DISTRIBUTED COOPERATIVE TRAJECTORY GENERATION FOR MULTIPLE AUTONOMOUS VEHICLES USING PYTHAGOREAN HODOGRAPH BÉZIER CURVES

BY

RONALD CHOE

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Aerospace Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Doctoral Committee:

Prof. Naira Hovakimyan, Chair
Prof. Petros G. Voulgaris
Prof. Yuliy Baryshnikov
Prof. Angelia Nedić
Assoc. Prof. Srinivasa M. Salapaka

Abstract

This dissertation presents a framework for multi-vehicle trajectory generation that enables efficient computation of sets of feasible, collision-free trajectories for teams of autonomous vehicles executing cooperative missions with common objectives. Existing methods for multi-vehicle trajectory generation generally rely on discretization in time or space and, therefore, ensuring safe separation between the paths comes at the expense of an increase in computational complexity. On the contrary, the proposed framework is based on a three-dimensional geometric-dynamic approach that uses continuous Bézier curves with Pythagorean hodographs, a class of polynomial functions with attractive mathematical properties and a collection of highly efficient computational procedures associated with them. The use of these curves is critical to generate cooperative trajectories that are guaranteed to satisfy minimum separation distances, a key feature from a safety standpoint. By the differential flatness property of the dynamic system, the dynamic constraints can be expressed in terms of the trajectories and, therefore, in terms of Bézier polynomials. This allows the proposed framework to efficiently evaluate and, hence, observe the dynamic constraints of the vehicles, and satisfy mission-specific assignments such as simultaneous arrival at predefined locations.

The dissertation also addresses the problem of distributing the computation of the trajectories over the vehicles, in order to prevent a single point of failure, inherently present in a centralized approach. The formulated cooperative trajectory-generation framework results in a semi-infinite programming problem, that falls under the class of nonsmooth optimization problems. The proposed distributed algorithm combines the bundle method, a widely-used solver for nonsmooth optimization problems, with a distributed nonlinear programming method. In the latter, a distributed formulation is obtained by introducing local estimates of the vector of optimization variables and leveraging on a particular structure, imposed on the local minimizer of an equivalent centralized optimization problem.

Acknowledgments

I am deeply indebted to my advisor Professor Naira Hovakimyan for giving me the opportunity to pursue my Ph.D. degree under her guidance. This dissertation would not have been possible without her unconditional support and continuous belief in me, and is the result of her constant pursuit of intellectual rigor and perfection, while at the same time granting me much freedom in my research.

I would also like to express my gratitude to the other members of my doctoral committee, Professor Petros G. Voulgaris, Professor Yuliy Baryshnikov, Professor Angelia Nedić, and Professor Srinivasa M. Salapaka, for their time and invaluable feedback. I would especially like to thank Professor Angelia Nedić for the fruitful discussions we had on the topic of distributed optimization. My special thanks go to Professor Marko M. Mäkelä of the University of Turku, Finland, for sharing his implementation of the Multiobjective Proximal Bundle Method in Fortran/Matlab®.

Throughout the years in Urbana-Champaign I have met many wonderful people without whom my Ph.D. journey would not be complete. I would like to thank my past and present officemates, co-authors, and friends, Dapeng, Evgeny, Zhiyuan, Hui, Jan, Kwang-Ki, Steve, Venanzio, Bilal, Kasey, Javier, Helena, Niko, Jens, and Nanjun. I am especially grateful to Enric Xargay, for being the person I could always turn to for in-depth technical discussions, careful proof-reading, and help of any kind, but more importantly, for being a great friend. I am particularly indebted to Ji Young Kim for her continuous support, encouragement and care.

Finally, I would like to thank my family, Yoo Chong Choe, Chai Lian Ee, Yuh Meng Choe, and Yuh Huei Choe, for always giving me their unconditional support. They are the source of inspiration for everything I embark on in life. This dissertation is dedicated to them.

* This work was supported in part by the Air Force Office of Scientific Research, and NASA Langley Research Center.

Contents

List of Figures

List of Tables

Notation, Symbols, and Acronyms

$\hat{t}$	dimensionless time variable for parameterization
$\hat{x}$	local estimate of the vector x
$\mathbb{I}_n$	$n \times n$ identity matrix
$\mathbb{1}_n$	vector in $\mathbb{R}^n$ whose components are all 1
$\mathbf{p}$	three-dimensional position
$\text{card}(S)$	cardinality of set S
a	acceleration
E	minimum spatial clearance
J	optimization cost function
L	Laplacian of the graph $\mathcal{G}$
l	Lagrangian function
N	number of vehicles
n	degree of a Bézier polynomial
n_θ	degree of the polynomial that describes the timing law
T	overall final mission time
t	time variable for parametrization
v	speed
x	vector of optimization variables

conv S	convex hull of set S

Greek Symbols

α	linearization error of function f
β	linearization error of function g
$\dot{\psi}$	turn-rate
γ	flight-path angle
κ_c	curvature
ψ	course angle
ρ	subgradient of function g
τ	step size
τ_c	torsion
θ	timing law
Ξ	vector of optimization variables for the trajectory-generation framework
ξ	subgradient of function f
ζ	dimensionless parameter $\zeta \in [0, 1]$

Superscripts

i	vehicle i
f	final
i	initial

Subscripts

d	desired during the trajectory-generation phase
i	vehicle i

Acronyms

DFP	direction finding problem
GJK	Gilbert-Johnson-Keerthi
KKT	Karush-Kuhn-Tucker
PH	Pythagorean Hodograph
PS	pseudospectral
SIP	semi-infinite programming
UAV	unmanned aerial vehicle
UxS	unmanned systems

Notation

Given a vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^m$. The Jacobian $\nabla f(x)$ is defined as

$$\nabla f(x) := \begin{bmatrix} \nabla f_1(x) \\ \vdots \\ \nabla f_m(x) \end{bmatrix},$$

where $\nabla f_i(x)$ are the gradients of the functions $f_i : \mathbb{R}^n \to \mathbb{R}$ with $f_i \in C^1$ and given by

$$\nabla f_i(x) = \begin{bmatrix} \frac{\partial f_i(x)}{\partial x_1}, \cdots, \frac{\partial f_i(x)}{\partial x_n} \end{bmatrix}.$$

A vector v is a column vector, unless otherwise stated, and the ith component is denoted by v_i. The standard Euclidean norm is denoted by $\|v\|$, where $\|v\| = \sqrt{v^\top v}$. The notation $v \leq 0$ is used, if all components $v_i \leq 0$. The cross product of two vectors v and w is denoted by $v \times w$, while their dot product is given by $\langle v, w \rangle$. The Kronecker product of two matrices A and B is denoted by $A \otimes B$.

Chapter 1

Introduction

1.1 Background

Over the last decade, the use of unmanned systems (UxSs) has experienced an exponential growth, with applications in military reconnaissance and strike operations, border patrol missions, aerobiological sampling, forest fire detection, police surveillance, and recovery operations, to name but a few. With the development of novel algorithms that enable higher levels of autonomy, the type of missions that these UxSs execute has become increasingly more complex: simple single vehicle applications have evolved to complex multi-vehicle missions. Therefore, it is anticipated that future operations will require teams of heterogeneous systems working in cooperation to achieve common objectives, while being able to safely operate and execute coordinated tasks in highly uncertain areas [1]. The growing complexity of the envisioned mission scenarios poses several new challenges to the design and integration of UxSs, especially in terms of *autonomy* and *cooperation*.

A key enabling element for the successful realization of these cooperative missions is the availability of a cooperative control framework, that is conceptually summarized by the block diagram shown in Figure 1.1. The envisioned architecture offers a solution to the problem of cooperative control of multiple heterogeneous autonomous vehicles that must operate under strict spatial and temporal constraints, while ensuring collision-free maneuvers. The theoretical framework adopted borrows from various disciplines, and integrates algorithms for trajectory generation, path following, time-critical coordination, and collision avoidance. Together, these techniques yield a control architecture that allows meeting strict performance requirements in the presence of complex vehicle dynamics, communication constraints, and partial vehicle failures. Successful results in time-coordinated path following, both theoretical and experimental, are reported in [21,22,104,106] and references therein, while suitable algorithms for collision avoidance are presented in [20,21,64–66].

This dissertation addresses the cooperative trajectory-generation element of the envisioned cooperative control framework. Every mission, whether single or multi-vehicle, starts with the planning phase, during which trajectories are generated that meet the mission objectives. In the past, this simply meant planning

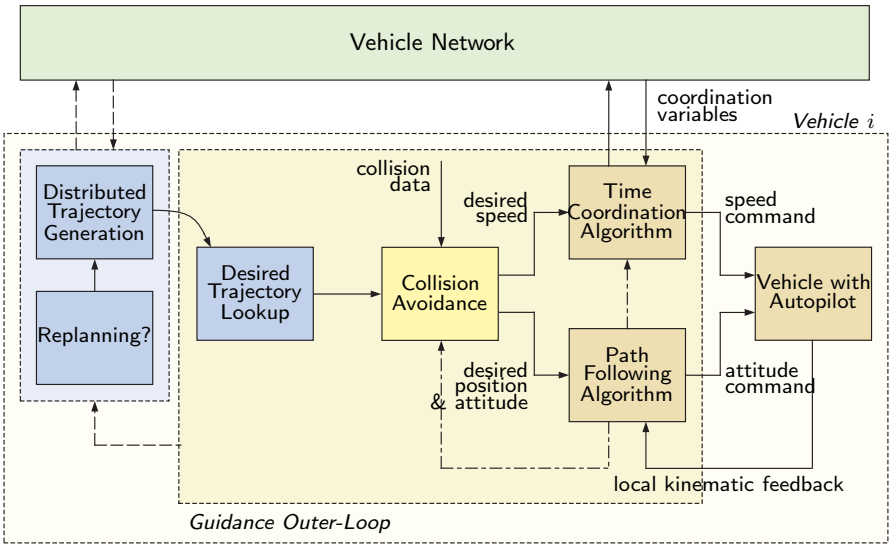


Figure 1.1: Conceptual architecture of the cooperative control framework adopted.

trajectories that would take the vehicle(s) along the points of interest. However, in modern-day operation of (teams of) autonomous vehicles, motion planning has grown considerably in complexity, requiring the trajectories to meet ever increasing number of constraints. Therefore, there is a growing need for efficient (cooperative) motion planning strategies that can be implemented onboard the vehicles. A planning algorithm has to work within a complex set of constraints, and (near) real-time generation of trajectories is desired to allow the vehicles to re-plan their trajectories, if necessary. More than often, the vehicles carry low power-consumption processors with limited memory in order to save weight for maximum payload capabilities. Therefore, the planning algorithm has to be computationally efficient.

Many novel approaches to (cooperative) path and trajectory planning have been reported in the literature. The vast majority of the methods can be classified as either *optimal control theory* or *geometric-dynamic* formulations. Methodologies of the former class find an optimal control input solution for the vehicle that generates desired trajectories for the states, whereas the geometric-dynamic approaches explicitly find a set of desired trajectories and rely on an additional onboard controller to track them.

Optimal control theory formulations have the benefit of obtaining a control input that minimizes a given cost function, while satisfying the dynamic constraints of the vehicles explicitly. However, in general, it is not possible to find a closed-form solution to the optimal control problem. Many popular approaches discretize the infinite-dimensional problem, in order to obtain a finite-dimensional problem that can be solved using various optimization solvers. For example, of special interest is the work that uses pseudospectral (PS)

optimal control theory, which has been applied successfully for solving trajectory and maneuver optimization problems; see, for example, [6, 9, 10, 30, 41] and references therein. Especially as a space technology, it has become the method of choice, after several high-profile successes, such as the execution of zero-propellant maneuver to reorient the International Space Station 90 degrees in November 2006 [5, 47].

As the number of vehicles or the duration of the missions increases, the size of the optimization problem will grow larger and, eventually, the computation of complete trajectories from start to endpoints becomes intractable and computationally very expensive. Receding horizon control —or Model Predictive Control (MPC)— aims to overcome this drawback by computing partial trajectories over a limited time horizon and, therefore, the trajectories are generated gradually over time as the mission unfolds. The work in [83] presents an MPC-based method for generating trajectories that avoid static and dynamic obstacles. A Sequential Quadratic Programming solver is used for the real-time implementation. Trajectory generation for multi-vehicle cooperative missions, based on MPC, is discussed in [73]. Depending on the constraints that are considered, several works, such as [12] and [84], frame the receding horizon control problem as a Mixed Integer Linear Programming problem. Lastly, an interesting development in this area is the work reported in [49], where a distributed cooperation algorithm is presented for problems with external disturbances and coupled hard state constraints that are non-convex.

Complementary to optimal control strategies are the geometric-dynamic formulations. In general, a purely geometric approach for the trajectory-generation problem is computationally efficient and requires only minimum computing power. However, the optimal trajectories may violate the dynamic constraints of the vehicle, since these are not taken into account during the planning phase. An extension of this class of methods are the geometric-dynamic formulations that handle dynamic constraints of the vehicles. In [18, 20, 53, 85, 86, 94, 98], the trajectories are described by Bézier curves and the dynamic constraints are imposed by bounding differential geometric properties of the path, such as curvature and torsion. As shown in the next section, this approach is less suitable for generating three-dimensional trajectories. Another interesting approach is to use the differential flatness property of the system, in order to compute and verify the inverse dynamics of the vehicle along a given trajectory [2, 67, 68, 107]. A similar approach is followed in [3], where the trajectory-generation problem for very agile multirotors is considered. A discretized kinematic model of the vehicles is considered to verify the acceleration and jerk along the generated trajectories. The authors use sequential convex programming that approximates non-convex constraints by convex counterparts. However, these approximations may lead to a sequence of over-constrained optimization problems and the algorithms may fail to find a feasible solution. Hence, the authors of [16] introduce an incremental sequential convex programming algorithm, where the constraints are added incrementally.

Finally, sampling-based search methods using Rapidly-exploring Random Trees algorithms were introduced in [50] and have gained ground in recent years. These motion planners, for example reported in [28,38] and references therein, are capable of quickly growing sets of feasible paths between vertices that are sampled from the entire environment.

The majority of motion planning algorithms are executed in a centralized way. This works well for single vehicle missions, as the vehicle does not require information that is not locally available and, in general, the resulting optimization problem is of a low order with a few constraints. However, in cooperative missions involving multiple vehicles, a centralized trajectory-planning approach has several severe drawbacks. First and foremost, the designated vehicle, tasked to generate the trajectories, becomes the single point of failure in the cooperative trajectory-generation framework. In the event of a complete loss of this particular vehicle, a sophisticated succession planning has to be in place, to designate the next vehicle in line to take over the task of trajectory generation. Nevertheless, it still remains to be seen, whether such a decision can be made in a timely matter, based on potentially limited information on the actual status of the failing vehicle. Secondly, in a centralized approach, the vehicle that is planning the trajectories needs to have access to the information of all vehicles in the network. This poses a heavy burden on the communication network, especially skewed towards a particular vehicle. And thirdly, the computational load is unevenly distributed over the number of vehicles, with one vehicle performing the bulk of computations. Hence, for large-scale multiple-vehicle missions, a distributed cooperative trajectory-generation framework is not only desired but in fact necessary.

Since every trajectory-generation problem results in an optimization problem, the key to distribution has to be sought in the field of distributed mulit-agent optimization. Excellent research in the area of distributed trajectory planning and, in particular, distributed optimization has been reported in the literature, for example, the work in [57–62,69–71,77,89–91,97,100]. In general, the global objective function is a combination of local objective functions that are only known to the corresponding agents. Each individual agent has access to its own local variables and constraints, and are able to exchange information only with its neighbors in a communication topology. The authors of [77] propose a simple decentralized algorithm to solve the optimization problem based on dual decomposition techniques. First, slack variables are introduced to decouple the objective function and, subsequently, the dual problem is formulated by dualizing all the constraints except the dynamic constraints. By doing so, the global (dual) problem can be replaced by a sequence of smaller problems that can be solved locally by each agent. A similar approach is followed in [97] that requires the vehicles to have access to all variables, including those local to non-neighboring vehicles. Hence, a multi-hop communication network is implemented and the algorithm is designed to handle communication delays.

A distributed consensus-based algorithm is proposed in [69, 70] to solve the unconstrained optimization problem, where the global objective function is the sum of local objective functions that are convex but not necessarily smooth. The algorithm uses subgradients since the objective functions are not necessarily differentiable. The results for constrained optimization problems are given in [71], where the local variables are assumed to lie in closed convex sets. The same constrained optimization problem, but with a noisy communication network, is considered in [90]. An interesting extension of the framework to min-max optimization problems is presented in [89, 91]. Another distributed consensus-like strategy is proposed in [100] for unconstrained convex optimization problems. The local objective functions in this approach are assumed to be twice continuously differentiable.

