposition that undergo an overlapping procedure using the convex hull of the centroids of adjacent cells.

8.3.2 Exploiting NURBS curve invariance properties

Among the properties of NURBS curves left out in the body of this thesis, we have the *Refinability property*. Consider the following NURBS curve:

$$\mathbf{c}(t) \quad = \quad \left[\begin{array}{ccc} \mathcal{R}^{(0)}_{\ell-d,d}(t,\mathbf{w}) & \ldots & \mathcal{R}^{(0)}_{\ell,d}(t,\mathbf{w}) \end{array} \right] \left[\begin{array}{c} p_{\ell-d} \\ \vdots \\ p_{\ell} \end{array} \right], \quad t \in [k_\ell, k_{\ell+1}).$$

As mentioned before, the NURBS basis functions are expressed in terms of B-spline basis functions as follows:

$$\mathcal{R}^{(0)}_{j,d}(t, w_0, \ldots, w_{N_c-1}) = \frac{\mathcal{B}^{(0)}_{j,d}(t)\ w_j}{\sum\limits_{i=0}^{N_c-1} \mathcal{B}^{(0)}_{i,d}(t)\ w_i}, \tag{8.3.1}$$

and the NURBS curve expressed in terms of B-splines becomes

$$c(t, \mathbf{w}, \mathbf{p}) = \frac{\left[\mathcal{B}^{(0)}_{d}(t)\right]^T \mathbf{p_w}}{\left[\mathcal{B}^{(0)}_{d}(t)\right]^T \mathbf{w}}, \tag{8.3.2}$$

where

$$\mathbf{p_w} = \left[\begin{array}{c} w_{\ell-d}\ p_{\ell-d} \\ \vdots \\ w_{\ell}\ p_{\ell} \end{array} \right]. \tag{8.3.3}$$

Let us begin by defining the B-spline function of degree 0.

$$\mathcal{B}_0(t) = \begin{cases} 1, & \text{if } t \in [0,1) \\ 0, & \text{otherwise} \end{cases}. \tag{8.3.4}$$

In general, we can obtain a B-spline of degree d by using the following convolution

$$\mathcal{B}_d(t) = \int_{-\infty}^{\infty} \mathcal{B}_{d-1}(s)\ \mathcal{B}_0(t-s)\ ds. \tag{8.3.5}$$

In addition, we connect these B-spline definitions to the ones in the linear combination using the following property: $\mathcal{B}^{(0)}_{k,d}(t) = \mathcal{B}^{(0)}_{d}(t-k)$.

The refinement equation for B-splines of degree d is given by

$$\mathcal{B}_d^{(0)}(t) = \frac{1}{2^d} \sum_{k=0}^{d+1} \begin{pmatrix} d+1 \\ k \end{pmatrix} \mathcal{B}_d^{(0)}(2t-k). \tag{8.3.6}$$

Basically, it states that a B-spline of degree d can be written as a linear combination of translated (k) and dilated $(2t)$ copies of itself. Expressing the refinement property in matrix form we have that

$$\begin{aligned} \mathcal{B}_d^{(0)}(t) &= \begin{bmatrix} \mathcal{B}_d^{(0)}(2t) & \mathcal{B}_d^{(0)}(2t-1) & \dots & \mathcal{B}_d^{(0)}(2t-(d+1)) \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{d+1} \end{bmatrix} \\ &= \mathcal{B}_d^{(0)}(2t)\ \mathbf{s}_d \end{aligned}$$

with $s_k = \frac{1}{2^d} \begin{pmatrix} d+1 \\ k \end{pmatrix}$. Using the refinement property, we are able to write the NURBS curve as follows:

$$c(t, \mathbf{w}, \mathbf{p}) = \frac{\left[\mathcal{B}_d^{(0)}(2t)\right]^T \mathbf{S}_d\ \mathbf{p_w}}{\left[\mathcal{B}_d^{(0)}(2t)\right]^T \mathbf{S}_d\ \mathbf{w}}, \qquad \mathbf{S}_d \in \mathbb{R}^{(2N_c+d)\times N_c}. \tag{8.3.7}$$

The non-zero entries of the matrix $\mathbf{S}_d$ are determine as follows:

$$\mathbf{S}_{k+2i,i} = \frac{1}{2^d} \begin{pmatrix} d+1 \\ k \end{pmatrix}, \qquad i \in \{0, \dots, N_c - 1\},\ k \in \{0, \dots, d+1\} \tag{8.3.8}$$

This way of expressing curves is at the heart of *Subdivision*. Note that this is still the same NURBS curve but with a change of basis: $\mathcal{B}_0(t) \mapsto \mathcal{B}_0(2t)$ and concurrently change of weights and control points: $\mathbf{w} \mapsto \mathbf{S}_d\ \mathbf{w}$ and $\mathbf{p_w} \mapsto \mathbf{S}_d\ \mathbf{p_w}$. Moreover, the support of the basis functions is half as wide, and they are spaced twice as dense.

Example 8.1. Consider the B-spline of degree 1:

$$\begin{aligned} \mathcal{B}_1^{(0)}(t) &= \frac{1}{2} \sum_{k=0}^{2} \begin{pmatrix} 2 \\ k \end{pmatrix} \mathcal{B}_1^{(0)}(2t-k) \\ &= \frac{1}{2} \begin{pmatrix} 2 \\ 0 \end{pmatrix} \mathcal{B}_1^{(0)}(2t) + \frac{1}{2} \begin{pmatrix} 2 \\ 1 \end{pmatrix} \mathcal{B}_1^{(0)}(2t-1) + \frac{1}{2} \begin{pmatrix} 2 \\ 2 \end{pmatrix} \mathcal{B}_1^{(0)}(2t-2) \\ &= \frac{1}{2}\, \mathcal{B}_1^{(0)}(2t) + 1\ \mathcal{B}_1^{(0)}(2t-1) + \frac{1}{2}\, \mathcal{B}_1^{(0)}(2t-2) \end{aligned}$$

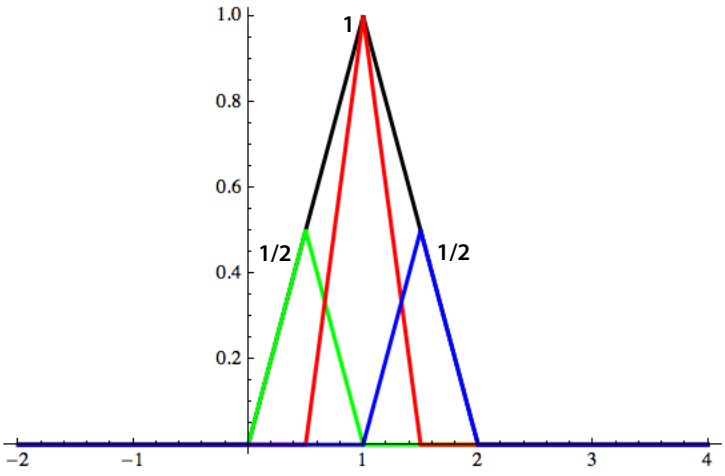

Figure 8.2: Refinement applied to $\mathcal{B}_1^{(0)}(t)$.