Different from consensus-based approaches is the work reported in [57–62]. The method uses standard nonlinear programming algorithms to solve (constrained) optimization problems in a distributed way and, hence, is applicable to a broad class of optimization problems. Nevertheless, the algorithm requires the problems to be smooth. The authors develop and analyze the proposed algorithm for equality and inequality constrained optimization problems in [58, 59, 62] and [60, 61], respectively.

1.2 Motivation behind the Approach

Motion planning and trajectory generation for vehicles have been extensively studied and many novel and mature methods exist in the literature. Every method has its own benefits and limitations, depending on the class of problems that they are applied to. We motivate our approach by evaluating existing methods of trajectory generation against the necessary requirements on the trajectory-generation framework, imposed by the time-critical cooperative multi-vehicle missions that we consider.

1.2.1 Collision-free Trajectories

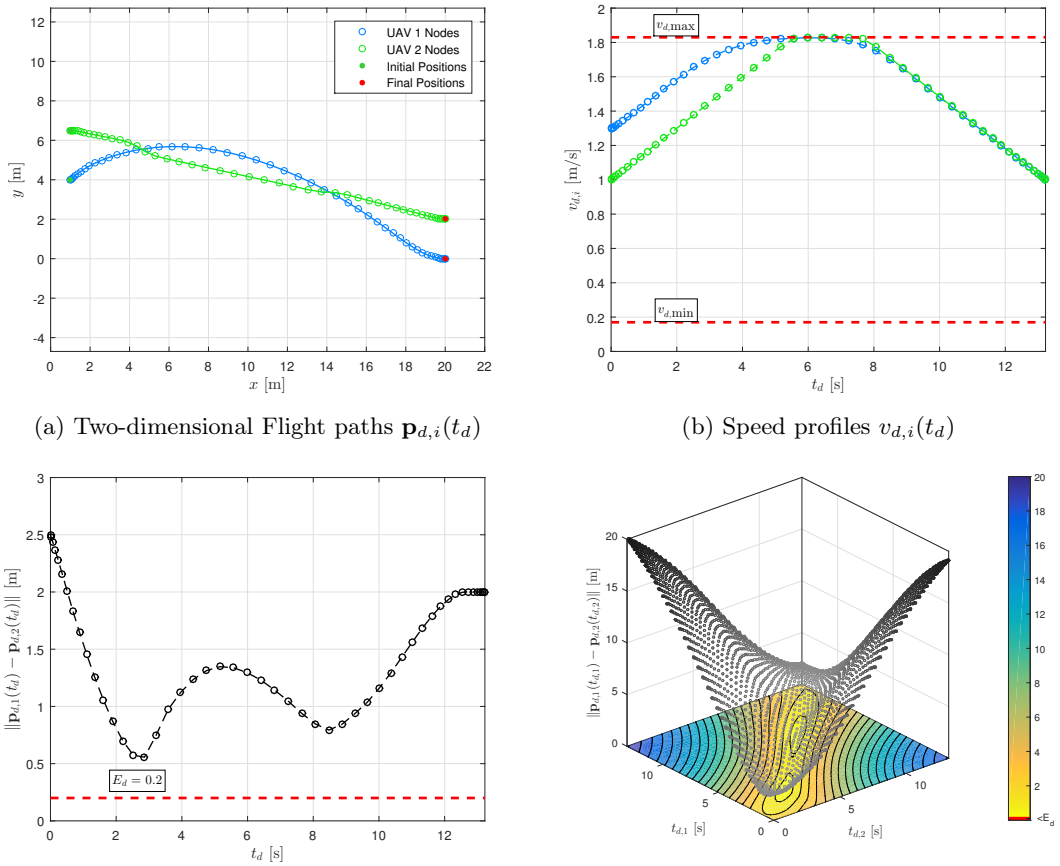
When dealing with single-vehicle missions, a collision-free trajectory means that the trajectory avoids all obstacles that are known a priori. In cooperative multi-vehicle missions, it is critical that, besides avoiding obstacles, the trajectories are deconflicted in space as well, so as to prevent inter-vehicle collisions. Deconfliction between trajectories can be guaranteed through *spatial separation* or *temporal separation*. Spatial separation is guaranteed if the minimum distance between any two points on the paths of the ith and jth vehicle is greater than or equal to the minimum spatial clearance. On the other hand, temporal separation is ensured if, for any time t, the minimum distance between the ith and jth vehicle is greater than or equal to the minimum spatial clearance.

Spatial and temporal separation ensure collision-free trajectories at all times, and the selection of the minimum spatial clearance is based on the performance of an underlying path-following controller onboard the vehicles. In addition, spatial deconfliction through temporal separation is also dependent on the performance of a necessary time-coordination controller, and on the quality and robustness of the communication network over which the vehicles exchange information with each other. Since the trajectories are allowed to intersect, a collision may potentially occur when the communication network is faulty or jammed. On the contrary, spatial separation results in a more conservative set of trajectories that may require a larger (air)space, but guarantees minimum spatial clearance at all times, even if the communication network is temporarily, or even permanently, unavailable.

The type of separation that is ultimately required depends on a variety of factors and considerations, such as the type of mission, the environment, the quality of the communication network, security requirements on data transmission, and has to be carefully decided on by the mission planner. Henceforth, the developed trajectory-generation framework has to be able to work with both types of deconfliction strategies. However, existing approaches to path planning that discretize the trajectories either in space or in time appear to be less suitable for guaranteeing spatial deconfliction through spatial separation. When discretizing the trajectories in one way or another, spatial separation between the trajectories can be ensured at the discretization nodes, but unfortunately, deconfliction is not guaranteed in between the nodes.

We illustrate this undesirable behavior through a simple example of a two-dimensional trajectory-generation problem, involving two unmanned aerial vehicles (UAVs) tasked to cooperatively execute a mission. The trajectories have to guarantee simultaneous arrival of vehicles, be *spatially separated* and, moreover, observe the dynamic constraints of the vehicles. This simple trajectory-generation problem can be solved using PS optimal control theory, a method based on a carefully chosen set of nodes for which the optimal control problem is solved. Most of the available literature on PS optimal control theory addresses the optimization of trajectories involving a single vehicle. Within the PS optimal control theory framework, to the best of our knowledge, only the work in [9] presents a solution to the problem of cooperative trajectory generation for two vehicles that must execute time-critical, collision-free maneuvers. The results reported in the paper demonstrate that PS optimal control theory can be used to efficiently generate trajectories that maintain a minimal separation between the two cooperating vehicles, while at the same time satisfying their dynamic constraints. However, for the particular problem in [9], the authors impose trajectory deconfliction through *temporal separation*, rather than *spatial separation* that is required in our example.

Figure 1.2 shows the solution to the aforementioned illustrative problem, where we impose deconfliction through *spatial separation*, obtained using DIDO [79], a Matlab® software package for solving PS optimal



(a) Two-dimensional Flight paths $\mathbf{p}_{d,i}(t_d)$

(b) Speed profiles $v_{d,i}(t_d)$

(c) Vehicle separation

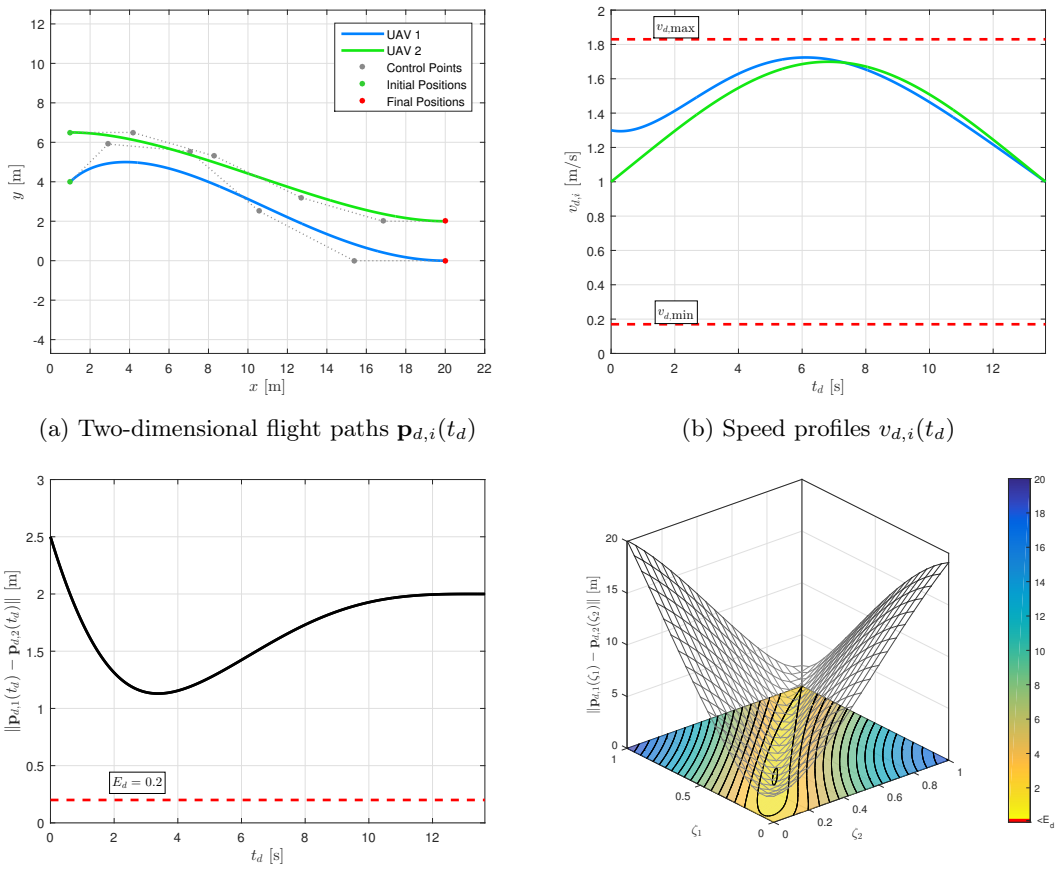
(d) Path separation UAV 1 and 2

Figure 1.2: Two-dimensional trajectories for a team of two cooperating UAVs using pseudospectral optimal control theory. E_d is the required minimum spatial clearance. The number of nodes is $n = 50$.

control problems. It can be seen that *at the nodes*, denoted by the open circles, the solution is feasible, since the dynamic constraints (Figure 1.2b), and the minimum spatial separation between the two paths (Figure 1.2d) are not violated. However, the *continuous* trajectories are by no means spatially separated as required, and are in fact intersecting, as shown in Figure 1.2a.

To avoid violation of the minimum separation requirement in between the nodes, the number of nodes can be increased. However, this considerably affects the spatial and temporal scalability of the method. For example, if the number of nodes increases proportionally, then the number of deconfliction constraints alone will increase quadratically. This problem does not limit itself to PS optimal control theory, but is characteristic to methods that are based on discretization. For instance, the authors of [3] and [16] recognize this issue as one of the limitations of their approaches, where the trajectories in both methodologies are discretized in time.

For comparison, the same illustrative two-dimensional trajectory-generation problem is solved using the



(a) Two-dimensional flight paths $\mathbf{p}_{d,i}(t_d)$

(b) Speed profiles $v_{d,i}(t_d)$

(c) Vehicle separation

(d) Path separation UAV 1 and 2

Figure 1.3: Two-dimensional trajectories for a team of two cooperating UAVs using proposed trajectory-generation framework. E_d is the required minimum spatial clearance.

approach proposed in this dissertation, that does not discretize the trajectories in time nor in space. The results are presented in Figure 1.3, and it is clear that the solution is indeed feasible. As we shall see in subsequent chapters, our formulation results in a nonsmooth optimization problem, which can be addressed by nonsmooth optimization methods, such as bundle methods [48, 56].

1.2.2 Dynamic Constraints

A purely geometric approach for the trajectory-generation problem avoids discretization by fitting a smooth curve between the two desired endpoints that is subject to constraints with regards to obstacles. Although these algorithms are very efficient and require only minimum computing power, the optimal trajectories may violate the dynamic constraints of the vehicle, since these are not taken into account during the planning phase. Several mixed geometric-dynamic methodologies have been developed, where geometric formulations also incorporate dynamic constraints. A common practice is to relate the dynamics of the vehicle to differen-

tial geometric properties of the path. For example, in [53, 85, 86, 94, 98] the dynamic constraints are imposed by bounding differential geometric properties of the path, such as curvature κ_c and torsion τ_c. In [18, 20], collision-free planar trajectories are generated that satisfy the dynamic constraints of the vehicles by means of bounding the maximum curvature of the path. The results show that this approach works well for the two-dimensional trajectory-generation problem.

The authors of [85, 86, 98] use a similar notion to extend their framework to three-dimensional trajectories. In addition to the curvature, bounds are also imposed on the torsion of the spatial path. The main drawback of using differential geometric properties to satisfy the dynamic constraints becomes evident from the definitions of both the curvature and the torsion of a three-dimensional curve $\mathbf{r}(\zeta)$:

$$\kappa_c(\zeta) = \frac{\|\mathbf{r}'(\zeta) \times \mathbf{r}''(\zeta)\|}{\|\mathbf{r}'(\zeta)\|^3}, \qquad \tau_c(\zeta) = \frac{\langle\, \mathbf{r}'(\zeta) \times \mathbf{r}''(\zeta)\,,\, \mathbf{r}'''(\zeta)\,\rangle}{\|\mathbf{r}'(\zeta) \times \mathbf{r}''(\zeta)\|^2},$$

where $\mathbf{r}'(\zeta)$, $\mathbf{r}''(\zeta)$, and $\mathbf{r}'''(\zeta)$ are the first, second, and third derivatives of $\mathbf{r}(\zeta)$ with respect to ζ, respectively. It can be observed that the torsion τ_c is not well-behaved and, in fact, has a singularity for curves where the curvature κ_c is (momentarily) zero. These include curves exhibiting inflection points, but also straight paths (or segments). Therefore, bounding the torsion is overly restrictive and highly undesirable, since that excludes, among others, the simplest family of curves, namely that of straight lines.

1.2.3 A Computationally Efficient Geometric-Dynamic Approach

This dissertation presents a cooperative trajectory-generation framework that aims to overcome the shortcomings described above and is based on the following key ideas:

i. the desired trajectory is generated for the entire mission without discretizing the trajectory temporally or spatially;

ii. the dynamic constraints are imposed explicitly and not via differential geometric properties of the paths;

iii. a specific family of polynomials is used that has favorable geometric and mathematical properties; and

iv. the desired trajectory is decomposed into a geometric and a temporal element.

Hence, we present a framework where trajectories for multiple cooperating vehicles are generated efficiently; moreover, the trajectories can be deconflicted through either *temporal* or *spatial* separation. This is achieved by using a specific family of curves to describe the trajectories, namely *Pythagorean Hodograph (PH)*

Bézier curves. The main motivations for using PH Bézier curves, instead of polynomials expressed in the monomial (or power) basis, are

i. the existence of computationally efficient algorithms to compute, for example, the minimum distance between two (rational) Bézier curves [14], or the extrema of an explicit (rational) Bézier curve [17]; and

ii. the existence of a closed-form solution for the arc lengths of the paths.

Bézier curves are widely used in computer graphics, animations, and type fonts such as postscript fonts and true type fonts. The curves were invented by Dr. Pierre Bézier in the early 1960s to aid the engineers and artists in the shape design of the cars at the Renault company in France. A Bézier curve is completely determined by a set of *control points*. The curve starts at the first and ends at the last control point, while the curve lies completely in the *convex hull* containing the set of control points. This latter property is extremely useful when path planning for aircraft systems is considered, as the airspace in which an aircraft is allowed to maneuver is often allocated and limited by strict boundaries. An example of spatially deconflicted paths generated for a bounded airspace using Bézier curves can be found in [53]. Enforcing a Pythagorean structure to the hodograph of a Bézier curve results in a PH Bézier curve and allows for an exact determination of its arc length.

In [85,86,98], PH Bézier curves were generated for multi-UAV missions with simultaneous times-of-arrival, such that the paths were of equal lengths while the desired speeds were constant and equal for all UAVs. Though this is a convenient way to ensure simultaneous arrival and temporal separation, these constraints on the path lengths and speed profiles are very restrictive. It will become clear that, in our trajectory-generation framework, the path lengths and the speed profiles are not restricted to such constraints and, hence, offer greater flexibility. This is achieved by using a timing law that facilitates the decomposition of a desired trajectory into a geometric and a temporal element. The concept of introducing a timing law in order to adjust the spatial path and the speed profile independently was first described in [95], and later applied in [42, 96, 107]. In [95] a separate reference function was used for the speed profile, whereas in [107], the speed profile was obtained by integrating the acceleration equation using a predetermined thrust history.