It is clear from Figure 8.2 that the black curve is constructed by adding the blue, red and green dilated and translated versions of the B-spline of degree 1 using the coefficients $(1/2, 1, 1/2)$. Consider the NURBS curve made up of $N_p = 5$ polynomial pieces of degree $d = 1$ and smoothness $s = 0$. The dimension of the vector space is $N_c = 5(2-1)+1 = 6$. The NURBS curve then is expressed as follows:

$$c(t, \mathbf{w}, \mathbf{p}) = \frac{\left[\mathcal{B}_1^{(0)}(t)\right]^T \mathbf{S}_1 \mathbf{p}_\mathbf{w}}{\left[\mathcal{B}_1^{(0)}(t)\right]^T \mathbf{S}_1 \ \mathbf{w}}. \tag{8.3.9}$$

Applying the refinement equation to the B-spline matrix $\left[\mathcal{B}_1^{(0)}(t)\right]^T$, we obtain:

$$\left[\mathcal{B}_1^{(0)}(t)\right]^T = \left[\mathcal{B}_1^{(0)}(2t)\right]^T \ \mathbf{S}_1, \tag{8.3.10}$$

where

$$\left[\mathcal{B}_1^{(0)}(t)\right]^T = \left[\ \mathcal{B}_{0,1}^{(0)}(t) \quad \mathcal{B}_{1,1}^{(0)}(t) \quad \ldots \quad \mathcal{B}_{5,1}^{(0)}(t) \ \right] \tag{8.3.11}$$

$$\left[\mathcal{B}_1^{(0)}(2t)\right]^T = \left[\ \mathcal{B}_{0,1}^{(0)}(2t) \quad \mathcal{B}_{1,1}^{(0)}(2t) \quad \ldots \quad \mathcal{B}_{12,1}^{(0)}(2t) \ \right]. \tag{8.3.12}$$

For the sake of simplicity, consider the case when the weights are all set to the same value $w_0 = \cdots = w_{N_c-1} = \omega > 0$. That is,

$$c(t, \mathbf{p}) \quad = \quad \left[\mathcal{B}_1^{(0)}(2t)\right]^T \mathbf{S}_1 \mathbf{p} \tag{8.3.13}$$

Let $\mathbf{p}^1 = \mathbf{S}_1\ \mathbf{p} = \mathbf{S}_1\ \mathbf{p}^0$. That is,

$$\underbrace{\begin{bmatrix} p_0^1 \\ p_1^1 \\ p_2^1 \\ p_3^1 \\ p_4^1 \\ p_5^1 \\ p_6^1 \\ p_7^1 \\ p_8^1 \\ p_9^1 \\ p_{10}^1 \\ p_{11}^1 \\ p_{12}^1 \end{bmatrix}}_{\mathbf{p}^1} = \underbrace{\begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}}_{\mathbf{S}_1} \underbrace{\begin{bmatrix} p_0^0 \\ p_1^0 \\ p_2^0 \\ p_3^0 \\ p_4^0 \\ p_5^0 \end{bmatrix}}_{\mathbf{p}^0}. \tag{8.3.14}$$

The question is how do we exploit this property to our advantage. For example, when constructing the overlapping set of polytopes one could apply the change of control points to get smaller polytopes and a larger set of basis functions and weights. Having a larger set of weights is useful in having a finer control over the curve in order to satisfy all constraints of the optimal control problem. The challenge resides in the fact that not all subdivisions lead to a scheme where the old control points keep their coordinates and new ones are added to divide the polytopal space (desired behavior). Instead, some schemes move all the control points, and in those cases feasibility is lost.

Appendix A

Derivation of Partial Derivatives of NURBS Curves

In this appendix, we prove Proposition 3.2.

Proposition A.1. Consider $c(t, \mathbf{w}, \mathbf{p})$ to be a smooth curve expressed in terms of a linear combination of NURBS basis functions. Then its rth order partial derivatives with respect to time and their first- and second-order partial derivatives with respect to control points and weights are obtained as follows:

$$
\begin{aligned}
c^{(r)}(t,\mathbf{w},\mathbf{p}) &= \mathbf{w}^T\mathbf{G}^r\mathbf{p} - \sum_{k=1}^{r}\begin{pmatrix} r \\ k \end{pmatrix}\mathbf{E}^k\ c^{(r-k)}(t,\mathbf{w},\mathbf{p}) \\
\mathbf{D_p}\left[c^{(r)}(t,\mathbf{w},\mathbf{p})\right] &= \mathbf{w}^T\mathbf{G}^r - \sum_{k=1}^{r}\begin{pmatrix} r \\ k \end{pmatrix}\mathbf{E}^k\ \left[\mathcal{R}_d^{(r-k)}(t,\mathbf{w})\right]^T \\
\mathbf{D_{pp}}\left[c^{(r)}(t,\mathbf{w},\mathbf{p})\right] &= \mathbf{0} \\
\mathbf{D_w}\left[c^{(r)}(t,\mathbf{w},\mathbf{p})\right] &= -\left[\left(\mathbf{F}^0-\mathbf{I}\right)\mathbf{G}^r\ \mathbf{p}\right]^T \\
&\quad + \sum_{k=1}^{r}\begin{pmatrix} r \\ k \end{pmatrix}\left[\left(\mathbf{F}^0-\mathbf{I}\right)\left(\mathbf{H}^k\right)^T\right]^T c^{(r-k)}(t,\mathbf{w},\mathbf{p}) \\
&\quad - \sum_{k=1}^{r}\begin{pmatrix} r \\ k \end{pmatrix}\mathbf{E}^k\ \mathbf{D_w}\left[c^{(r-k)}(t,\mathbf{w},\mathbf{p})\right] \\
\mathbf{D_{ww}}\left[c^{(r)}(t,\mathbf{w},\mathbf{p})\right] &= \left(\mathbf{F}^0-\mathbf{I}\right)\mathbf{G}^r\ \mathbf{p}\ \mathbf{H}^0 + \left[\left(\mathbf{F}^0-\mathbf{I}\right)\mathbf{G}^r\ \mathbf{p}\ \mathbf{H}^0\right]^T \\
&\quad + \sum_{k=1}^{r}\begin{pmatrix} r \\ k \end{pmatrix}\left(\mathbf{AB}+[\mathbf{AB}]^T\right) \\
&\quad - \sum_{k=1}^{r}\begin{pmatrix} r \\ k \end{pmatrix}\mathbf{E}^k\ \mathbf{D_{ww}}\left[c^{(r-k)}(t,\mathbf{w},\mathbf{p})\right]
\end{aligned}
$$