Lastly, the control points of Bézier curves also allow for intuitive (re)design of their shapes. In [64–66], an algorithm is presented where collision avoidance is achieved through proper relocation of the control points of the Bézier curves.

1.2.4 Distributed Optimization: Nonsmooth Approach

The objective of this part of the dissertation is to develop a distributed optimization algorithm that allows to generate trajectories in a distributed fashion within the proposed cooperative trajectory-generation framework. By using continuous curves to describe the trajectories, our trajectory-generation framework results in a semi-infinite programming problem. To the best of our knowledge, there do not exist any viable distributed semi-infinite programming methods to date. Nevertheless, we are able to formulate the semi-infinite programming as an ordinary (finite-dimensional) nonsmooth optimization problem, for which several distributed algorithms exist. These techniques are based on the subgradient of a nonsmooth convex function and, thus, require the problem to be convex.

Since the cooperative trajectory-generation problem is nonsmooth and nonconvex, most of the existing work on distributed optimization is not applicable in one way or another. Therefore, our proposed approach uses centralized nonsmooth optimization methods and combines them with existing techniques for distributing the computations over the vehicles.

1.3 Overview

Within the context of geometric-dynamic approaches, this dissertation presents the formulation of a cooperative three-dimensional trajectory-generation framework that uses PH Bézier curves. The setup builds on the work presented in [18, 20], where we addressed the planar case. In these previous works, the trajectories satisfied the dynamic constraints of the vehicles and, moreover, were deconflicted in space. It was shown that Bézier curves can reduce the computational load, by exploiting their favorable geometric properties. Besides extending the previous results to three-dimensional trajectories, we introduce a less conservative approach in satisfying the dynamic constraints, that does not limit the set of admissible curves that satisfy the spatial and temporal specifications. Moreover, we also use a different timing law than the one proposed in [18, 20], which allows to ensure either *temporal* or *spatial* separation of the trajectories. Therefore, the approach presented in this dissertation offers more flexibility in generating collision-free trajectories. To distribute the trajectory generation over the vehicles, we present a theoretical framework of a distributed solver for nonsmooth optimization problems, suitable for application to the cooperative trajectory-generation framework. The dissertation is organized as follows:

Chapter 2 formulates the problem of trajectory generation for multiple vehicles. The timing law is introduced, that decomposes a trajectory into a geometric element and a temporal element. The timing law allows the speed to be adjusted independently, without altering the spatial paths along which these vehicles

travel. The chapter also derives the dynamic constraints expressed in terms of the trajectories, by using the differential flatness property of the vehicle's dynamics. Finally, the chapter gives a formal expression for the different types of spatial deconfliction.

Chapter 3 gives a brief overview of Pythagorean Hodograph Bézier curves. These are Bézier curves of which the hodograph has a Pythagorean structure. First, the properties of generic Pythagorean Hodograph curves are discussed, followed by a short summary on Bézier curves.

Chapter 4 presents the development of the cooperative trajectory-generation framework, by incorporating PH Bézier curves into the trajectory-generation framework from Chapter 2. The cooperative trajectory-generation problem results in a nonconvex constrained optimization problem. Illustrative examples are shown where desired trajectories are generated for a team of cooperating vehicles that is tasked to execute a time-critical mission. The numerical results are obtained by employing standard Matlab® optimization tools, and are provided to show the feasibility of the proposed approach. The theoretical development of customized distributed optimizers to efficiently solve the resulting optimization problem is postponed till Chapter 6.

Chapter 5 characterizes the optimization problem resulting from the cooperative trajectory-generation framework. It is explained that the constrained optimization is not only nonconvex, but in fact belongs to a class of nonsmooth optimization problems. Hence, a first step towards a distributed approach in solving the trajectory-generation problem, is to familiarize with nonsmooth optimization theory. This chapter continues by providing a short summary on nonsmooth analysis and nonsmooth optimization theory; essential tools that will be needed in the next chapter on the development of nonsmooth distributed optimization algorithms.

Chapter 6 presents a theoretical framework of a distributed algorithm for solving nonsmooth optimization problems. The nonsmooth functions are assumed of a particular form, conform those appearing in the cooperative trajectory-generation framework as discussed in Chapter 4. The approach is based on bundle methods, widely used to solve nonsmooth optimization problems, in combination with a distributed nonlinear programming method. The latter uses centralized algorithms, but by leveraging on a structure that is imposed on the solution of an equivalent problem, the algorithm results in a distributed formulation.

Chapter 7 gives the conclusions and touches on potential extensions of the work presented in this dissertation.

Chapter 2

General Trajectory-Generation Framework

In this chapter, we formulate the problem of trajectory generation for multiple vehicles and introduce the *timing law* that decomposes a trajectory into a geometric element and a temporal element. Subsequently, mathematical definitions are given of the set of constraints that has to be satisfied in order for the desired trajectories to be *feasible*. Following the terminology in [98], trajectories that satisfy the dynamic constraints of the vehicles are called *flyable* trajectories. The trajectories are said to be *feasible* when they are also deconflicted in space by satisfying additional spatial or temporal separation constraints. Hence, the set of constraints consists of mission-specific constraints, dynamic constraints of the vehicles, and inter-vehicle safety distance requirements.

2.1 Problem Formulation

In the three-dimensional trajectory-generation problem, the objective is to generate N feasible trajectories $\mathbf{p}_{d,i}(t_d)$

$$\mathbf{p}_{d,i} : [0, t^{\mathrm{f}}_{d,i}] \to \mathbb{R}^3 \qquad i = 1, 2, \ldots, N\,, \tag{2.1}$$

where N is the number of vehicles, $t^{\mathrm{f}}_{d,i} \in \mathbb{R}^+$ are the individual final mission times of the vehicles, and $t_d \in [0, T_d\,]$, with $T_d := \max\{t^{\mathrm{f}}_{d,1}, \ldots, t^{\mathrm{f}}_{d,N}\}$, is the time variable used during the trajectory-generation phase. Note that the mission for vehicle i has terminated for all $t_d > t^{\mathrm{f}}_{d,i}$. In general, the variable t_d does not progress at the same rate as the actual mission (clock) time t if, for example, time coordination among the vehicles is desired [20]. The feasible trajectories $\mathbf{p}_{d,i}(t_d)$ together minimize a global cost function $J(\cdot)$ and satisfy boundary conditions, spatial constraints, and temporal constraints. Spatial and temporal constraints can be mission-specific, such as simultaneous arrival of the vehicles, but can also be related to the dynamics of the vehicles, for example adhering to the maximum speed and acceleration.

At this point it is important to distinguish clearly between a (spatial) *path* and a *trajectory*. A spatial path is a curve in space, that is parameterized by a (dimensionless) variable defined on an arbitrary interval. A

spatial path specifies the location of the vehicle, without imposing any *temporal* specifications. A trajectory is also a path along which a vehicle travels through space, however, it is defined as a function of *time* (Equation (2.1)) and, hence, describes the (desired) position of the vehicle at any point in time. It is clear that, besides the spatial path, a trajectory also contains information about the (desired) speed profile $v_{d,i}$ with which the vehicle moves along this path.

Therefore, instead of generating the trajectories explicitly as a function of time, we first decompose the trajectory into a *spatial path*, a geometric element with no temporal specifications, and a *timing law* associated with this path, that captures the temporal assignments of the trajectory. To this purpose, first, we introduce a dimensionless parameter ζ_i. For convenience, we let $\zeta_i \in [0, 1]$; as it will become clear later, this is a natural choice since we will be working with Bézier curves that are defined on the interval $[0, 1]$. Then, the following map defines the spatial path $\mathbf{p}_{d,i}(\zeta_i)$:

$$\mathbf{p}_{d,i} : [0, 1] \to \mathbb{R}^3 \qquad \zeta_i \in [0, 1]\,, \quad i = 1, 2, \ldots, N. \tag{2.2}$$

Next, in order to reconstruct the spatial trajectory $\mathbf{p}_{d,i}(t_d)$, the dimensionless parameter ζ_i needs to be related to the time t_d. This is provided by the timing law $\theta_i(\cdot)$. The timing law dictates how the variable ζ_i for the ith vehicle evolves with the time variable t_d and, as such, affects the desired rate at which the vehicle moves along the path. Hence, the timing law offers a means to meet the temporal requirements of the mission. Let the timing law $\theta_i(\cdot)$ be defined through a dynamic relation of the form

$$\theta_i(\cdot) = \frac{\mathrm{d}\zeta_i}{\mathrm{d}t_d}\,,$$

where $\theta_i(\cdot)$ is a positive function, smooth in its arguments. This function will be defined in the subsequent section; however, it is important to note that the timing law $\theta_i(\cdot)$ will be chosen such that an analytical expression for the function $\zeta_i(t_d)$ exists. This is highly desirable, as the map $\zeta_i(t_d)$ will allow us to relate the time variable t_d to the parameter ζ_i, with which the desired position $\mathbf{p}_{d,i}$ of the ith vehicle at time t_d can be found through the map in Equation (2.2).

Now, the cooperative trajectory-generation problem can be defined as follows in terms of the three-dimensional spatial paths $\mathbf{p}_{d,i}$ and the corresponding timing laws $\theta_i(\cdot)$:

Definition 1 (Cooperative Trajectory-Generation Problem) *Find N pairs of*

1. *three-dimensional spatial paths $\mathbf{p}_{d,i}(\zeta_i)$, that are conveniently parameterized by a dimensionless variable $\zeta_i \in [0, 1]$, and*

2. *corresponding timing laws $\theta_i(\cdot)$, appropriately defined such that the functions $\zeta_i(t_d)$ can be expressed analytically,*

which together minimize a cost function $J(\cdot)$, satisfy desired boundary conditions, do not violate the dynamic constraints of each vehicle, ensure that the vehicles maintain a prespecified spatial clearance, and satisfy predefined mission-specific constraints.

Given the preceding problem formulation, the trajectory-generation framework can be cast into a constrained optimization problem where a set of desired trajectories are obtained by minimizing the cost function $J(\cdot)$. For computational efficiency, we will describe the spatial paths $\mathbf{p}_{d,i}(\cdot)$ and timing laws $\theta_i(\cdot)$ by real polynomials. The optimization problem can be formulated as follows:

$$\begin{aligned} &\min_{\substack{\mathbf{p}_{d,i}\in\mathcal{P}\\ \theta_i\in\Theta\\ i=1,\dots,N}} \; J(\cdot) \\ &\text{subject to} \quad \text{boundary conditions,} \\ &\qquad\qquad\;\; \text{dynamic constraints of the vehicles,} \\ &\qquad\qquad\;\; \text{mission-specific constraints,} \\ &\qquad\qquad\;\; \text{minimum separation constraints,} \end{aligned} \tag{2.3}$$

where the set $\mathcal{P}$ is the set of degree n polynomial curves, the set Θ denotes the set of degree n_θ polynomial curves, and $J(\cdot)$ is a given cost function and may include terms related to mission-specific goals. The main challenge is to solve this multi-vehicle optimization problem in (near) real-time, with a possibility that the curse of dimensionality phenomena arises as the number of vehicles increases. We present an approach to reduce the computation time of the trajectory-generation algorithm by exploiting the mathematical and geometric properties of a particular class of curves, known in the literature as Pythagorean Hodograph Bézier Curves.

In the remainder of this chapter, we will discuss the timing law and the set of constraints, that together define the trajectory-generation framework.

2.2 Definition of the Timing Law

The timing law allows us to independently adjust the speed of the vehicles, without altering the spatial paths along which these vehicles travel. In other words, we can assign different speed profiles $v_{d,i}(\cdot)$ to a given spatial path $\mathbf{p}_{d,i}(\zeta_i)$ by changing the parameters of the timing law. This can be easily illustrated by

the expression of the speed profile. The speed profile $v_{d,i}(\cdot)$ can be derived as follows:

$$v_{d,i}(\zeta_i, \theta_i) = \left\| \frac{\mathrm{d}\mathbf{p}_{d,i}(\zeta_i)}{\mathrm{d}t_d} \right\| = \left\| \frac{\mathrm{d}\mathbf{p}_{d,i}(\zeta_i)}{\mathrm{d}\zeta_i} \frac{\mathrm{d}\zeta_i}{\mathrm{d}t_d} \right\| = \theta_i(\cdot) \left\| \mathbf{p}'_{d,i}(\zeta_i) \right\| , \qquad \zeta_i \in [0,1], \tag{2.4}$$

where $\|\cdot\|$ denotes the Euclidean norm and the first parametric derivative of the spatial path $\mathbf{p}_{d,i}(\zeta_i)$ is defined as $\mathbf{p}'_{d,i}(\zeta_i) = \mathrm{d}\mathbf{p}_{d,i}(\zeta_i)/\mathrm{d}\zeta_i$. The last equality is obtained by noting that the timing law is a positive function, that is, $\theta_i(\cdot) > 0$. From Equation (2.4) it is clear that, with a timing law other than $\theta_i(\cdot) \equiv 1$, the speed profile can be chosen independently from the spatial path $\mathbf{p}_{d,i}(\zeta_i)$, in order to meet the temporal specifications.

In our approach we seek to find a suitable structure and parametrization for the timing law, such that the temporal constraints are met and an analytical expression for the function $\zeta_i(t_d)$ exists. Taking into consideration that the spatial paths $\mathbf{p}_{d,i}(\zeta_i)$ are described by polynomials for computational efficiency, from Equation (2.4) it can be seen that the timing laws $\theta_i(\cdot)$ have to be polynomial functions as well, in order to maintain the polynomial structure for the speed profiles. However, there are two feasible approaches to parametrize the timing laws.

First, let the timing law $\theta_i(\cdot)$ be a smooth positive polynomial function of the parameter ζ_i, that is

$$\theta_i(\zeta_i) = \frac{\mathrm{d}\zeta_i}{\mathrm{d}t_d} , \qquad \zeta_i \in [0,1] . \tag{2.5}$$

The functions $t_{d,i}(\zeta_i)$ that map the parameter ζ_i to the time variable t_d can be found through integration of (2.5):

$$t_{d,i}(\zeta_i) = \int_0^{\zeta_i} \frac{1}{\theta_i(\tau)} \mathrm{d}\tau. \tag{2.6}$$

In general, the timing laws $\theta_i(\zeta_i)$ are distinct, and hence, the time *functions* $t_{d,i}(\zeta_i)$ are different for each vehicle as well. However, the time *variable* t_d, which denotes the time at which the mission is progressing, is a common parameter and available to *all* vehicles. Therefore, it is desirable to derive an analytical expression for the map $\zeta_i(t_d)$, since the trajectories are parameterized by the variable ζ_i. It can be shown that, with a proper choice of the timing law, an analytical solution exists for the integral in Equation (2.6). From the definition of $\theta_i(\zeta_i)$ it follows that $t_{d,i}(\zeta_i)$ is a strictly monotonically increasing function, and thus its inverse function $t_{d,i}^{-1} = \zeta_i(t_d)$ exists. Another advantage is that the temporal constraints are explicitly parametrized by ζ_i. For example, the desired speed profile, given by Equation (2.4), will be a function of the parameter ζ_i only.

A preliminary *planar* trajectory-generation framework is presented in [18, 20], where the timing law θ_i

was chosen as a second degree polynomial in ζ_i, so that Equation (2.6) allowed for a solution in closed-form *and* the inverse function $\zeta_i(t_d)$ could be derived analytically as well. Following this approach, in combination with Bézier curves to describe the paths $\mathbf{p}_{d,i}(\zeta_i)$ and the timing laws $\theta_i(\zeta_i)$, the results in [18, 20] indicate that the developed framework is able to generate efficiently a set of collision-free trajectories that satisfy all boundary conditions and dynamic constraints of the vehicles. However, there are three major drawbacks in this particular choice of structure for the timing law:

1. The time integral (2.6) admits a closed-form solution $t_{d,i}(\zeta_i)$ for a polynomial function $\theta_i(\zeta_i)$ of any degree $n_\theta \in \mathbb{N}$. However, for $n_\theta > 2$ it becomes extremely intractable or even impossible to obtain an analytical expression for the inverse function $\zeta_i(t_d)$. Therefore, the only practically feasible choice for the timing law is a second-degree polynomial. Unfortunately, this leaves the framework with no room for increasing n_θ in scenarios where more free variables are necessary in order to satisfy additional (temporal) constraints.

2. In the case where $n_\theta = 2$, the solutions to the time integral (2.6) are either trigonometric or hyperbolic functions [18, 20] and, hence, the inverse functions $\zeta_i(t_d)$ are not polynomials. This poses a problem when spatial deconfliction has to be ensured through *temporal* separation, that is the minimum distance between two points on the paths of the ith and jth vehicle at time t_d, is greater than or equal to the minimum spatial clearance E:

$$\min_{\substack{i,j=1,\dots,N \\ i\neq j}} \left\|\mathbf{p}_{d,i}(\zeta_i(t_d)) - \mathbf{p}_{d,j}(\zeta_j(t_d))\right\|^2 \geq E^2\,, \quad \forall\, t_d \in [0, t^{\mathrm{f}}_{d_{ij}}],$$

   where $t^{\mathrm{f}}_{d_{ij}} = \min\{t^{\mathrm{f}}_{d,i}, t^{\mathrm{f}}_{d,j}\}$. Recalling the fact that the spatial paths $\mathbf{p}_{d,i}(\zeta_i)$ are chosen to be polynomials for computational efficiency, the above equation becomes impractical to evaluate if the functions $\zeta_i(t_d)$ are not of polynomial form as well.

3. The timing law $\theta(\zeta_i)$ has to be strictly positive for the map $\zeta_i(t_d)$ to exist. This poses a problem for vehicles that are able to remain stationary, such as hovering multirotors. In these cases, it can be deduced from Equation (2.4), that $\theta_i(\cdot)$ must be zero, as paths with $\left\|\mathbf{p}'_{d,i}(\zeta_i)\right\| = 0$ exhibit cusps that are highly undesirable. Hence, in order to accommodate for this class of platforms, the requirement on the timing law has to be relaxed.

To overcome these three drawbacks, the timing law $\theta_i(\cdot)$ will be defined differently. Let the timing

law $\theta_i(\cdot)$ be a smooth *nonnegative* polynomial function of the parameter t_d, that is

$$\theta_i(t_d) = \frac{\mathrm{d}\zeta_i}{\mathrm{d}t_d}\,, \qquad t_d \in [0, t_{d,i}^{\mathrm{f}}]\,. \tag{2.7}$$

With the timing law $\theta_i(t_d)$ defined as in Equation (2.7), the map $\zeta_i(t_d)$ is given by the integral

$$\zeta_i(t_d) = \int_0^{t_d} \theta_i(\tau)\mathrm{d}\tau\,, \tag{2.8}$$

where, from the definition of ζ_i, the functions $\zeta_i(t_d)$ must satisfy $\zeta_i(0) = 0$ and $\zeta_i(t_{d,i}^{\mathrm{f}}) = 1$. If the timing laws $\theta_i(t_d)$ are polynomials of degree n_θ, then integrating Equation (2.8) will yield functions $\zeta_i(t_d)$ that are polynomials of degree $(n_\theta + 1)$. It is clear, that in this case we are not limited to $n_\theta = 2$, in order to express the functions $\zeta_i(t_d)$ analytically and, hence, can choose n_θ according to the set of constraints that needs to be satisfied. Moreover, the temporal separation constraint given above reduces to a polynomial since the compositions $(\mathbf{p}_{d,i} \circ \zeta_i)(t_d)$ and $(\mathbf{p}_{d,j} \circ \zeta_j)(t_d)$ are polynomials of the same degree. Nevertheless, the temporal separation constraint results in a high-degree polynomial and determining its roots is not straightforward. We shall see that using Bézier curves to describe the spatial paths and the timing laws significantly simplifies constructing the composition of functions and reduces the computational load while evaluating the temporal separation constraint. A similar observation can be made for the temporal constraints, which are equations in two variables, namely in t_d (through the timing law $\theta_i(t_d)$) and ζ_i. We can convert the temporal constraints to (high-degree) polynomials of one single variable t_d by substituting Equation (2.8).

Considering the favorable properties of describing the timing law by Equation (2.7) instead of Equation (2.5), the second approach will be adopted in the subsequent development of the three-dimensional trajectory-generation framework, in contrast to the work described in [18, 20]. Nonetheless, the results reported in [18, 20] are valid and as such the planar trajectory-generation framework presented therein serves as a feasibility study and baseline for the spatial framework described hereafter.

2.3 Boundary Conditions and Constraints

In the problem of trajectory generation for autonomous vehicles, we typically deal with missions for which the initial and final conditions on the trajectory, i.e. the boundary conditions, are prespecified. Therefore, in the following development of the trajectory-generation framework, it is assumed that the positions, speeds

and course angles of each individual aircraft at the endpoints are prescribed, that is

$$\begin{aligned} &\mathbf{p}_{d,i}(\zeta_i = 0) = \mathbf{p}_i^{\mathrm{i}}\,, \quad \gamma_i(\zeta_i = 0) = \gamma_i^{\mathrm{i}}\,, \quad \psi_i(\zeta_i = 0) = \psi_i^{\mathrm{i}}\,, \qquad v_{d,i}(t_{d,i} = 0) = v_i^{\mathrm{i}}\,, \\ &\mathbf{p}_{d,i}(\zeta_i = 1) = \mathbf{p}_i^{\mathrm{f}}\,, \quad \gamma_i(\zeta_i = 1) = \gamma_i^{\mathrm{f}}\,, \quad \psi_i(\zeta_i = 1) = \psi_i^{\mathrm{f}}\,, \quad v_{d,i}(t_{d,i} = t_{d,i}^{\mathrm{f}}) = v_i^{\mathrm{f}}\,, \end{aligned} \tag{2.9}$$

for $i = 1, \ldots, N$. Here, $\mathbf{p}_i^{\mathrm{i}}$, γ_i^{i}, ψ_i^{i}, and v_i^{i} are the initial position, flight-path angle, course, and speed, respectively, while $\mathbf{p}_i^{\mathrm{f}}$, γ_i^{f}, ψ_i^{f}, and v_i^{f} are the specified quantities at the final endpoint of the trajectory. Equation (2.9) gives the boundary conditions that need to be satisfied by the generated trajectory for the ith vehicle. In fact, constructing a smooth curve with given endpoints and derivatives, such as the boundary conditions given above, is defined as the first-order Hermite interpolation problem [32].

Additionally, a set of generated trajectories has to satisfy mission-specific constraints, for example simultaneous arrival at pre-defined destinations or specified lengths of each individual path. When simultaneous time-of-arrival of the vehicles is required, and the absolute time-of-arrival is not specified, then an additional constraint is

$$t_{d,i}^{\mathrm{f}} - t_{d,j}^{\mathrm{f}} = 0 \qquad \text{for} \quad i \neq j, \quad i, j = 1, 2, \ldots, N. \tag{2.10}$$

Another mission-specific requirement could be ensuring continuity of the turn-rate $\dot{\psi}$ at the final point of the trajectory, if the vehicle has to smoothly transition from the path onto a circular orbit.

As mentioned before, our objective is not solely to satisfy the mission requirements, but, equally importantly, to generate trajectories that the vehicles are able to follow and, at the same time, are collision-free. Therefore, the trajectories have to satisfy additional constraints, namely a set of dynamic constraints, and also requirements regarding the minimum spatial clearance between the paths or the vehicles.

2.3.1 Flyable trajectories: dynamic constraints of the vehicles

Flyable trajectories comply with the dynamic constraints of the vehicles and, therefore, can be closely followed if the vehicles executing the mission are equipped with autopilots that enable accurate tracking of the control commands. Trajectories that are not flyable will inevitably result in path-following errors and may jeopardize the successful completion of the mission or, in the worst case, lead to a loss of a vehicle. In the case of fixed-wing aircraft, violating the stall speed $v_{\min}$ is a good example of the latter.

In [18, 20, 86, 98] the rotational dynamic constraints are given in terms of intrinsic geometrical properties. For planar curves, the *curvature* κ_c is bounded so as to meet the maximum turn-rate $\dot{\psi}_{\max}$ of the vehicle. This works well for planar trajectories, since there is a direct relation between the maximum turn-rate $\dot{\psi}_{\max}$

and the maximum allowable curvature $\kappa_{c,\max}$:

$$\kappa_{c,\max} = \frac{1}{R_{\min}} = \frac{\dot{\psi}_{\max}}{v_{\max}} ,$$

where $R_{\min}$ is the minimum turn radius, and $v_{\max}$ is the maximum speed of the vehicle. An observation that one can make is that bounding the curvature is in fact a conservative approach, since a turn at $\dot{\psi}_{\max}$ with a turn radius smaller than $R_{\min}$ is perfectly fine, if the speed $v < v_{\max}$. Nevertheless, the results in [18,20] for planar trajectories justify meeting the dynamic constraints by means of bounding the maximum curvature of the path.

The authors of [86,98] use a similar notion to extend their framework to spatial trajectories, by introducing and bounding the *torsion* τ_c, additional to the bound on the curvature. Similar arguments hold regarding the conservatism, however, for spatial curves there is an added complexity to this approach. It is not straightforward to translate dynamic bounds on the yaw, pitch, and roll dynamics of the vehicle into limits on the curvature κ_c and torsion τ_c. For example, the yaw motion is no longer associated only with the curvature, but is a coupling of both the curvature and torsion.

The major drawback of using geometric properties to satisfy dynamic constraints is the fact that bounding the torsion τ_c excludes, for example, straight lines and curves exhibiting inflection points. This can be seen as follows. Consider the definitions of both the curvature and the torsion of a curve $\mathbf{r}(\zeta)$:

$$\kappa_c(\zeta) = \frac{\|\mathbf{r}'(\zeta) \times \mathbf{r}''(\zeta)\|}{\|\mathbf{r}'(\zeta)\|^3} , \qquad \tau_c(\zeta) = \frac{\langle\, \mathbf{r}'(\zeta) \times \mathbf{r}''(\zeta)\,, \mathbf{r}'''(\zeta)\,\rangle}{\|\mathbf{r}'(\zeta) \times \mathbf{r}''(\zeta)\|^2} ,$$

where $\mathbf{r}'(\zeta)$, $\mathbf{r}''(\zeta)$, and $\mathbf{r}'''(\zeta)$ are the first, second, and third derivative of $\mathbf{r}(\zeta)$ with respect to ζ, respectively. For straight paths (or segments) and curves with inflection points, the curvature κ_c is (momentarily) zero and, hence, the torsion becomes unbounded. Therefore, by bounding the torsion this family of (simple) curves is not admissible as a feasible solution.

In our framework, we proceed from the dynamics of the vehicles, and derive bounds that are directly related to the dynamics and do not suffer from the shortcomings of a purely geometric approach (see Section 1.2 for details). First, let an inertial reference frame $\{\mathcal{I}\}$ be characterized by the orthonormal vectors $\{\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}}\}$ (see Figure 2.1). The unit vectors $\hat{\mathbf{i}}$ and $\hat{\mathbf{j}}$ lie in the horizontal plane, while the unit vector $\hat{\mathbf{k}}$ points up vertically in the opposite direction of the gravity vector. Then, the (translational) equations of a

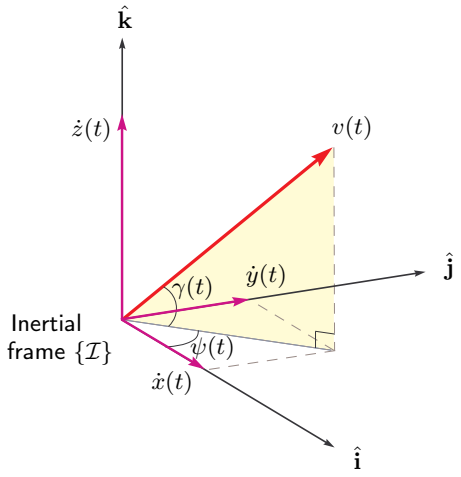


Figure 2.1: Definition of the flight-path angle $\gamma(t)$ and the course angle $\psi(t)$.

vehicle, governing the dynamics in this inertial reference frame $\{\mathcal{I}\}$, are given by:

$$\begin{aligned} \dot{x} &= v\cos\gamma\cos\psi\,, \\ \dot{y} &= v\cos\gamma\sin\psi\,, \\ \dot{z} &= v\sin\gamma\,, \end{aligned}$$

subject to

$$\begin{aligned} &v_{\min} \leq v(t) \leq v_{\max}\,, \quad |a(t)| \leq a_{\max}\,, \\ &\gamma_{\min} \leq \gamma(t) \leq \gamma_{\max}\,, \quad |\dot{\gamma}(t)| \leq \dot{\gamma}_{\max}\,, \quad |\dot{\psi}(t)| \leq \dot{\psi}_{\max}\,, \qquad \forall t \geq 0\,, \end{aligned} \tag{2.11}$$

where $[x(t), y(t), z(t)]^\top$, $v(t)$, and $a(t)$ are the coordinates, speed, and acceleration, respectively, of the UAV given in the inertial reference frame $\{\mathcal{I}\}$, $\gamma(t)$ denotes the flight-path angle, and $\psi(t)$ represents the course of the vehicle (see Figure 2.1). The bounds in Equation (2.11) are vehicle-specific and, based on the differential flatness property [68], can be derived from their dynamics. The conditions in (2.11) prescribe the physical limitations of the vehicle and, therefore, the generated desired trajectories shall not demand maneuvering along a path that exceeds these dynamic constraints. Hence, we need to derive expressions for the desired speed and acceleration profiles as well as for the flight-path angle, rate of change of the flight-path angle, and the turn rate along the generated trajectory of the ith vehicle, in terms of the spatial paths $\mathbf{p}_{d,i}(\zeta_i)$ and the timing laws $\theta_i(t_d)$. First, let $\mathbf{p}'_{d,i}(\zeta_i)$ and $\mathbf{p}''_{d,i}(\zeta_i)$ denote the first and second derivatives of $\mathbf{p}_{d,i}(\zeta_i)$ with respect to ζ_i, respectively. The flight-path angle γ_i is a pure *geometric* property of the spatial path and,

hence, it is natural to have it parameterized by ζ_i:

$$\gamma_i(\zeta_i) = \arcsin\left(\frac{\langle\, \mathbf{p}'_{d,i}(\zeta_i), \hat{\mathbf{k}}\,\rangle}{\left\|\mathbf{p}'_{d,i}(\zeta_i)\right\|}\right). \tag{2.12}$$

The other kinematic properties are of *temporal* nature and, therefore, we characterize them by the time variable t_d. The corresponding expressions are

$$\begin{aligned} v_{d,i}(t_d) &= \sigma_i(\zeta_i(t_d))\,\theta_i(t_d)\,, \\ \dot{\gamma}_i(t_d) &= \frac{\sigma_i(\zeta_i(t_d))\,\langle\, \mathbf{p}''_{d,i}(\zeta_i(t_d)), \hat{\mathbf{k}}\,\rangle - \sigma'_i(\zeta_i(t_d))\langle\, \mathbf{p}'_{d,i}(\zeta_i(t_d)), \hat{\mathbf{k}}\,\rangle}{\sigma_i(\zeta_i(t_d))\,\left(\langle\, \mathbf{p}'_{d,i}(\zeta_i(t_d)), \hat{\mathbf{i}}\,\rangle^2 + \langle\, \mathbf{p}'_{d,i}(\zeta_i(t_d)), \hat{\mathbf{j}}\,\rangle^2\right)^{\frac{1}{2}}}\,\theta_i(t_d)\,, \\ \dot{\psi}_i(t_d) &= \frac{\langle\, \mathbf{p}'_{d,i}(\zeta_i(t_d)) \times \mathbf{p}''_{d,i}(\zeta_i(t_d)), \hat{\mathbf{k}}\,\rangle}{\langle\, \mathbf{p}'_{d,i}(\zeta_i(t_d)), \hat{\mathbf{i}}\,\rangle^2 + \langle\, \mathbf{p}'_{d,i}(\zeta_i(t_d)), \hat{\mathbf{j}}\,\rangle^2}\,\theta_i(t_d)\,, \end{aligned} \tag{2.13}$$

where

$$\sigma'_i(\zeta_i) = \frac{\mathrm{d}}{\mathrm{d}\zeta_i}\left\|\mathbf{p}'_{d,i}(\zeta_i)\right\| = \frac{\langle\, \mathbf{p}'_{d,i}(\zeta_i), \mathbf{p}''_{d,i}(\zeta_i)\,\rangle}{\left\|\mathbf{p}'_{d,i}(\zeta_i)\right\|}.$$

The time derivatives of $\gamma_i(t_d)$ and $\psi_i(t_d)$ in Equation (2.13) are taken with respect to t_d. Also notice that the functions in Equation (2.13) are written as functions of a single parameter t_d by substituting Equation (2.8) for the variable ζ_i. Depending on the vehicle platform for which the trajectories are generated, the *acceleration along the path* or the *magnitude of the total acceleration* can be considered, given by

$$a^{\mathrm{p}}_{d,i}(t_d) = \frac{\mathrm{d}}{\mathrm{d}t_d} v_{d,i}(t_d) = \sigma_i(\zeta_i(t_d))\dot{\theta}_i(t_d) + \sigma'_i(\zeta_i(t_d))\theta_i^2(t_d)\,, \tag{2.14a}$$

$$a^{\mathrm{t}}_{d,i}(t_d) = \left\|\frac{\mathrm{d}^2\mathbf{p}_{d,i}(\zeta_i(t_d))}{\mathrm{d}t_d^2}\right\| = \left\|\mathbf{p}'_{d,i}(\zeta_i(t_d))\dot{\theta}_i(t_d) + \mathbf{p}''_{d,i}(\zeta_i(t_d))\theta_i^2(t_d)\right\|, \tag{2.14b}$$

respectively. Similarly, the time derivative of $\theta_i(t_d)$ in Equation (2.14) is taken with respect to t_d. For example, if fixed-wing aircraft are considered, then the acceleration along the path (Equation (2.14a)) will be subject to the maximum available excess thrust. However, in the case of multirotors, for which the thrust vector can be (nearly) freely directed in the inertial frame due to their inherent agility, it is the magnitude of the total acceleration (Equation (2.14b)) that is subject to the maximum excess thrust. With this setup, a trajectory is said to be *flyable* if the equations in (2.12), (2.13), and (2.14a) (or (2.14b)) meet similar bounds as in Equation (2.11), which implies that a vehicle is able to follow the trajectory without exceeding its dynamic constraints, such as the minimum (stall) and maximum speeds, and the maximum turn rate.