with $t \in [t_0, t_f]$, $w_j \in (0, \infty)$, $p_j \in (-\infty, \infty)$, $\left[\mathcal{B}_d^{(r)}(t)\right] = \begin{bmatrix} \mathcal{B}_{0,d}^{(r)}(t) \\ \vdots \\ \mathcal{B}_{N_c-1,d}^{(r)}(t) \end{bmatrix}$, $\mathbf{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_{N_c-1} \end{bmatrix}$,
$\mathbf{p} = \begin{bmatrix} p_0 \\ \vdots \\ p_{N_c-1} \end{bmatrix}$, $\text{diag}(\{\mathcal{B}_d^{(r)}(t)\}) = \begin{bmatrix} \mathcal{B}_{0,d}^{(r)}(t) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \mathcal{B}_{N_c-1,d}^{(r)}(t) \end{bmatrix}$, $\mathbf{H}^\ell = \frac{\left[\mathcal{B}_d^{(\ell)}(t)\right]^T}{\left[\mathcal{B}_d^{(0)}(t)\right]^T \mathbf{w}}$,
$\mathbf{E}^\ell = \mathbf{H}^\ell\ \mathbf{w}$, $\mathbf{F}^\ell = \left(\mathbf{w}\ \mathbf{H}^\ell\right)^T$, $\mathbf{G}^\ell = \frac{\text{diag}(\{\mathcal{B}_d^{(\ell)}(t)\})}{\left[\mathcal{B}_d^{(0)}(t)\right]^T \mathbf{w}}$, $\mathbf{A} = \mathbf{E}^k\ \left[\mathcal{B}_d^{(0)}(t)\right] - \left[\mathcal{B}_d^{(k)}(t)\right]$ and
$\mathbf{B} = \frac{\mathbf{D}_\mathbf{w}\left[c^{(r-k)}(t,\mathbf{w},\mathbf{p})\right]}{\left[\mathcal{B}_d^{(0)}(t)\right]^T \mathbf{w}} - \mathbf{H}^\ell\ \frac{c^{(r-k)}(t,\mathbf{w},\mathbf{p})}{\left[\mathcal{B}_d^{(0)}(t)\right]^T \mathbf{w}}$.

Proof. We begin with the derivation of the rth-order partial derivatives with respect to time of the NURBS curve and then proceed to derive the first- and second-order partial derivatives of these with respect to weights and control points.

Partial derivatives with respect to time

The time-derivatives of the NURBS curve parameterization can succinctly be expressed as follows:

$$c^{(r)}(t, \mathbf{w}, \mathbf{p}) = \sum_{j=0}^{N_c-1} \mathcal{R}_{j,d}^{(r)}(t, \mathbf{w})\ p_j, \quad t \in [t_0, t_f], w_j \in (0, \infty), p_j \in (-\infty, \infty), \tag{A.0.1}$$

where $\mathbf{w} = \begin{bmatrix} w_0 & \dots & w_{N_c-1} \end{bmatrix}^T$ and $\mathbf{p} = \begin{bmatrix} p_0 & \dots & p_{N_c-1} \end{bmatrix}^T$, and $\mathcal{R}_{j,d}^{(r)}(t, \mathbf{w}) = \frac{\partial^r}{\partial t^r}\left[\mathcal{R}_{j,d}^{(0)}(t, \mathbf{w})\right]$.

Clearly, the time dependence is directly related to the NURBS basis functions only. The first four time derivatives of the NURBS basis functions are expressed as follows:

$$\begin{aligned} \mathcal{R}_{j,d}^{(0)}(t, \mathbf{w}) &= \frac{\mathcal{B}_{j,d}^{(0)}(t)\ w_j}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t)\ w_i} \\ \mathcal{R}_{j,d}^{(1)}(t, \mathbf{w}) &= \frac{\mathcal{B}_{j,d}^{(1)}(t)\ w_j}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t)\ w_i} - \frac{\sum_{i=0}^{N_c-1} \mathcal{B}_{j,d}^{(1)}(t)\ w_j}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t)\ w_i} \mathcal{R}_{j,d}^{(0)}(t, \mathbf{w}) \end{aligned}$$

$$\mathcal{R}^{(2)}_{j,d}(t,\mathbf{w}) = \frac{\mathcal{B}^{(2)}_{j,d}(t)\ w_j}{\sum_{i=0}^{N_c-1}\mathcal{B}^{(0)}_{i,d}(t)\ w_i} - 2\frac{\sum_{i=0}^{N_c-1}\mathcal{B}^{(1)}_{j,d}(t)\ w_j}{\sum_{i=0}^{N_c-1}\mathcal{B}^{(0)}_{i,d}(t)\ w_i}\ \mathcal{R}^{(1)}_{j,d}(t,\mathbf{w}) - \frac{\sum_{i=0}^{N_c-1}\mathcal{B}^{(2)}_{j,d}(t)\ w_j}{\sum_{i=0}^{N_c-1}\mathcal{B}^{(0)}_{i,d}(t)\ w_i}\mathcal{R}^{(0)}_{j,d}(t,\mathbf{w})$$

$$\begin{aligned}\mathcal{R}^{(3)}_{j,d}(t,\mathbf{w}) &= \frac{\mathcal{B}^{(3)}_{j,d}(t)\ w_j}{\sum_{i=0}^{N_c-1}\mathcal{B}^{(0)}_{i,d}(t)\ w_i} - 3\frac{\sum_{i=0}^{N_c-1}\mathcal{B}^{(1)}_{j,d}(t)\ w_j}{\sum_{i=0}^{N_c-1}\mathcal{B}^{(0)}_{i,d}(t)\ w_i}\ \mathcal{R}^{(2)}_{j,d}(t,\mathbf{w}) - 3\frac{\sum_{i=0}^{N_c-1}\mathcal{B}^{(2)}_{j,d}(t)\ w_j}{\sum_{i=0}^{N_c-1}\mathcal{B}^{(0)}_{i,d}(t)\ w_i}\mathcal{R}^{(1)}_{j,d}(t,\mathbf{w}) \\ &\quad - \frac{\sum_{i=0}^{N_c-1}\mathcal{B}^{(3)}_{j,d}(t)\ w_j}{\sum_{i=0}^{N_c-1}\mathcal{B}^{(0)}_{i,d}(t)\ w_i}\mathcal{R}^{(0)}_{j,d}(t,\mathbf{w}).\end{aligned}$$

In general, the time derivatives of the NURBS basis functions take the following form:

$$\mathcal{R}^{(r)}_{j,d}(t,\mathbf{w}) = \frac{\mathcal{B}^{(r)}_{j,d}(t)\ w_j}{\sum_{i=0}^{N_c-1}\mathcal{B}^{(0)}_{i,d}(t)\ w_i} - \sum_{k=1}^{r}\binom{r}{k}\frac{\sum_{i=0}^{N_c-1}\mathcal{B}^{(k)}_{i,d}(t)\ w_i}{\sum_{i=0}^{N_c-1}\mathcal{B}^{(0)}_{i,d}(t)\ w_i}\ \mathcal{R}^{(r-k)}_{j,d}(t,\mathbf{w}). \tag{A.0.2}$$