Hence, Equations (2.12), (2.13) and (2.14a) (or (2.14b)) have to satisfy

$$\begin{gathered} v_{d,\min} \leq v_{d,i}(t_d) \leq v_{d,\max}\,, \quad \gamma_{\min} \leq \gamma_i(\zeta_i) \leq \gamma_{\max} \\ |\dot{\gamma}_i(t_d)| \leq \dot{\gamma}_{\max}\,, \quad |\dot{\psi}_i(t_d)| \leq \dot{\psi}_{\max} \\ |a^{\mathrm{p}}_{d,i}(t_d)| \leq a_{d,\max} \quad \text{or} \quad |a^{\mathrm{t}}_{d,i}(t_d)| \leq a_{d,\max}\,, \end{gathered} \tag{2.15}$$

for all $(\zeta_i, t_d) \in [0,1] \times [0, t^{\mathrm{f}}_{d,i}]$. Note that, $0 < v_{\min} \leq v_{d,\min} < v_{d,\max} \leq v_{\max}$ and $a_{d,\max} \leq a_{\max}$ have to be satisfied for the trajectories $\mathbf{p}_{d,i}(\zeta_i(t_d))$ to be flyable. Needless to say that the expressions in Equations (2.12)-(2.14) can be very complex and result in high-degree polynomials and, because of this, evaluating Equation (2.15) is not trivial and may be computationally expensive.

Remark 1 *Note that, in general, $v_{d,i}(t_d)$ is the total commanded speed to the autopilot and, therefore, $v_{d,min} = v_{min}$, $v_{d,max} = v_{max}$, and $a_{d,max} = a_{max}$. However, for certain specific missions, such as time-coordinated missions [22, 39, 63, 104–106], or in particular circumstances, for example in the presence of uncooperative vehicles [20], the speeds of the vehicles are adjusted from $v_{d,i}(t_d)$ during the execution of the mission, as to achieve coordination, or to avert an imminent collision, respectively. In these cases, $v_{d,i}(t_d)$ is no longer the total commanded speed and, hence, the more restrictive bounds given in Equation (2.15) are adopted during the trajectory-generation phase, in order to prevent premature saturation of the total speed command during mission execution.*

2.3.2 Feasible trajectories: spatial and temporal separation

Trajectories of different vehicles have to be spatially deconflicted a priori, in order to avoid collision and ensure safe simultaneous operation in the airspace. Trajectories that are flyable and safe to fly are called *feasible* trajectories. Deconfliction between trajectories can be guaranteed through *spatial* separation or *temporal* separation, where the vehicles are separated in space or time, respectively. Spatial separation is guaranteed if the minimum distance between any two points on the paths of the ith and jth vehicle is greater than or equal to the minimum spatial clearance E_d, i.e.

$$\min_{\substack{i,j=1,\ldots,N \\ i \neq j}} \left\| \mathbf{p}_{d,i}(\zeta_i) - \mathbf{p}_{d,j}(\zeta_j) \right\|^2 \geq E_d^2\,, \qquad \forall \zeta_i, \zeta_j \in [0,1]. \tag{2.16}$$

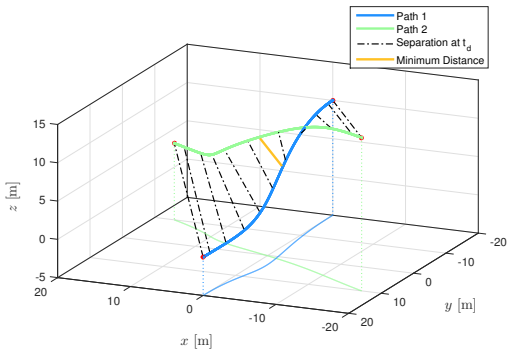
On the other hand, temporal separation is ensured if, for any time t_d, the minimum distance between the ith and jth vehicle is greater than or equal to the minimum spatial clearance E_d, that is

$$\min_{\substack{i,j=1,\dots,N \\ i\neq j}} \|\mathbf{p}_{d,i}(\zeta_i(t_d)) - \mathbf{p}_{d,j}(\zeta_j(t_d))\|^2 \geq E_d^2\,, \quad \forall t_d \in [0, t^{\mathrm{f}}_{d_{ij}}], \tag{2.17}$$

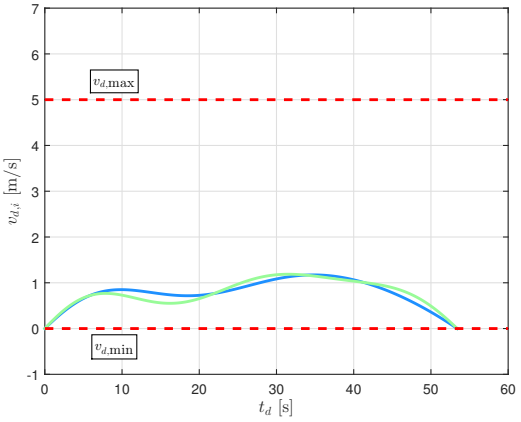
where $t^{\mathrm{f}}_{d_{ij}} = \min\{t^{\mathrm{f}}_{d,i}, t^{\mathrm{f}}_{d,j}\}$. In Equation (2.17) the paths are allowed to intersect, but the vehicles are time separated and, hence, do not arrive at the intersection point simultaneously. The generated trajectories are spatially deconflicted, and form a set of feasible trajectories if and only if they are flyable and either Equation (2.16) or (2.17) is satisfied. Note that, in Equations (2.16) and (2.17), E_d is larger than the actual required minimum distance E, so as to allow for path-following errors, or deviations from the desired paths necessary in order to prevent a potential collision with uncooperative vehicles.

Both deconfliction strategies are illustrated by the following two examples. Figure 2.2 shows two trajectories that are deconflicted through spatial separation. The yellow line in Figure 2.2a indicates the minimum distance between the two paths, and from Figure 2.2d it can be seen that this minimum distance is greater than or equal to the required minimum spatial clearance E_d. Note that by definition the temporal separation is always greater than or equal to the spatial separation and, therefore, the two trajectories are also sufficiently separated in time. That the converse does not hold, is shown in Figure 2.3. Despite the fact that the minimum distance between the spatial paths does not meet the required minimum spatial clearance E_d (see Figure 2.3d), the two trajectories are still considered collision-free, but in temporal sense (Figure 2.3c).

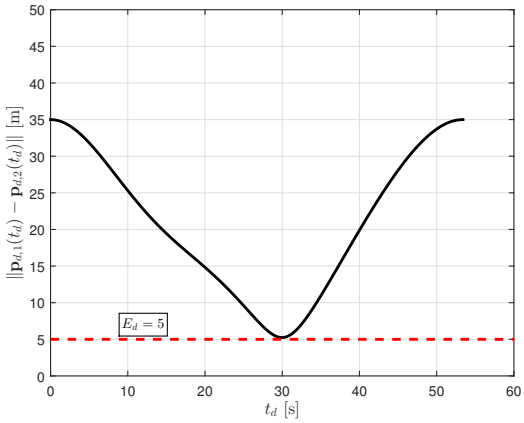
One of the factors that determine the minimum spatial clearance E_d, which can be ensured through spatial or temporal separation, is the error of an underlying path-following controller onboard the vehicles. In the case when temporal separation is preferred over spatial separation, the trajectories are allowed to intersect and, hence, a time-coordination controller is also necessary in order to keep the vehicles sufficiently separated at all times. However, safe operation is dependent on the quality and robustness of the communication network over which the vehicles exchange information, and is potentially compromised whenever this network is faulty or jammed. On the other hand, spatial separation guarantees collision-free trajectories at all times, even if the communication network is temporarily or permanently unavailable, but results in a more conservative set of trajectories that may require a larger (air)space. Which deconfliction strategy is used, depends on numerous factors and considerations, such as the type of mission, the environment, the quality of the communication network, and security requirements on data transmission during the mission.



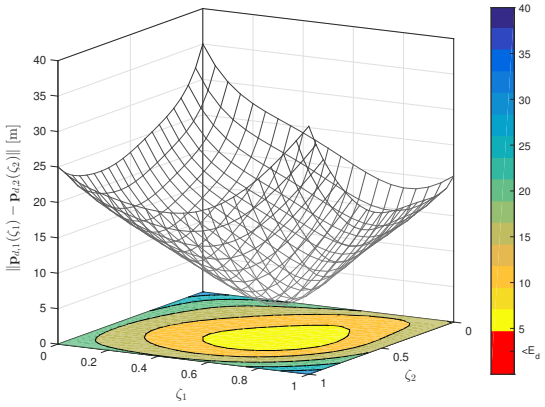
(a) Three-dimensional flight paths



(b) Speed profile $v_{d,i}(t_d)$

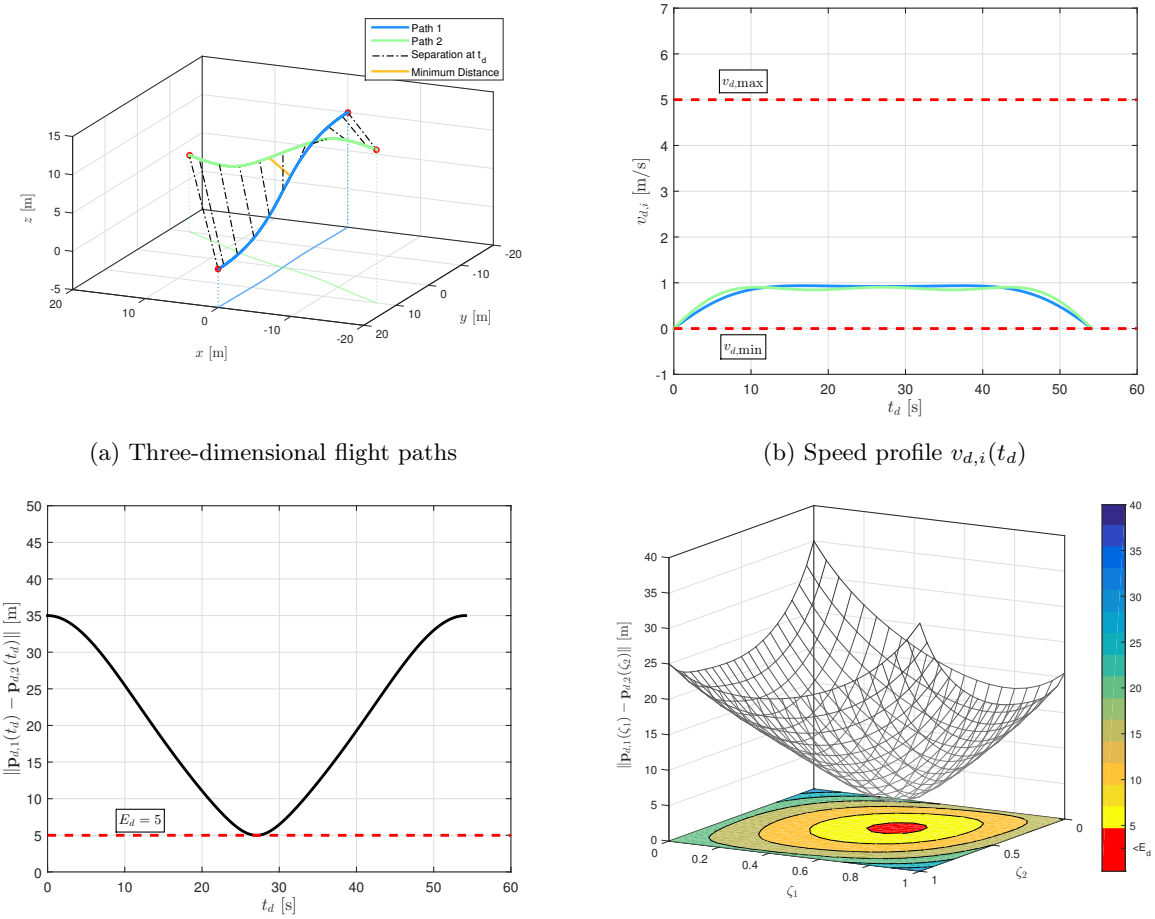


(c) Vehicle separation



(d) Path separation UAV 1 and 2

Figure 2.2: Example of deconfliction of two UAV trajectories through spatial separation.



(a) Three-dimensional flight paths

(b) Speed profile $v_{d,i}(t_d)$

(c) Vehicle separation

(d) Path separation UAV 1 and 2

Figure 2.3: Example of deconfliction of two UAV trajectories through temporal separation.

Chapter 3

Pythagorean Hodograph Bézier Curves

The cooperative trajectory-generation framework was formulated and set up mathematically in Chapter 2. The necessary operations involved includes differentiation, integration, and root finding of, generally, high-degree polynomials and rational functions of polynomials, which may be computationally expensive to perform. Additionally, multiplication and determining the composition of two polynomial functions may not be trivial and, more than often, the resulting high-degree polynomial exhibits undesirable behavior.

Hence, trajectory planning is not a trivial issue. In general, UAVs carry low power-consumption processors with limited memory in order to save weight to maximize the payload capacity. However, as shown in Chapter 2, the constraints that have to be satisfied are highly complex, while it is desirable that the trajectories are generated (near) real-time using these onboard processors. Therefore, the trajectory-planning algorithm has to be computationally efficient. One approach to relieve the computational load, is to seek a class of curves with geometric properties that can be exploited in satisfying the set of imposed constraints as described in Section 2.3. One such family of curves, possessing attractive geometric properties, is formed by the *Bézier curves*. A Bézier curve is completely determined by a set of *control points*. The curve starts at the first and ends at the last control point, while the curve lies completely in the *convex hull* containing the set of control points. This latter property is extremely useful when path planning for aircraft systems is considered, as the airspace in which an aircraft is allowed to maneuver is often allocated and limited by strict boundaries. An example of spatially deconflicted paths generated for a bounded airspace using Bézier curves can be found in [53]. Since Bézier curves are polynomials, the set of Bézier curves is closed under the operations of addition/subtraction, multiplication, and scaling, and is also closed under differentiation and integration. Computationally efficient algorithms are available to perform these operations on the control points.

Within the class of polynomial curves, there exists a sub-class of curves of which the components of the hodographs satisfy a Pythagorean quartuple of polynomials. These are called *Pythagorean Hodograph* (PH) curves. The hodograph of a curve is the locus described by the first parametric derivative of the curve [31,37]. PH curves have the attractive property that the speed is a polynomial function of its parameter and, therefore,

admit an exact measurement of the arc length by simply evaluating a polynomial. This of course reduces the computational load considerably compared to obtaining the arc lengths of the curves numerically.

In this chapter we introduce the *Pythagorean Hodograph Bézier curves* that will be used to mathematically describe the trajectories. These curves are Bézier curves of which the hodographs satisfy the Pythagorean polynomial quartuple. In this chapter, we give a treatment of these curves separately: first, we give an overview of the properties and construction of PH curves, followed by a similar discussion for Bézier curves. In the next chapter we will describe in more detail how PH Bézier curves can be constructed using a *quaternion representation*.

3.1 Pythagorean Hodograph Curves

The question that comes first to one's mind when dealing with curves, and especially with trajectories, is what is the exact arc length of the curve? It seems like an innocent question, however, from a differential geometer's point of view, the answer is not trivial at all. Let $s(\zeta)$ be the arc length of a differentiable spatial curve $\mathbf{r}(\zeta) = [x(\zeta), y(\zeta), z(\zeta)]^\top$ and given by:

$$s(\zeta) = \int_0^\zeta \|\mathbf{r}'(\tau)\| d\tau = \int_0^\zeta \sqrt{x'^2(\tau) + y'^2(\tau) + z'^2(\tau)}\, d\tau\,, \tag{3.1}$$

where $\mathbf{r}'(\zeta) = \mathrm{d}\mathbf{r}(\zeta)/\mathrm{d}\zeta = [x'(\zeta), y'(\zeta), z'(\zeta)]^\top$ denotes the hodograph of $\mathbf{r}(\zeta)$. The *parametric speed* is given by

$$\|\mathbf{r}'(\zeta)\| = \frac{\mathrm{d}}{\mathrm{d}\zeta} s(\zeta) = \sqrt{x'^2(\zeta) + y'^2(\zeta) + z'^2(\zeta)}\,.$$

Note that the parametric speed $\|\mathbf{r}'(\zeta)\|$ is *not* the desired speed profile $v_{d,i}(t_d)$ from Equation (2.13). In general, Equation (3.1) does not allow a solution in closed-form since the argument of the square root in the integrand is a polynomial function and, hence, the integral has to be approximated using numerical methods.

In the ideal case, an analytical expression for Equation (3.1) can be found if $\|\mathbf{r}'(\zeta)\| \equiv 1$, i.e. the "unit speed" parametrization and, therefore, the curve is conveniently parameterized by its arc length since $s(\zeta) = \zeta$. Unfortunately, as shown by Theorem 16.1 in [32], such a parametrization is not possible for any planar (and spatial) curve, other than for straight lines. However, even if a unit speed parametrization does not exist, there is still an elegant way to simplify Equation (3.1) significantly. For that, we need to incorporate *a priori* a Pythagorean structure in the hodograph of $\mathbf{r}(\zeta)$. To this end, let

$$x'^2(\zeta) + y'^2(\zeta) + z'^2(\zeta) = \sigma^2(\zeta)\,, \tag{3.2}$$

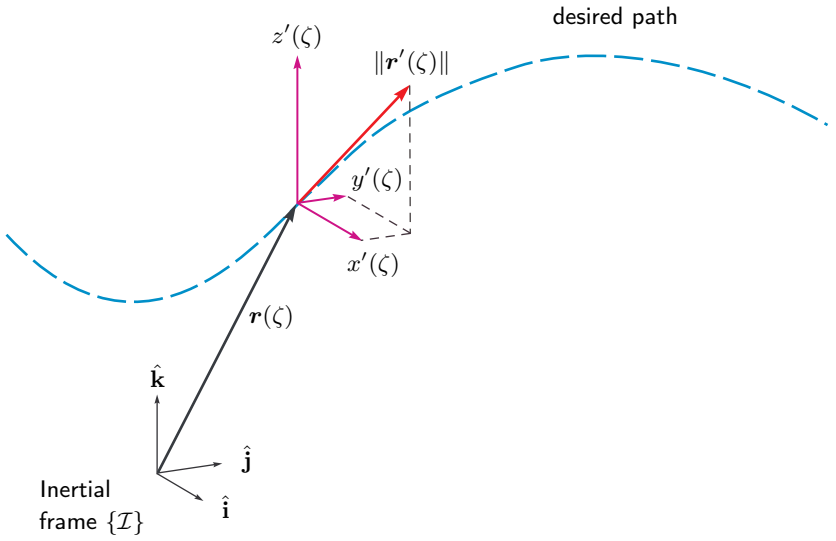


Figure 3.1: PH curves have the property that the parametric speed $\|\boldsymbol{r}'(\zeta)\| = \sigma(\zeta)$ is a polynomial function.

for some polynomial $\sigma(\zeta)$. In other words, we ensure that $x'^2(\zeta) + y'^2(\zeta) + z'^2(\zeta)$ is a perfect square of the polynomial $\sigma(\zeta)$ (see Figure 3.1). Curves with hodographs satisfying Equation (3.2) are called *Pythagorean Hodograph* curves [32]. Substituting Equation (3.2) into (3.1) results in

$$s(\zeta) = \int_0^\zeta \sigma(\tau) d\tau,$$

and the arc length $s(\zeta)$ can be found as a closed-form function through a simple integration of polynomial $\sigma(\zeta)$ of any degree. Theorem 21.1 in [32] states that the Pythagorean condition (3.2) is satisfied if and only if $x'(\zeta)$, $y'(\zeta)$, $z'(\zeta)$, and $\sigma(\zeta)$ can be expressed in terms of other real polynomials $u(\zeta)$, $v(\zeta)$, $p(\zeta)$ and $q(\zeta)$ in the form

$$x'(\zeta) = u^2(\zeta) + v^2(\zeta) - p^2(\zeta) - q^2(\zeta)\,, \tag{3.3a}$$

$$y'(\zeta) = 2\left[u(\zeta)q(\zeta) + v(\zeta)p(\zeta)\right]\,, \tag{3.3b}$$

$$z'(\zeta) = 2\left[v(\zeta)q(\zeta) - u(\zeta)p(\zeta)\right]\,, \tag{3.3c}$$

$$\sigma(\zeta) = u^2(\zeta) + v^2(\zeta) + p^2(\zeta) + q^2(\zeta)\,. \tag{3.3d}$$

Although Equation (3.3) is a sufficient and necessary condition, constructing PH spatial curves using Equations (3.3a)-(3.3d) in this form results in a system of nine coupled quadratic equations for twelve real

unknowns. A more compact system is described in [32, 34–36], based on the quaternion representation of Equation (3.3), that was first introduced in [19].

A quaternion $\mathcal{A} = a + a_x\hat{\mathbf{i}} + a_y\hat{\mathbf{j}} + a_z\hat{\mathbf{k}}$ comprises of a scalar part a and a vector part $\mathbf{a} = a_x\hat{\mathbf{i}} + a_y\hat{\mathbf{j}} + a_z\hat{\mathbf{k}}$, and its conjugate $\mathcal{A}^*$ is defined as $\mathcal{A}^* = a - a_x\hat{\mathbf{i}} - a_y\hat{\mathbf{j}} - a_z\hat{\mathbf{k}}$. We can also write the quaternion $\mathcal{A}$ as $\mathcal{A} = |\mathcal{A}|\,\mathcal{U}$, where $|\mathcal{A}|^2 = \mathcal{A}\mathcal{A}^* = a^2 + |\mathbf{a}|^2$ is the square of the magnitude of the quaternion $\mathcal{A}$, and $\mathcal{U}$ is the *unit* quaternion with $|\mathcal{U}| = 1$, defined as $\mathcal{U} = \cos\frac{1}{2}\delta + \sin\frac{1}{2}\delta\,\mathbf{n}$ for some angle δ and unit vector $\mathbf{n}$. Then, given a pure vector quaternion $\mathbf{v}$ and a unit quaternion $\mathcal{U}$, the quaternion product $\mathcal{U}\,\mathbf{v}\,\mathcal{U}^*$ represents a rotation of $\mathbf{v}$ through an angle δ about the unit vector $\mathbf{n}$, and results in a pure vector quaternion.

The first-order Hermite interpolation problem, concerned with the construction of a smooth curve matching given endpoints and derivatives, often yields equations of the following form when formulated in the quaternion representation:

$$\mathcal{A}\,\hat{\mathbf{i}}\,\mathcal{A}^* = \mathbf{c}\,, \tag{3.4}$$

with $\mathbf{c} = c_x\hat{\mathbf{i}} + c_y\hat{\mathbf{j}} + c_z\hat{\mathbf{k}}$. The *one-parameter family of general solutions* to Equation (3.4) is given by:

$$\mathcal{A}(\phi) = \sqrt{\frac{1}{2}(1+\lambda)|\mathbf{c}|}\left(-\sin\phi + \cos\phi\,\hat{\mathbf{i}} + \frac{\mu\cos\phi + \nu\sin\phi}{1+\lambda}\hat{\mathbf{j}} + \frac{\nu\cos\phi - \mu\sin\phi}{1+\lambda}\hat{\mathbf{k}}\right), \tag{3.5}$$

where (λ, μ, ν) and $|\mathbf{c}| \in \mathbb{R}^+$ are, respectively, the direction-cosines and the magnitude of the vector $\mathbf{c}$. Next, consider a quaternion *polynomial*

$$\mathcal{A}(\zeta) = u(\zeta) + v(\zeta)\hat{\mathbf{i}} + p(\zeta)\hat{\mathbf{j}} + q(\zeta)\hat{\mathbf{k}}\,. \tag{3.6}$$

Then the following product $\mathcal{A}(\zeta)\,\hat{\mathbf{i}}\,\mathcal{A}^*(\zeta)$ results in:

$$\begin{aligned}\mathcal{A}(\zeta)\,\hat{\mathbf{i}}\,\mathcal{A}^*(\zeta) = {}& \left[u^2(\zeta) + v^2(\zeta) - p^2(\zeta) - q^2(\zeta)\right]\hat{\mathbf{i}} \\ & + 2\left[u(\zeta)q(\zeta) + v(\zeta)p(\zeta)\right]\hat{\mathbf{j}} + 2\left[v(\zeta)q(\zeta) - u(\zeta)p(\zeta)\right]\hat{\mathbf{k}}\,.\end{aligned} \tag{3.7}$$

Comparing this result with Equation (3.3), we observe that the hodograph $\mathbf{r}'(\zeta) = \mathcal{A}(\zeta)\,\hat{\mathbf{i}}\,\mathcal{A}^*(\zeta)$ satisfies the conditions to be the hodograph of a PH spatial curve $\mathbf{r}(\zeta)$. Rewriting the spatial hodograph $\mathbf{r}'(\zeta)$ in terms of the unit quaternion $\mathcal{U}(\zeta)$ yields $\mathbf{r}'(\zeta) = |\mathcal{A}(\zeta)|^2\,\mathcal{U}(\zeta)\,\hat{\mathbf{i}}\,\mathcal{U}^*(\zeta)$. Hence, we can interpret Equation (3.7) geometrically as generating a spatial hodograph through a continuous family of spatial rotations and scalings of the basis vector $\hat{\mathbf{i}}$.

With the above formulation of spatial PH curves in terms of quaternions, and using Bézier curves to describe the polynomials $u(\zeta)$, $v(\zeta)$, $p(\zeta)$, and $q(\zeta)$, we shall see in the next chapter that the construction of

Pythagorean Hodograph Bézier curves simplifies considerably both in number of equations and unknowns. PH Bézier curves also provide a closed-form solution for the energy integral, and exact representations of offset curves at a distance d from the curve for all ζ as rational Bézier curves. The former property allows the designer to obtain the optimal shape of the curves by minimizing the energy, while the latter results in offset curves, that are rational Bézier curves themselves and, hence, inherit the nice geometric properties of the family of Bézier curves.

The efforts in finding an exact measurement of the arc length is not only to satisfy the differential geometer's curiosity. In the context of cooperative trajectory generation for multiple vehicles, more than often, simultaneous time-of-arrival of the vehicles is required. This is practically only feasible if the intersection of the *arrival-time windows* of the vehicles is non-empty, where the arrival-time window for the ith vehicle is defined as:

$$\delta t^{\mathrm{f}}_{d,i} := \left[t^{\mathrm{f}}_{d_{\min},i}, t^{\mathrm{f}}_{d_{\max},i} \right] ,$$

where

$$t^{\mathrm{f}}_{d_{\min},i} := \frac{s_i}{v_{\max}} , \qquad t^{\mathrm{f}}_{d_{\max},i} := \frac{s_i}{v_{\min}} ,$$

and s_i is the arc length of the spatial path. Clearly, in order to determine the arrival-time windows, one needs to compute the exact arc length of the curve. In [104] the *arrival margin* is derived from arrival-time windows, and it is argued that this arrival margin plays a key role in providing robustness to the mission operation at the coordination level.

3.2 Bézier Curves

Next, an overview is given of Bézier curves that are used to mathematically describe the trajectories, i.e. the spatial paths $\mathbf{p}_{d,i}(\zeta_i)$ and the timing laws $\theta_i(t_d)$. Bézier curves are polynomials over the finite interval $[0, 1]$ expressed in the *Bernstein* basis, consisting of Bernstein polynomials:

$$b^n_k(\zeta) = \binom{n}{k} (1-\zeta)^{n-k} \zeta^k , \qquad \zeta \in [0, 1] , \tag{3.8}$$

where n is the degree of the polynomial. The Bernstein polynomials are named after the Russian mathematician S.N. Bernstein, who published a constructive proof of the *Weierstrass Theorem* and where this polynomial basis was first introduced. Although the Bernstein polynomial approximation $P_n(t) = \sum_{k=0}^{n} f\left(\frac{k}{n}\right) b^n_k(t)$ converges to a continuous function $f(t)$ for $n \to \infty$, this happens at such a slow rate, that many deemed the Bernstein basis being of no practical use. In fact, P.J. Davis made the following comment in his book [26]:

> "The fact seems to have precluded any numerical application of Bernstein polynomials from having been made. Perhaps they will find application when the properties of the approximant in the large are of more importance than the closeness of the approximation."

And true enough, with the widespread proliferation of computers in the industry, there was a growing need for a means of intuitively defining and modifying curves and surfaces. This was precisely where the Bernstein polynomials found their niche. Besides inheriting the simplicity of polynomials, Bézier curves have the extra advantage of providing "shape handles" [32] in the form of the control points, that allow the designer to shape the curve or surface intuitively to his or her liking. As such, the Bézier curves have seen their applications in especially computer aided design, but also in the world of computer games and animations.

In this thesis, we explore the suitability and feasibility of using Bézier curves, and in particular PH Bézier curves, to describe the flight trajectories mathematically. Expressing the desired flight paths in terms of Bézier polynomials is attractive, due to numerous favorable geometric properties that these curves exhibit. A degree n Bézier curve is given by:

$$\mathbf{r}(\zeta) = \sum_{k=0}^{n} \bar{\mathbf{r}}_k \, b_k^n(\zeta) \,, \qquad \zeta \in [0,1] \,, \tag{3.9}$$

where ζ is a dimensionless parameter, the points $\bar{\mathbf{r}}_0, \ldots, \bar{\mathbf{r}}_n \in \mathbb{R}^m$ are the *control points* of the Bézier curve, and $b_k^n(\zeta)$ are the Bernstein polynomials given in (3.8). It is clear, that for spatial paths the control points $\bar{\mathbf{r}}_k$ are in $\mathbb{R}^3$. The set of Bézier polynomials is closed under the arithmetic operations of addition, subtraction, and multiplication, and under differentiation, integration, and composition. Besides being the coefficients that completely determine the Bézier polynomial, the control points are also key elements that define the attractive geometric properties of Bézier curves. One important property that is exploited extensively is the fact that a Bézier curve lies completely in the convex hull of these control points $\bar{\mathbf{r}}_0, \ldots, \bar{\mathbf{r}}_n$. Hence, the convex hull is a polyhedron (or a polygon for planar curves) with a subset of the control points as its vertices.