Alternatively, in matrix form

$$\mathcal{R}^{(r)}_{j,d}(t,\mathbf{w}) = \frac{\mathcal{B}^{(r)}_{j,d}(t)\ w_j}{\left[\mathcal{B}^{(0)}_{d}(t)\right]^T\mathbf{w}} - \sum_{k=1}^{r}\binom{r}{k}\frac{\left[\mathcal{B}^{(k)}_{d}(t)\right]^T\mathbf{w}}{\left[\mathcal{B}^{(0)}_{d}(t)\right]^T\mathbf{w}}\ \mathcal{R}^{(r-k)}_{j,d}(t,\mathbf{w}). \tag{A.0.3}$$

Consequently, the time derivatives of the curve become:

$$\begin{aligned}c^{(r)}(t,\mathbf{w},\mathbf{p}) &= \sum_{j=0}^{N_c-1}\mathcal{R}^{(r)}_{j,d}(t,\mathbf{w})\ p_j \\ &= \sum_{j=0}^{N_c-1}\frac{\mathcal{B}^{(r)}_{j,d}(t)\ w_j}{\sum_{i=0}^{N_c-1}\mathcal{B}^{(0)}_{i,d}(t)\ w_i}p_j - \sum_{j=0}^{N_c-1}\left(\sum_{k=1}^{r}\binom{r}{k}\frac{\sum_{i=0}^{N_c-1}\mathcal{B}^{(k)}_{i,d}(t)\ w_i}{\sum_{i=0}^{N_c-1}\mathcal{B}^{(0)}_{i,d}(t)\ w_i}\ \mathcal{R}^{(r-k)}_{j,d}(t,\mathbf{w})\right)p_j \\ &= \frac{\sum_{j=0}^{N_c-1}\mathcal{B}^{(r)}_{j,d}(t)\ w_j p_j}{\sum_{i=0}^{N_c-1}\mathcal{B}^{(0)}_{i,d}(t)\ w_i} - \sum_{k=1}^{r}\binom{r}{k}\frac{\sum_{i=0}^{N_c-1}\mathcal{B}^{(k)}_{i,d}(t)\ w_i}{\sum_{i=0}^{N_c-1}\mathcal{B}^{(0)}_{i,d}(t)\ w_i}\ \sum_{j=0}^{N_c-1}\mathcal{R}^{(r-k)}_{j,d}(t,\mathbf{w})p_j \\ &= \frac{\sum_{j=0}^{N_c-1}\mathcal{B}^{(r)}_{j,d}(t)\ w_j p_j}{\sum_{i=0}^{N_c-1}\mathcal{B}^{(0)}_{i,d}(t)\ w_i} - \sum_{k=1}^{r}\binom{r}{k}\frac{\sum_{i=0}^{N_c-1}\mathcal{B}^{(k)}_{i,d}(t)\ w_i}{\sum_{i=0}^{N_c-1}\mathcal{B}^{(0)}_{i,d}(t)\ w_i}\ c^{(r-k)}(t,\mathbf{w},\mathbf{p}).\end{aligned}$$

Alternatively, in matrix form:

$$c^{(r)}(t,\mathbf{w},\mathbf{p}) \;=\; \frac{\mathbf{w}^T \mathrm{diag}(\{\mathcal{B}_d^{(r)}(t)\})\mathbf{p}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T \mathbf{w}} - \sum_{k=1}^{r} \begin{pmatrix} r \\ k \end{pmatrix} \frac{\left[\mathcal{B}_d^{(k)}(t)\right]^T \mathbf{w}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T \mathbf{w}}\, c^{(r-k)}(t,\mathbf{w},\mathbf{p}) \quad \text{(A.0.4)}$$

with $t \in [t_0, t_f]$, $w_j \in (0,\infty)$, and $p_j \in (-\infty,\infty)$.

Partial derivatives with respect to control points

Next we will derive the first-order control point derivatives with respect to a NURBS curve.

$$\mathbf{D}_{\mathbf{p}}\left[c^{(r)}(t,\mathbf{w},\mathbf{p})\right] = \left[\begin{array}{cccc} \frac{\partial c^{(r)}(t,\mathbf{w},\mathbf{p})}{\partial p_0} & \frac{\partial c^{(r)}(t,\mathbf{w},\mathbf{p})}{\partial p_1} & \cdots & \frac{\partial c^{(r)}(t,\mathbf{w},\mathbf{p})}{\partial p_{N_c-1}} \end{array}\right]. \quad \text{(A.0.5)}$$

Let us consider the partial derivative of the NURBS curve with respect to the ℓth control point.

$$\frac{\partial c^{(r)}(t,\mathbf{w},\mathbf{p})}{\partial p_\ell} \;=\; \frac{\partial}{\partial p_\ell}\left[\sum_{j=0}^{N_c-1} \mathcal{R}_{j,d}^{(r)}(t,\mathbf{w})\; p_j\right] = \sum_{j=0}^{N_c-1} \mathcal{R}_{j,d}^{(r)}(t,\mathbf{w}) \frac{\partial p_j}{\partial w_\ell} = \mathcal{R}_{\ell,d}^{(r)}(t,\mathbf{w}).$$

As a result, the first-order control point derivatives for a NURBS curve becomes:

$$\mathbf{D}_{\mathbf{p}}\left[c^{(r)}(t,\mathbf{w},\mathbf{p})\right] \;=\; \left[\mathcal{R}_d^{(r)}(t,\mathbf{w})\right]^T = \frac{\mathbf{w}^T \mathrm{diag}(\{\mathcal{B}_d^{(r)}(t)\})}{\left[\mathcal{B}_d^{(0)}(t)\right]^T \mathbf{w}} - \sum_{k=1}^{r} \begin{pmatrix} r \\ k \end{pmatrix} \frac{\left[\mathcal{B}_d^{(k)}(t)\right]^T \mathbf{w}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T \mathbf{w}} \left[\mathcal{R}_d^{(r-k)}(t,\mathbf{w})\right]^T$$

Partial derivatives with respect to weights

At this point, we will derive the first-order weight derivatives with respect to a NURBS curve

$$\mathbf{D}_{\mathbf{w}}\left[c^{(r)}(t,\mathbf{w},\mathbf{p})\right] = \left[\begin{array}{cccc} \frac{\partial c^{(r)}(t,\mathbf{w},\mathbf{p})}{\partial w_0} & \frac{\partial c^{(r)}(t,\mathbf{w},\mathbf{p})}{\partial w_1} & \cdots & \frac{\partial c^{(r)}(t,\mathbf{w},\mathbf{p})}{\partial w_{N_c-1}} \end{array}\right]. \quad \text{(A.0.6)}$$

Let us consider the partial derivative of the NURBS curve with respect to the ℓth weight

$$\frac{\partial c^{(r)}(t,\mathbf{w},\mathbf{p})}{\partial w_\ell} \;=\; \frac{\partial}{\partial w_\ell}\left[\sum_{j=0}^{N_c-1} \mathcal{R}_{j,d}^{(r)}(t,\mathbf{w})\; p_j\right] = \sum_{j=0}^{N_c-1} \frac{\partial \mathcal{R}_{j,d}^{(r)}(t,\mathbf{w})}{\partial w_\ell} p_j.$$

Expanding the partial derivative of the jth NURBS basis function with respect to ℓth weight we obtain

$$
\begin{aligned}
\frac{\partial \mathcal{R}_{j,d}^{(r)}(t,\mathbf{w})}{\partial w_\ell} &= \frac{\partial}{\partial w_\ell}\left[\frac{\mathcal{B}_{j,d}^{(r)}(t)\ w_j}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)\ w_i} - \sum_{k=1}^{r}\binom{r}{k}\frac{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(k)}(t)\ w_i}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)\ w_i}\ \mathcal{R}_{j,d}^{(r-k)}(t,\mathbf{w})\right] \\
&= \frac{\partial}{\partial w_\ell}\left[\frac{\mathcal{B}_{j,d}^{(r)}(t)\ w_j}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)\ w_i}\right] - \sum_{k=1}^{r}\binom{r}{k}\frac{\partial}{\partial w_\ell}\left[\frac{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(k)}(t)\ w_i}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)\ w_i}\right]\mathcal{R}_{j,d}^{(r-k)}(t,\mathbf{w}) \\
&\quad - \sum_{k=1}^{r}\binom{r}{k}\frac{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(k)}(t)\ w_i}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)\ w_i}\frac{\partial \mathcal{R}_{j,d}^{(r-k)}(t,\mathbf{w})}{\partial w_\ell} \\
&= -\frac{\mathcal{B}_{j,d}^{(r)}(t)}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)\ w_i}\left(\frac{\mathcal{B}_{\ell,d}^{(0)}(t)}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)\ w_i}w_j - \frac{\partial w_j}{\partial w_\ell}\right) \\
&\quad + \sum_{k=1}^{r}\binom{r}{k}\left(\frac{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(k)}(t)w_i}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)w_i}\frac{\mathcal{B}_{\ell,d}^{(0)}(t)}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)w_i} - \frac{\mathcal{B}_{\ell,d}^{(k)}(t)}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)w_i}\right)\mathcal{R}_{j,d}^{(r-k)}(t,\mathbf{w}) \\
&\quad - \sum_{k=1}^{r}\binom{r}{k}\frac{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(k)}(t)\ w_i}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)\ w_i}\frac{\partial \mathcal{R}_{j,d}^{(r-k)}(t,\mathbf{w})}{\partial w_\ell}.
\end{aligned}
$$

Then, the partial derivative of the NURBS curve with respect to the ℓth weight becomes

$$
\begin{aligned}
\frac{\partial c^{(r)}(t,\mathbf{w},\mathbf{p})}{\partial w_\ell} &= \sum_{j=0}^{N_c-1}\frac{\partial \mathcal{R}_{j,d}^{(r)}(t,\mathbf{w})}{\partial w_\ell}p_j \\
&= -\sum_{j=0}^{N_c-1}\left(\frac{\mathcal{B}_{j,d}^{(r)}(t)\ w_j\ p_j}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)\ w_i}\frac{\mathcal{B}_{\ell,d}^{(0)}(t)}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)\ w_i} - \frac{\mathcal{B}_{j,d}^{(r)}(t)\frac{\partial w_j}{\partial w_\ell}p_j}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)\ w_i}\right) \\
&\quad + \sum_{j=0}^{N_c-1}\left(\sum_{k=1}^{r}\binom{r}{k}\frac{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(k)}(t)w_i}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)w_i}\frac{\mathcal{B}_{\ell,d}^{(0)}(t)}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)w_i}\mathcal{R}_{j,d}^{(r-k)}(t,\mathbf{w})\ p_j\right) \\
&\quad - \sum_{j=0}^{N_c-1}\left(\sum_{k=1}^{r}\binom{r}{k}\frac{\mathcal{B}_{\ell,d}^{(k)}(t)}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)w_i}\mathcal{R}_{j,d}^{(r-k)}(t,\mathbf{w})p_j\right) \\
&\quad - \sum_{j=0}^{N_c-1}\left(\sum_{k=1}^{r}\binom{r}{k}\frac{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(k)}(t)\ w_i}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)\ w_i}\frac{\partial \mathcal{R}_{j,d}^{(r-k)}(t,\mathbf{w})}{\partial w_\ell}p_j\right)
\end{aligned}
$$

$$
\begin{aligned}
= & \frac{\sum_{j=0}^{N_c-1} \mathcal{B}_{j,d}^{(r)}(t) \frac{\partial w_j}{\partial w_\ell} p_j}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t)\ w_i} - \frac{\sum_{j=0}^{N_c-1} \mathcal{B}_{j,d}^{(r)}(t)\ w_j\ p_j}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t)\ w_i} \frac{\mathcal{B}_{\ell,d}^{(0)}(t)}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t)\ w_i} \\
& + \sum_{k=1}^{r} \begin{pmatrix} r \\ k \end{pmatrix} \frac{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(k)}(t) w_i}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t) w_i} \frac{\mathcal{B}_{\ell,d}^{(0)}(t)}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t) w_i} \sum_{j=0}^{N_c-1} \mathcal{R}_{j,d}^{(r-k)}(t, \mathbf{w})\ p_j \\
& - \sum_{k=1}^{r} \begin{pmatrix} r \\ k \end{pmatrix} \frac{\mathcal{B}_{\ell,d}^{(k)}(t)}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t) w_i} \sum_{j=0}^{N_c-1} \mathcal{R}_{j,d}^{(r-k)}(t, \mathbf{w}) p_j \\
& - \sum_{k=1}^{r} \begin{pmatrix} r \\ k \end{pmatrix} \frac{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(k)}(t)\ w_i}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t)\ w_i} \sum_{j=0}^{N_c-1} \frac{\partial \mathcal{R}_{j,d}^{(r-k)}(t, \mathbf{w})}{\partial w_\ell} p_j.
\end{aligned}
$$