Figure 3.2 shows an example of a *quintic* planar Bézier curve. The curve here lies completely in the green dash-dotted polygon. Additionally, a Bézier curve starts at the first control point $\bar{\mathbf{r}}_0$ and ends at the last control point $\bar{\mathbf{r}}_n$, and it lies tangent to the *control polygon* (not to be confused with the *convex hull*) at these two end control points. Many operations performed on Bézier curves are computationally efficient, such as arithmetic operations, differentiation and integration, which in general reduce to recursive algorithms in terms of the control points. We list a few important properties of Bézier curves that will be extensively used in our trajectory-generation framework. A more comprehensive list is given in [27, 32, 33]. Let $\mathbf{r}(\zeta)$ be a

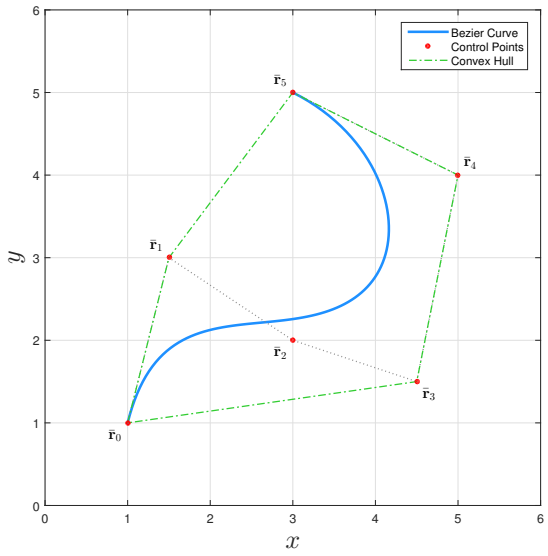


Figure 3.2: Example of a planar quintic Bézier curve. The Bézier curve is shown by a blue-solid line. The control points are indicated by red-solid dots. The control polygon is presented with gray-dotted lines, while the convex hull of the control points is shown by a green dash-dotted polygon. It can be observed that the Bézier curve lies completely in the convex hull of the control points.

degree n Bézier curve. Then the following properties hold:

1. **Derivatives.** The first parametric derivative $\mathbf{r}'(\zeta) = \mathrm{d}\mathbf{r}(\zeta)/\mathrm{d}\zeta$, is a degree $(n-1)$ Bézier curve and its control points $\bar{\mathbf{h}}_k$ are solely determined by the control points $\bar{\mathbf{r}}_k$ of the curve:

$$\mathbf{r}'(\zeta) = \mathbf{h}(\zeta) = \sum_{k=0}^{n-1} \bar{\mathbf{h}}_k \, b_k^{n-1}(\zeta)\,, \qquad \zeta \in [0,1]\,,$$

$$\bar{\mathbf{h}}_k = n(\bar{\mathbf{r}}_{k+1} - \bar{\mathbf{r}}_k) \qquad \text{for} \quad k = 0, \ldots, (n-1).$$

2. **Integrals.** Likewise, the indefinite integral of $\mathbf{r}(\zeta)$ can be found as:

$$\int \mathbf{r}(\zeta)\mathrm{d}\zeta = \sum_{k=1}^{n+1} \bar{\mathbf{R}}_k \, b_k^{n+1}(\zeta) + \mathrm{c}\,, \qquad \zeta \in [0,1]\,,$$

$$\bar{\mathbf{R}}_k = \frac{1}{n+1} \sum_{j=0}^{k-1} \bar{\mathbf{r}}_j \qquad \text{for} \quad k = 1, \ldots, (n+1).$$

   The first control point $\bar{\mathbf{R}}_0$ and the integration constant c can be determined from the boundary conditions at $\zeta = 0$ and $\zeta = 1$, respectively.

3. **Degree elevation.** The polynomial $\mathbf{r}(\zeta)$ of *true* degree n can be expressed as a Bézier polynomial of

degree $(n+t)$, for all $t > 0$:

$$\mathbf{r}(\zeta) = \sum_{k=0}^{n+t} \bar{\mathbf{r}}_k^{n+t}\, b_k^{n+t}(\zeta)\,, \qquad \zeta \in [0,1]\,,$$

$$\bar{\mathbf{r}}_k^{n+t} = \sum_{j=\max(0,k-t)}^{\min(n,k)} \frac{\binom{t}{k-j}\binom{n}{j}}{\binom{n+t}{k}} \bar{\mathbf{r}}_j \qquad \text{for} \quad k = 0,\ldots,(n+t)\,,$$

where $\bar{\mathbf{r}}_k^{n+t}$ are the control points of the elevated Bézier curve.

4. **Multiplication.** Let $\mathrm{s}(\zeta)$ be a degree m Bézier curve with control points $\bar{\mathrm{s}}_k \in \mathbb{R}$. Then, control points $\bar{\mathbf{c}}_k$ of the product of Bézier curves $\mathbf{r}(\zeta)$ and $\mathrm{s}(\zeta)$ are given by:

$$\bar{\mathbf{c}}_k = \sum_{j=\max(0,k-n)}^{\min(m,k)} \frac{\binom{m}{j}\binom{n}{k-j}}{\binom{m+n}{k}} \bar{\mathbf{r}}_{k-j}\bar{\mathrm{s}}_j \qquad \text{for} \quad k = 0,\ldots,(m+n)\,.$$

5. **Composition.** Let $\mathrm{s}(\zeta)$ be a degree m Bézier curve with control points $\bar{\mathrm{s}}_k \in \mathbb{R}$. Then, the control points $\bar{\mathbf{c}}_j$ of the composition of two Bézier curves $(\mathbf{r}(\zeta) \circ \mathrm{s}(\zeta))$ are obtained via the values $\mathbf{a}_{i,j}^k$, where $\mathbf{a}_{i,0}^0 = \bar{\mathbf{r}}_i$ for $i = 0,\ldots,n$, and:

$$\bar{\mathbf{a}}_{i,j}^k = \frac{1}{\binom{km}{j}} \sum_{l=\max(0,j-m)}^{\min(j,km-m)} \binom{km-m}{l}\binom{m}{j-l} \left[(1-\bar{\mathrm{s}}_{j-l})\,\bar{\mathbf{a}}_{i,l}^{k-1} + \bar{\mathrm{s}}_{j-l}\bar{\mathbf{a}}_{i+1,l}^{k-1}\right]$$

is used for $k = 1,\ldots,n$, $i = 0,\ldots,n-k$, and $j = 0,\ldots,km$. Then the control points $\bar{\mathbf{c}}_j$ are given by:

$$\bar{\mathbf{c}}_j = \bar{\mathbf{a}}_{0,j}^n\,, \qquad\qquad j = 0,\ldots,mn\,.$$

One computational procedure that is often associated with Bézier curves is the *de Casteljau* algorithm. For a degree n Bézier curve and given $\alpha \in (0,1)$, this recursive algorithm computes the point $\mathbf{r}(\alpha)$ on the curve that subdivides the curve into two subsegments for which the union is equivalent to the original curve, and provides the control points of each of the two (sub) Bézier curves, which are also of degree n and parameterized over $[0,1]$. Moreover, the control points of the two newly formed Bézier curves converge toward the curve as the subdivision procedure is repeated.

The *de Casteljau* algorithm is fundamental to many computational procedures applied to Bézier curves, such as determining the minimum distance between two Bézier curves [14, 15]. The authors in [14] propose an iterative algorithm to compute the minimum distance between two Bézier curves, where the convex hull property and the *de Casteljau* algorithm are cleverly and extensively exploited in combination with the

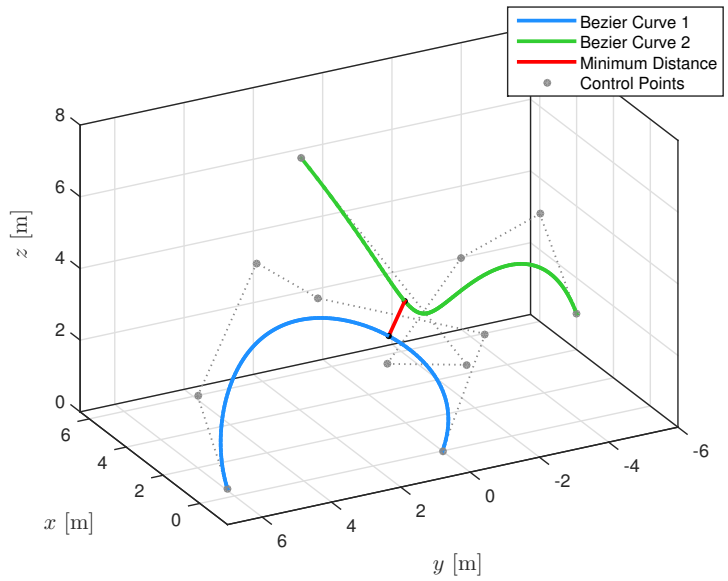


Figure 3.3: Two spatial Bézier curves where the minimum distance d is computed in < 40 msec.

Gilbert-Johnson-Keerthi (GJK) distance algorithm [29, 40, 52, 99]. The latter is an algorithm that is widely used in the computer graphics and animation world, to compute the minimum distance between two *convex shapes*. This GJK algorithm works flawlessly for Bézier curves, since these are always contained in their *convex hulls*. The *de Casteljau* algorithm continuously subdivides the curves, as to eventually converge to the points of minimum distance between the curves. The complete algorithm works extremely efficiently and computes the minimum distance between the two Bézier curves to within the desired tolerance and, therefore, is not an approximation method. For example, the minimum distance between two spatial Bézier curves, shown in Figure 3.3, is found in less than 40 msec using an implementation of the algorithm in Matlab®.

In the course of the work, the minimum-distance algorithm is modified to efficiently compute the global minima and maxima of scalar Bézier polynomials. Recall, that the constraints in Equation (2.15) require computing the global minima and maxima of the dynamic equations given in Equations (2.12) and (2.13). In Chapter 4, these expressions will be rewritten in terms of Bézier curves. Hence, the modified minimum distance algorithm allows us to efficiently compute the exact value of the global minima and maxima of these kinematic expressions. An example of a dynamic constraint, the speed profile $v_{d,i}(t_d)$, is shown in Figure 3.4. The global minimum and maximum, including the corresponding time t_d, are found in less than 40 msec.

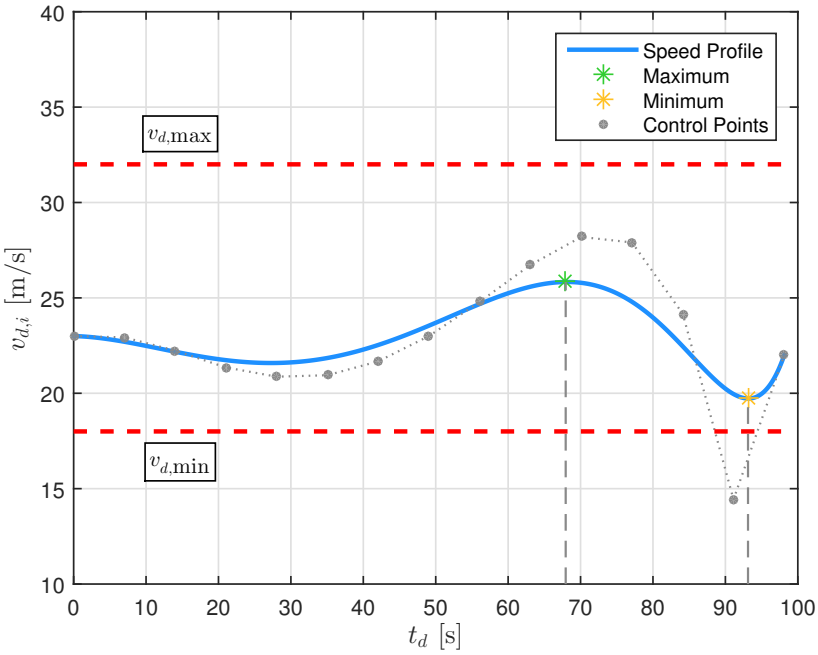


Figure 3.4: Typical speed profile $v_{d,i}(t_d)$. The maximum and minimum speed is computed in < 40 msec.

Lastly, besides (polynomial) Bézier curves that are described by Equation (3.9), there are also *rational* Bézier curves. A degree n rational Bézier curve is given by

$$\mathbf{r}(\zeta) = \frac{\sum_{k=0}^{n} w_k \bar{\mathbf{r}}_k \, b_k^n(\zeta)}{\sum_{k=0}^{n} w_k \, b_k^n(\zeta)}, \qquad \zeta \in [0,1]\,,$$

where $\bar{\mathbf{r}}_k \in \mathbb{R}^m$ are the control points, $b_k^n(\zeta)$ are the familiar Bernstein polynomials, and $w_k \in \mathbb{R}$ are the weights of the rational Bézier curve. These weights provide an extra degree of freedom for modifying the shape of the curve. In fact, rational Bézier curves are the only way to describe conic sections, other than the parabola. Many properties of (polynomial) Bézier curves carry over to rational Bézier curves. Likewise, computational procedures designed for Bézier curves can be extended for rational Bézier curves, such as the minimum distance and extremes finding algorithm. This is extremely helpful since, as we shall see in Chapter 4, many of the expressions in Equations (2.12) and (2.13) can be expressed as rational Bézier curves.

Remark 2 *The convex-hull property, on which practically many algorithms and properties hinge, also carries over to rational Bézier curves* if *all the weights w_k are positive. Hence, the weights have to be chosen carefully when possible; otherwise, this condition has to be verified and satisfied first, before proceeding with algorithms that rely on the convex hull property.*

Chapter 4

Cooperative Trajectory Generation using PH Bézier Curves

The framework for trajectory generation that is presented hereafter combines the favorable geometric properties of Bézier curves with the Pythagorean property of PH curves. The resulting PH Bézier curves retain their computational efficiency, while also allowing closed-form functions for the arc length $s_i(\zeta_i)$. Bézier curves were used in [53] to generate paths for multiple UAVs in a bounded airspace, while PH Bézier curves were generated for simultaneous arrival of multiple UAVs in [86]. In the latter work, the trajectories were generated such that the paths were of equal lengths, while the desired speeds were constant and equal for all UAVs. As discussed earlier, in our trajectory-generation framework, the arc lengths and the speed profiles are not restricted to such constraints and, hence, offer greater flexibility. Quintic PH Bézier curves are most commonly used to generate paths in several works on trajectory generation [85, 86, 94].

In our approach, we also seek to generate *quintic* PH Bézier curves to describe the spatial paths and use *quadratic* Bézier polynomials to represent the timing laws, that is

$$\mathbf{p}_{d,i}(\zeta_i) = \sum_{k=0}^{5} \bar{\mathbf{p}}_{i,k}\, b_k^5(\zeta_i)\,, \qquad \theta_i(t_d) = \sum_{k=0}^{2} \bar{\theta}_{i,k}\, b_k^2(t_d)\,,$$

for $i = 1, \ldots, N$, $(\zeta_i, t_d) \in [0,1] \times [0, t_{d,i}^{\mathrm{f}}]$, and where $\bar{\mathbf{p}}_{i,k} \in \mathbb{R}^3$ and $\bar{\theta}_{i,k} \in \mathbb{R}$ are the control points of the spatial path $\mathbf{p}_{d,i}(\zeta_i)$ and the timing law $\theta_i(t_d)$, respectively. The choice for working with PH Bézier curves of degree 5 may seem restrictive, as it limits the class of trajectories that can be generated; however, it is based on several considerations. First, increasing the degree of the spatial PH Bézier curves offers more flexibility in satisfying the constraints, but can potentially result in an increased computational cost. Second, PH Bézier curves of higher degree may exhibit unknown undesirable behavior and characteristics, since they have not been extensively explored yet. On the contrary, the properties and behavior of the quintic PH Bézier curves are extensively studied in, for example, [31, 32, 37]. Third, quintic PH Bézier curves are the simplest PH Bézier curves that allow constructing smooth curves with given endpoints and derivatives [32], where the latter are boundary conditions that are common to any trajectory-generation problem. The motivation behind the choice of quadratic Bézier polynomials for the timing laws is elaborated in Section 4.2.