In matrix form:

$$
\begin{aligned}
\frac{\partial c^{(r)}(t, \mathbf{w}, \mathbf{p})}{\partial w_\ell} = & \frac{\mathbf{e}_\ell^T \mathrm{diag}(\{\mathcal{B}_d^{(r)}(t)\})\mathbf{p}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T \mathbf{w}} - \frac{\mathcal{B}_{\ell,d}^{(0)}(t)}{\left[\mathcal{B}_d^{(0)}(t)\right]^T \mathbf{w}} \frac{\mathbf{w}^T \mathrm{diag}(\{\mathcal{B}_d^{(r)}(t)\})\mathbf{p}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T \mathbf{w}} \\
& + \sum_{k=1}^{r} \begin{pmatrix} r \\ k \end{pmatrix} \left(\frac{\left[\mathcal{B}_d^{(k)}(t)\right]^T \mathbf{w}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T \mathbf{w}} \frac{\mathcal{B}_{\ell,d}^{(0)}(t)}{\left[\mathcal{B}_d^{(0)}(t)\right]^T \mathbf{w}} - \frac{\mathcal{B}_{\ell,d}^{(k)}(t)}{\left[\mathcal{B}_d^{(0)}(t)\right]^T \mathbf{w}} \right) c^{(r-k)}(t, \mathbf{w}, \mathbf{p}) \\
& - \sum_{k=1}^{r} \begin{pmatrix} r \\ k \end{pmatrix} \frac{\left[\mathcal{B}_d^{(k)}(t)\right]^T \mathbf{w}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T \mathbf{w}} \frac{\partial c^{(r-k)}(t, \mathbf{w}, \mathbf{p})}{\partial w_\ell}.
\end{aligned}
$$

As a result, the first-order weight derivatives for a NURBS curve becomes:

$$
\begin{aligned}
\mathbf{D}_{\mathbf{w}} \left[c^{(r)}(t, \mathbf{w}, \mathbf{p}) \right] = & - \left[\left(\frac{\left[\mathcal{B}_d^{(0)}(t)\right] \mathbf{w}^T}{\left[\mathcal{B}_d^{(0)}(t)\right]^T \mathbf{w}} - \mathbf{I} \right) \frac{\mathrm{diag}(\{\mathcal{B}_d^{(r)}(t)\})\mathbf{p}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T \mathbf{w}} \right]^T \\
& + \sum_{k=1}^{r} \begin{pmatrix} r \\ k \end{pmatrix} \left\{ \left[\left(\frac{\left[\mathcal{B}_d^{(0)}(t)\right] \mathbf{w}^T}{\left[\mathcal{B}_d^{(0)}(t)\right]^T \mathbf{w}} - \mathbf{I} \right) \frac{\left[\mathcal{B}_d^{(k)}(t)\right]}{\left[\mathcal{B}_d^{(0)}(t)\right]^T \mathbf{w}} \right]^T c^{(r-k)}(t, \mathbf{w}, \mathbf{p}) \right. \\
& \left. - \frac{\left[\mathcal{B}_d^{(k)}(t)\right]^T \mathbf{w}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T \mathbf{w}} \mathbf{D}_{\mathbf{w}} \left[c^{(r-k)}(t, \mathbf{w}, \mathbf{p}) \right] \right\}.
\end{aligned}
$$

Lastly, let us compute the second-order weight derivatives of a NURBS curve:

$$\mathbf{D}_{\mathbf{ww}}\left[c^{(r)}(t,\mathbf{w},\mathbf{p})\right] = \begin{bmatrix} \frac{\partial^2 c^{(r)}(t,\mathbf{w},\mathbf{p})}{\partial w_0^2} & \frac{\partial^2 c^{(r)}(t,\mathbf{w},\mathbf{p})}{\partial w_1\ \partial w_0} & \cdots & \frac{\partial^2 c^{(r)}(t,\mathbf{w},\mathbf{p})}{\partial w_{N_c-1}\ \partial w_0} \\ \frac{\partial^2 c^{(r)}(t,\mathbf{w},\mathbf{p})}{\partial w_0\ \partial w_1} & \frac{\partial^2 c^{(r)}(t,\mathbf{w},\mathbf{p})}{\partial w_1^2} & \cdots & \frac{\partial^2 c^{(r)}(t,\mathbf{w},\mathbf{p})}{\partial w_{N_c-1}\ \partial w_1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 c^{(r)}(t,\mathbf{w},\mathbf{p})}{\partial w_0\ \partial w_{N_c-1}} & \frac{\partial^2 c^{(r)}(t,\mathbf{w},\mathbf{p})}{\partial w_1\ \partial w_{N_c-1}} & \cdots & \frac{\partial^2 c^{(r)}(t,\mathbf{w},\mathbf{p})}{\partial w_{N_c-1}^2} \end{bmatrix}. \tag{A.0.7}$$

Let us begin by considering the second-order partial derivative of the NURBS curve with respect to the nth and ℓth weight

$$\frac{\partial^2 c^{(r)}(t,\mathbf{w},\mathbf{p})}{\partial w_n \partial w_\ell} = \frac{\partial^2}{\partial w_n \partial w_\ell}\left[\sum_{j=0}^{N_c-1} \mathcal{R}_{j,d}^{(r)}(t,\mathbf{w})\ p_j\right] = \sum_{j=0}^{N_c-1} \frac{\partial^2 \mathcal{R}_{j,d}^{(r)}(t,\mathbf{w})}{\partial w_n \partial w_\ell} p_j$$

$$\begin{aligned} \frac{\partial^2 \mathcal{R}_{j,d}^{(r)}(t,\mathbf{w})}{\partial w_n \partial w_\ell} &= \frac{\partial}{\partial w_n}\left[\frac{\mathcal{B}_{j,d}^{(r)}(t)}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t)\ w_i}\frac{\partial w_j}{\partial w_\ell}\right] - \frac{\partial}{\partial w_n}\left[\frac{\mathcal{B}_{j,d}^{(r)}(t)\ w_j}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t)\ w_i}\frac{\mathcal{B}_{\ell,d}^{(0)}(t)}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t)\ w_i}\right] \\ &\quad - \sum_{k=1}^{r}\binom{r}{k}\frac{\partial}{\partial w_n}\left[\frac{\mathcal{B}_{\ell,d}^{(k)}(t)}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t) w_i}\mathcal{R}_{j,d}^{(r-k)}(t,\mathbf{w})\right] \\ &\quad - \sum_{k=1}^{r}\binom{r}{k}\frac{\partial}{\partial w_n}\left[\frac{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(k)}(t)\ w_i}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t)\ w_i}\frac{\partial \mathcal{R}_{j,d}^{(r-k)}(t,\mathbf{w})}{\partial w_\ell}\right] \\ &\quad + \sum_{k=1}^{r}\binom{r}{k}\frac{\partial}{\partial w_n}\left[\frac{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(k)}(t) w_i}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t) w_i}\frac{\mathcal{B}_{\ell,d}^{(0)}(t)}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t) w_i}\mathcal{R}_{j,d}^{(r-k)}(t,\mathbf{w})\right] \end{aligned}$$

$$\begin{aligned} \frac{\partial^2 \mathcal{R}_{j,d}^{(r)}(t,\mathbf{w})}{\partial w_n \partial w_\ell} &= \frac{\mathcal{B}_{j,d}^{(r)}(t)}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t)\ w_i}\left(\frac{\mathcal{B}_{\ell,d}^{(0)}(t)}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t)\ w_i}w_j - \frac{\partial w_j}{\partial w_\ell}\right)\frac{\mathcal{B}_{n,d}^{(0)}(t)}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t)\ w_i} \\ &\quad + \frac{\mathcal{B}_{j,d}^{(r)}(t)}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t)\ w_i}\left(\frac{\mathcal{B}_{n,d}^{(0)}(t)}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t)\ w_i}w_j - \frac{\partial w_j}{\partial w_n}\right)\frac{\mathcal{B}_{\ell,d}^{(0)}(t)}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t)\ w_i} \\ &\quad + \sum_{k=1}^{r}\binom{r}{k}\left(\frac{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(k)}(t) w_i}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t) w_i}\frac{\mathcal{B}_{\ell,d}^{(0)}(t)}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t) w_i} - \frac{\mathcal{B}_{\ell,d}^{(k)}(t)}{\sum_{i=0}^{N_c-1} \mathcal{B}_{i,d}^{(0)}(t) w_i}\right)\frac{\partial \mathcal{R}_{j,d}^{(r-k)}(t,\mathbf{w})}{\partial w_n} \end{aligned}$$

$$
\begin{aligned}
&+\sum_{k=1}^{r}\binom{r}{k}\left(\frac{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(k)}(t)\ w_i}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)\ w_i}\frac{\mathcal{B}_{n,d}^{(0)}(t)}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)\ w_i}-\frac{\mathcal{B}_{n,d}^{(k)}(t)}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)\ w_i}\right)\frac{\partial\mathcal{R}_{j,d}^{(r-k)}(t,\mathbf{w})}{\partial w_\ell}\\
&-\sum_{k=1}^{r}\binom{r}{k}\left(\frac{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(k)}(t)w_i}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)w_i}\frac{\mathcal{B}_{\ell,d}^{(0)}(t)}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)w_i}-\frac{\mathcal{B}_{\ell,d}^{(k)}(t)}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)w_i}\right)\frac{\mathcal{B}_{n,d}^{(0)}(t)}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)w_i}\mathcal{R}_{j,d}^{(r-k)}(t,\mathbf{w})\\
&-\sum_{k=1}^{r}\binom{r}{k}\left(\frac{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(k)}(t)w_i}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)w_i}\frac{\mathcal{B}_{n,d}^{(0)}(t)}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)w_i}-\frac{\mathcal{B}_{n,d}^{(k)}(t)}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)w_i}\right)\frac{\mathcal{B}_{\ell,d}^{(0)}(t)}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)w_i}\mathcal{R}_{j,d}^{(r-k)}(t,\mathbf{w})\\
&-\sum_{k=1}^{r}\binom{r}{k}\frac{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(k)}(t)\ w_i}{\sum_{i=0}^{N_c-1}\mathcal{B}_{i,d}^{(0)}(t)\ w_i}\frac{\partial^2\mathcal{R}_{j,d}^{(r-k)}(t,\mathbf{w})}{\partial w_n\partial w_\ell}.
\end{aligned}
$$

Then, the second-order partial derivative of the NURBS curve with respect to the nth and ℓth weights becomes

$$
\begin{aligned}
\frac{\partial^2 c^{(r)}(t,\mathbf{w},\mathbf{p})}{\partial w_n\partial w_\ell} &= \sum_{j=0}^{N_c-1}\frac{\partial^2\mathcal{R}_{j,d}^{(r)}(t,\mathbf{w})}{\partial w_n\partial w_\ell}p_j\\
&= \frac{\mathcal{B}_{n,d}^{(0)}(t)}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}\left(\frac{\mathcal{B}_{\ell,d}^{(0)}(t)}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}\frac{\mathbf{w}^T\mathrm{diag}(\{\mathcal{B}_d^{(r)}(t)\})\mathbf{p}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}-\frac{\mathbf{e}_\ell^T\mathrm{diag}(\{\mathcal{B}_d^{(r)}(t)\})\mathbf{p}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}\right)\\
&\quad+\frac{\mathcal{B}_{\ell,d}^{(0)}(t)}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}\left(\frac{\mathcal{B}_{n,d}^{(0)}(t)}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}\frac{\mathbf{w}^T\mathrm{diag}(\{\mathcal{B}_d^{(r)}(t)\})\mathbf{p}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}-\frac{\mathbf{e}_n^T\mathrm{diag}(\{\mathcal{B}_d^{(r)}(t)\})\mathbf{p}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}\right)\\
&\quad+\sum_{k=1}^{r}\binom{r}{k}\left(\frac{\left[\mathcal{B}_d^{(k)}(t)\right]^T\mathbf{w}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}\frac{\mathcal{B}_{\ell,d}^{(0)}(t)}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}-\frac{\mathcal{B}_{\ell,d}^{(k)}(t)}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}\right)\frac{\partial c^{(r-k)}(t,\mathbf{w},\mathbf{p})}{\partial w_n}\\
&\quad+\sum_{k=1}^{r}\binom{r}{k}\left(\frac{\left[\mathcal{B}_d^{(k)}(t)\right]^T\mathbf{w}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}\frac{\mathcal{B}_{n,d}^{(0)}(t)}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}-\frac{\mathcal{B}_{n,d}^{(k)}(t)}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}\right)\frac{\partial c^{(r-k)}(t,\mathbf{w},\mathbf{p})}{\partial w_\ell}\\
&\quad-\sum_{k=1}^{r}\binom{r}{k}\frac{\mathcal{B}_{n,d}^{(0)}(t)}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}\left(\frac{\left[\mathcal{B}_d^{(k)}(t)\right]^T\mathbf{w}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}\frac{\mathcal{B}_{\ell,d}^{(0)}(t)}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}-\frac{\mathcal{B}_{\ell,d}^{(k)}(t)}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}\right)c^{(r-k)}(t,\mathbf{w},\mathbf{p})\\
&\quad-\sum_{k=1}^{r}\binom{r}{k}\frac{\mathcal{B}_{\ell,d}^{(0)}(t)}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}\left(\frac{\left[\mathcal{B}_d^{(k)}(t)\right]^T\mathbf{w}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}\frac{\mathcal{B}_{n,d}^{(0)}(t)}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}-\frac{\mathcal{B}_{n,d}^{(k)}(t)}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}\right)c^{(r-k)}(t,\mathbf{w},\mathbf{p})\\
&\quad-\sum_{k=1}^{r}\binom{r}{k}\frac{\left[\mathcal{B}_d^{(k)}(t)\right]^T\mathbf{w}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}\frac{\partial^2 c^{(r-k)}(t,\mathbf{w},\mathbf{p})}{\partial w_n\partial w_\ell}
\end{aligned}
$$

.

Consequently, the second-order partial derivatives of a NURBS curve with respect to weights are

expressed as follows

$$
\begin{aligned}
\mathbf{D_{ww}}\left[c^{(r)}(t,\mathbf{w},\mathbf{p})\right] &= \left(\frac{\left[\mathcal{B}_d^{(0)}(t)\right]\mathbf{w}^T}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}} - \mathbf{I}\right)\frac{\mathrm{diag}(\{\mathcal{B}_d^{(r)}(t)\})\mathbf{p}\left[\mathcal{B}_d^{(0)}(t)\right]^T}{\left(\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}\right)^2} \\
&+ \left[\left(\frac{\left[\mathcal{B}_d^{(0)}(t)\right]\mathbf{w}^T}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}} - \mathbf{I}\right)\frac{\mathrm{diag}(\{\mathcal{B}_d^{(r)}(t)\})\mathbf{p}\left[\mathcal{B}_d^{(0)}(t)\right]^T}{\left(\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}\right)^2}\right]^T \\
&+ \sum_{k=1}^{r}\binom{r}{k}\left\{\left(\frac{\left[\mathcal{B}_d^{(k)}(t)\right]^T\mathbf{w}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}\left[\mathcal{B}_d^{(0)}(t)\right] - \left[\mathcal{B}_d^{(k)}(t)\right]\right)\right. \\
&\left(\frac{\mathbf{D_w}\left[c^{(r-k)}(t,\mathbf{w},\mathbf{p})\right]}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}} - \frac{\left[\mathcal{B}_d^{(0)}(t)\right]^T c^{(r-k)}(t,\mathbf{w},\mathbf{p})}{\left(\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}\right)^2}\right) \\
&+ \left(\frac{\mathbf{D_w}\left[c^{(r-k)}(t,\mathbf{w},\mathbf{p})\right]}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}} - \frac{\left[\mathcal{B}_d^{(0)}(t)\right]^T c^{(r-k)}(t,\mathbf{w},\mathbf{p})}{\left(\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}\right)^2}\right)^T \\
&\left(\frac{\left[\mathcal{B}_d^{(k)}(t)\right]^T\mathbf{w}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}\left[\mathcal{B}_d^{(0)}(t)\right] - \left[\mathcal{B}_d^{(k)}(t)\right]\right)^T \\
&\left.- \frac{\left[\mathcal{B}_d^{(k)}(t)\right]^T\mathbf{w}}{\left[\mathcal{B}_d^{(0)}(t)\right]^T\mathbf{w}}\mathbf{D_{ww}}\left[c^{(r-k)}(t,\mathbf{w},\mathbf{p})\right]\right\}.
\end{aligned}
$$

□

Bibliography

A. Barvinok. *A Course in Convexity*. American Mathematical Society, 2002. 17

J. T. Betts. Survey of numerical methods for trajectory optimization. *J. of Guidance, Control, and Dynamics*, 21(2):193–207, 1998. 1, 2, 53

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. 17

A. E. Bryson and Y.-C. Ho. *Applied Optimal Control*. Taylor and Francis, revised printing edition, 1975. 1, 2

B. Büeler, A. Enge, and K. Fukuda. Exact volume computation for polytopes: A practical study. *Polytopes — Combinatorics and Computation*, pages 131–154, 2000. 32, 119

R. H. Byrd, J. Nocedal, and R. A. Waltz. Knitro: An integrated package for nonlinear optimization. In *Large-Scale Nonlinear Optimization*, pages 35–59. Springer-Verlag, 2006. 7

J. F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, 1988. 4

L. Cesari. *Optimization - Theory and Application: Problems with Ordinary Differential Equations*. Springer-Verlag, 1983. 1

H. Chen and F. Allgower. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1217, 1998. 2

G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. *Lecture Notes in Computer Science*, 33:135–183, 1975. 4

D. A. Cox, J. B. Little, and D. O'Shea. *Using Algebraic Geometry*. Springer, second edition edition, 2005. 4

C. de Boor. *A Practical Guide to Splines*. Springer-Verlag, 1978. 8, 35, 41

M. M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*. PhD thesis, University of Heidelberg, 2001. 1

N. Faiz, S. K. Agrawal, and R. M. Murray. Trajectory planning of differentially flat systems with dynamics and inequalities. *J. of Guidance, Control and Dynamics*, 24(2):219–227, 2001. 3

M. Fliess, J. Levine, Ph. Martin, and P. Rouchon. Flatness and defect of nonlinear systems: introductory theory and examples. *Int. J. Control*, 61(6):1327–1361, 1995. 4, 5, 75, 78

K. Fukuda and A. Prodon. Double description method revisited. In *Combinatorics and Computer Science*, pages 91–111, 1995. 22, 119

P. F. Gath. *CAMTOS - A software suite combining Direct and Inderect Trajectory Optimization Methods*. PhD thesis, Universitat Stuttgart, 2002. 2

P. E. Gill, W. Murray, and M. A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–131, 2005. 7, 50

D. Kraft. On converting optimal control problems into nonlinear programming codes. *Computational Mathematical Programming*, F15, 1985. 2, 35

J. C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991. 4

N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996. 13

D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predicitve control: stability and optimality. *Automatica*, 36:789–814, 2000. 2

M. B. Milam. *Real-Time Optimal Trajectory Generation for Constrained Dyanamical Systems*. PhD thesis, California Institute of Technology, 2003. 1, 4, 8

R. Murray, editor. *Control in an Information Rich World: Report of the Panel on Future Directions in Control, Dynamics and Systems*. SIAM, 2003. 13

G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley and Sons, 1988. 3

Y. Pan, W. Peng, and X. Lu. A genetic algorithm on multi-sensor networks lifetime optimization. *Wireless Algorithms, Systems, and Applications*, 4130:295–306, 2006. 13

H. Pesch. Real-time computation of feedback controls for constrained optimal control problems. part 1: Neighboring extremals. *Optimal Control Applications and Methods*, 10:129–145, 1989. 1, 2

L. Piegle and W. Tiller. *The NURBS Book*. Springer-Verlag, second edition edition, 1997. 8, 41

L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mischchenko. *The Mathematical Theory of Optimal Processes.* Wiley-Interscience, 1962. 1, 2, 6

M. Rathinam. *Differentially Flat Nonlinear Control Systems.* PhD thesis, California Institute of Technology, 1997. 4

C. Tomlin. *Hybrid Control of Air Traffic Management Systems.* PhD thesis, University of California at Berkeley, 1998. 13

M. J. van Nieuwstadt. *Trajectory Generation for Nonlinear Control Systems.* PhD thesis, California Institute of Technology, 1996. 4

O. von Stryk and R. Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 37:357–373, 1992. 1, 2, 35

M. B. Wall. *A Genetic Algorithm for Resource-Constrained Scheduling.* PhD thesis, Massachusetts Institute of Technology, 1996. 13

R. Webster. *Convexity.* Oxford University Press Inc., 1994. 17

M. Wooldridge. *An Introduction to MultiAgent Systems.* John Wiley and Sons, 2002. 13