4.1 Quaternion Representation of Spatial PH Bézier Curves

As shown in [32], spatial quintic PH Bézier curves can be represented compactly by introducing a quadratic Bézier (quaternion) polynomial. In our formulation, we take advantage of this result, and represent the spatial paths $\mathbf{p}_{d,i}(\zeta_i)$ by their corresponding Bézier (quaternion) polynomials, which we denote by $\mathcal{A}_i(\zeta_i)$ with control points $\bar{\mathcal{A}}_{i,k}$ for $k = 0, 1, 2$. The control points of the spatial path $\mathbf{p}_{d,i}(\zeta_i)$, in the form

$$\bar{\mathbf{p}}_{i,k} = \bar{\mathrm{p}}_{x_{i,k}}\hat{\mathbf{i}} + \bar{\mathrm{p}}_{y_{i,k}}\hat{\mathbf{j}} + \bar{\mathrm{p}}_{z_{i,k}}\hat{\mathbf{k}}\,,$$

are related to the control points $\bar{\mathcal{A}}_{i,k}$ as follows [32]:

$$\begin{aligned}
\bar{\mathbf{p}}_{i,1} &= \bar{\mathbf{p}}_{i,0} + \frac{1}{5}\bar{\mathcal{A}}_{i,0}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}^*_{i,0}\,,\\
\bar{\mathbf{p}}_{i,2} &= \bar{\mathbf{p}}_{i,1} + \frac{1}{10}\left(\bar{\mathcal{A}}_{i,0}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}^*_{i,1} + \bar{\mathcal{A}}_{i,1}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}^*_{i,0}\right)\,,\\
\bar{\mathbf{p}}_{i,3} &= \bar{\mathbf{p}}_{i,2} + \frac{1}{30}\left(\bar{\mathcal{A}}_{i,0}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}^*_{i,2} + 4\bar{\mathcal{A}}_{i,1}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}^*_{i,1} + \bar{\mathcal{A}}_{i,2}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}^*_{i,0}\right)\,,\\
\bar{\mathbf{p}}_{i,4} &= \bar{\mathbf{p}}_{i,3} + \frac{1}{10}\left(\bar{\mathcal{A}}_{i,1}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}^*_{i,2} + \bar{\mathcal{A}}_{i,2}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}^*_{i,1}\right)\,,\\
\bar{\mathbf{p}}_{i,5} &= \bar{\mathbf{p}}_{i,4} + \frac{1}{5}\bar{\mathcal{A}}_{i,2}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}^*_{i,2}\,,
\end{aligned} \tag{4.1}$$

where $\bar{\mathbf{p}}_{i,0} = \mathbf{p}_i^{\mathrm{i}}$ and $\bar{\mathbf{p}}_{i,5} = \mathbf{p}_i^{\mathrm{f}}$. Moreover, as shown in Appendix A, the spatial path $\mathbf{p}_{d,i}(\zeta_i)$ belongs to a family of PH Bézier curves characterized by four free parameters: two parameters, denoted here as $|\mathbf{d}|_i^{\mathrm{i}}$ and $|\mathbf{d}|_i^{\mathrm{f}}$, corresponding each to the magnitude of a vector, and two angular parameters, denoted as $\phi_{i,0}$ and $\phi_{i,2}$. Appendix A provides further insights into these parameters. Here, we only note that, as will become clear later, these four parameters will be part of the vector of optimization parameters for the ith trajectory. The timing law, characterized in the next section, will provide additional degrees of freedom to meet the dynamic and mission-specific constraints as well as the minimum spatial clearance between the vehicles.

4.2 Timing Law as a Bézier Polynomial

In Section 2.2, we defined the timing law $\theta_i(t_d)$ as (see Equation (2.7)):

$$\theta_i(t_d) = \frac{\mathrm{d}\zeta_i}{\mathrm{d}t_d}\,, \qquad \text{for} \quad t_d \in [0, t_{d,i}^{\mathrm{f}}]\,.$$

To preserve the favorable properties of Bézier curves, it is intuitive to describe the timing law as a Bézier polynomial. An immediate observation is that the parameter t_d for vehicle i is defined on $[0, t_{d,i}^{\mathrm{f}}]$, whereas

the dimensionless variable parameterizing Bézier polynomials is defined on $[0, 1]$. Hence, we introduce the *dimensionless* time variable

$$\hat{t}_{d,i} := \frac{t_d}{t^{\mathrm{f}}_{d,i}}\,, \qquad \mathrm{d}\hat{t}_{d,i} = \frac{1}{t^{\mathrm{f}}_{d,i}}\mathrm{d}t_d\,, \tag{4.2}$$

and re-define the timing law as follows:

$$\theta_i(\hat{t}_{d,i}) = \frac{\mathrm{d}\zeta_i}{\mathrm{d}\hat{t}_{d,i}}\,, \qquad \hat{t}_{d,i} \in [0, 1]\,. \tag{4.3}$$

The timing law $\theta_i(\cdot)$ is subsequently given by Equation (4.3) instead of Equation (2.7). Note that the timing law still relates the variable ζ_i to the time variable t_d via the normalized time variable $\hat{t}_{d,i}$. The function $\zeta_i(\hat{t}_{d,i})$ is represented by the integral

$$\zeta_i(\hat{t}_{d,i}) = \int_0^{\hat{t}_{d,i}} \theta_i(\tau)\mathrm{d}\tau$$

and if we assume that $\theta_i(\hat{t}_{d,i})$ is a Bézier polynomial of degree n_θ, then $\zeta_i(\hat{t}_{d,i})$ will be a degree $(n_\theta + 1)$ Bézier polynomial. Using the integration property of Bézier polynomials [32, 33], the corresponding control points $\bar{\zeta}_{i,k}$ can be expressed in terms of the control points $\bar{\theta}_{i,k}$. The first control point $\bar{\zeta}_{i,0} = 0$ follows from the fact that $\zeta_i(0) = 0$, while

$$\bar{\zeta}_{i,k} = \frac{1}{n_\theta + 1}\sum_{l=0}^{k-1} \bar{\theta}_{i,l} \quad \text{for} \quad k = 1, \ldots, (n_\theta + 1)\,.$$

We note that $\zeta_i(1) = 1$ and, therefore, $\bar{\zeta}_{i,n_\theta+1} = 1$; with this, we can derive an equality constraint in terms of the control points $\bar{\theta}_{i,k}$ that has to be satisfied:

$$\frac{1}{n_\theta + 1}\sum_{k=0}^{n_\theta} \bar{\theta}_{i,k} = 1\,. \tag{4.4}$$

Next, we observe that the dynamic relation of the parameter ζ_i with respect to the time variable t_d is now given by

$$\frac{\mathrm{d}\zeta_i}{\mathrm{d}t_d} = \frac{1}{t^{\mathrm{f}}_{d,i}}\theta_i(\hat{t}_{d,i})\,.$$

Then, the desired speed profile becomes

$$v_{d,i}(\hat{t}_{d,i}) = \frac{1}{t^{\mathrm{f}}_{d,i}}\left\|\mathbf{p}'_{d,i}(\zeta_i(\hat{t}_{d,i}))\right\|\theta_i(\hat{t}_{d,i})\,, \qquad \hat{t}_{d,i} \in [0, 1]$$

and, by using the differentiation property of Bézier polynomials [32,33], the initial and final speed conditions can be written as:

$$v_i^{\mathrm{i}} = \frac{5}{t_{d,i}^{\mathrm{f}}} \left\| \bar{\mathbf{p}}_{i,1} - \bar{\mathbf{p}}_{i,0} \right\| \bar{\theta}_{i,0} \,, \qquad v_i^{\mathrm{f}} = \frac{5}{t_{d,i}^{\mathrm{f}}} \left\| \bar{\mathbf{p}}_{i,5} - \bar{\mathbf{p}}_{i,4} \right\| \bar{\theta}_{i,n_\theta} \,, \tag{4.5}$$

where $\bar{\theta}_{i,0}$ and $\bar{\theta}_{i,n_\theta}$ are the first and last control points, respectively, of the timing law Bézier polynomial. It is obvious that, given the three equalities in Equations (4.4) and (4.5), at least *three* free parameters (including $t_{d,i}^{\mathrm{f}}$) are required to solve this system of equations. Therefore, it is necessary that $n_\theta \geq 2$ such that there remains at least one free parameter for meeting the temporal specifications. As mentioned earlier, in this framework we decide to work with *quadratic* Bézier polynomials to describe the timing laws $\theta_i(\hat{t}_{d,i})$, that is

$$\theta_i(\hat{t}_{d,i}) = \sum_{k=0}^{2} \bar{\theta}_{i,k}\, b_k^2(\hat{t}_{d,i}) \,, \qquad \text{for} \quad i = 1, \ldots, N \,, \quad \hat{t}_{d,i} \in [0,1] \,. \tag{4.6}$$

Note that the timing laws are Bézier polynomials that do not have Pythagorean hodographs. With the timing law defined as in Equation (4.6), it is easy to verify that the function $\zeta_i(\hat{t}_{d,i})$ can be found as:

$$\begin{gathered} \zeta_i(\hat{t}_{d,i}) = \sum_{k=0}^{3} \bar{\zeta}_{i,k} b_k^3(\hat{t}_{d,i}) \,, \\ \bar{\zeta}_{i,0} = 0 \,, \quad \bar{\zeta}_{i,1} = \frac{1}{3}\bar{\theta}_{i,0} \,, \quad \bar{\zeta}_{i,2} = \frac{1}{3}\left(\bar{\theta}_{i,0} + \bar{\theta}_{i,1}\right) \,, \quad \bar{\zeta}_{i,3} = 1 \,. \end{gathered}$$

Also, with the timing laws defined as a quadratic Bézier polynomial, Equations (4.4) and (4.5) form a system of three equations with four unknowns. For convenience, we let $t_{d,i}^{\mathrm{f}}$ be the free parameter here. Then, combining the four free parameters that characterize the spatial PH Bézier curve $\mathbf{p}_{d,i}(\zeta_i)$, we can define, for the ith vehicle, a vector of free parameters as $\boldsymbol{\Xi}_i = \left[\, |\mathbf{d}_i^{\mathrm{i}}|, |\mathbf{d}_i^{\mathrm{f}}|, \phi_{i,0}, \phi_{i,2}, t_{d,i}^{\mathrm{f}} \,\right]^\top$.

4.3 Set of Constraints in Bézier Form

In this section, we re-formulate the set of constraints that was presented in Section 2.3 in terms of (PH) Bézier polynomials. In the previous sections, we derived that, if for the ith vehicle the boundary conditions

$$\begin{aligned} &\mathbf{p}_{d,i}(\zeta_i = 0) = \mathbf{p}_i^{\mathrm{i}} \,, \quad \gamma_i(\zeta_i = 0) = \gamma_i^{\mathrm{i}} \,, \quad \psi_i(\zeta_i = 0) = \psi_i^{\mathrm{i}} \,, \quad v_{d,i}(\hat{t}_{d,i} = 0) = v_i^{\mathrm{i}} \,, \\ &\mathbf{p}_{d,i}(\zeta_i = 1) = \mathbf{p}_i^{\mathrm{f}} \,, \quad \gamma_i(\zeta_i = 1) = \gamma_i^{\mathrm{f}} \,, \quad \psi_i(\zeta_i = 1) = \psi_i^{\mathrm{f}} \,, \quad v_{d,i}(\hat{t}_{d,i} = 1) = v_i^{\mathrm{f}} \,, \end{aligned} \tag{4.7}$$

and the vector $\boldsymbol{\Xi}_i = \left[\, |\mathbf{d}_i^{\mathrm{i}}|, |\mathbf{d}_i^{\mathrm{f}}|, \phi_{i,0}, \phi_{i,2}, t_{d,i}^{\mathrm{f}} \,\right]^\top$ are given, then a quintic PH Bézier curve of the form

$$\mathbf{p}_{d,i}(\zeta_i) = \sum_{k=0}^{5} \bar{\mathbf{p}}_{i,k} b_k^5(\zeta_i)$$

can be constructed that interpolates these endpoint conditions. Moreover, if the control points for the timing law

$$\theta_i(\hat{t}_{d,i}) = \sum_{k=0}^{2} \bar{\theta}_{i,k}\, b_k^2(\hat{t}_{d,i})$$

are chosen as follows:

$$\bar{\theta}_{i,0} = \frac{t_{d,i}^{\mathrm{f}} v_i^{\mathrm{i}}}{|\mathbf{d}_i^{\mathrm{i}}|}\,, \quad \bar{\theta}_{i,2} = \frac{t_{d,i}^{\mathrm{f}} v_i^{\mathrm{f}}}{|\mathbf{d}_i^{\mathrm{f}}|}\,, \quad \bar{\theta}_{i,1} = 3 - \bar{\theta}_{i,0} - \bar{\theta}_{i,2}\,,$$

then the function

$$\zeta_i(\hat{t}_{d,i}) = \sum_{k=0}^{3} \bar{\zeta}_{i,k} b_k^3(\hat{t}_{d,i})\,,$$

with control points

$$\bar{\zeta}_{i,0} = 0\,, \quad \bar{\zeta}_{i,1} = \frac{1}{3}\bar{\theta}_{i,0}\,, \quad \bar{\zeta}_{i,2} = \frac{1}{3}\left(\bar{\theta}_{i,0} + \bar{\theta}_{i,1}\right)\,, \quad \bar{\zeta}_{i,3} = 1\,,$$

also satisfies its boundary conditions $\zeta_i(0) = 0$ and $\zeta_i(1) = 1$. Hence, for any vector $\boldsymbol{\Xi}_i$, we can construct a pair of quintic spatial PH Bézier curve $\mathbf{p}_{d,i}(\zeta_i)$ and timing law $\theta_i(\hat{t}_{d,i})$ in quadratic Bézier polynomial form, such that the boundary conditions for $\mathbf{p}_{d,i}(\zeta_i)$ and $\zeta_i(\hat{t}_{d,i})$ are satisfied.

As discussed in Chapter 2, we seek to generate trajectories that are collision-free and meet the dynamic constraints of the vehicles. The dynamics along the trajectory are given by Equations (2.12)-(2.14) and their bounds in (2.15). In general, it is possible to preserve the Bézier structure in these equations and take full advantage of the favorable properties of Bézier curves. For those cases where this is not immediately possible, we aim to rewrite and transform the equations into Bézier (or rational Bézier) form. Evaluating these dynamic constraints involves finding the global minima and maxima of high-degree polynomials. However, as described in Section 3.2 and [17], finding the global minima and maxima of polynomial and rational Bézier curves is extremely computationally efficient. To begin with, we derive from the above-given spatial path $\mathbf{p}_{d,i}(\zeta_i)$, the first and second derivatives with respect to ζ_i:

$$\mathbf{p}'_{d,i}(\zeta_i) = 5 \sum_{k=0}^{4} \left(\bar{\mathbf{p}}_{i,k+1} - \bar{\mathbf{p}}_{i,k}\right) b_k^4(\zeta_i)\,, \tag{4.8a}$$

$$\mathbf{p}''_{d,i}(\zeta_i) = 20 \sum_{k=0}^{3} \left(\bar{\mathbf{p}}_{i,k+2} - 2\bar{\mathbf{p}}_{i,k+1} + \bar{\mathbf{p}}_{i,k}\right) b_k^3(\zeta_i)\,. \tag{4.8b}$$

Subsequently, as an inherent property of PH Bézier curves, the parametric speed $\sigma_i(\zeta_i)$ and its first deriva-

tive $\sigma_i'(\zeta_i)$ with respect to ζ_i are Bézier polynomials as well

$$\sigma_i(\zeta_i) = \sum_{k=0}^{4} \bar{\sigma}_{i,k} b_k^4(\zeta_i) \ , \tag{4.9a}$$

$$\sigma_i'(\zeta_i) = 4 \sum_{k=0}^{3} \left(\bar{\sigma}_{i,k+1} - \bar{\sigma}_{i,k}\right) b_k^3(\zeta_i) \, , \tag{4.9b}$$

where the control points $\bar{\sigma}_{i,k}$ are functions of $\bar{\mathcal{A}}_{i,0}$, $\bar{\mathcal{A}}_{i,1}$, and $\bar{\mathcal{A}}_{i,2}$:

$$\bar{\sigma}_{i,0} = |\bar{\mathcal{A}}_{i,0}|^2 \, , \qquad \bar{\sigma}_{i,1} = \frac{1}{2}\left(\bar{\mathcal{A}}_{i,0}\bar{\mathcal{A}}_{i,1}^* + \bar{\mathcal{A}}_{i,1}\bar{\mathcal{A}}_{i,0}^*\right) \, ,$$

$$\bar{\sigma}_{i,2} = \frac{1}{6}\left(\bar{\mathcal{A}}_{i,0}\bar{\mathcal{A}}_{i,2}^* + 4|\bar{\mathcal{A}}_{i,1}|^2 + \bar{\mathcal{A}}_{i,2}\bar{\mathcal{A}}_{i,0}^*\right) \, ,$$

$$\bar{\sigma}_{i,3} = \frac{1}{2}\left(\bar{\mathcal{A}}_{i,1}\bar{\mathcal{A}}_{i,2}^* + \bar{\mathcal{A}}_{i,2}\bar{\mathcal{A}}_{i,1}^*\right) \, , \qquad \bar{\sigma}_{i,4} = |\bar{\mathcal{A}}_{i,2}|^2 \, .$$

Now, we can use Equations (4.8) and (4.9), along with the expression for the timing law $\theta_i(\hat{t}_{d,i})$, to formulate the dynamic equations. Equation (2.12) shows that the flight-path angle $\gamma_i(\zeta_i)$ involves an $\arcsin(\cdot)$ function, which clearly is not of the Bézier form. Also, given the fact that the bounds for the flight-path angle are asymmetric and that $\gamma_i(\zeta_i)$ can take negative values, there is a need for a signed quantity with which an equivalent constraint can be constructed and preferably expressed in Bézier form. Fortunately, this is achieved by evaluating $\sin\left(\gamma_i(\zeta_i)\right)$ instead of $\gamma_i(\zeta_i)$, since firstly, the argument of the $\arcsin(\cdot)$ function in Equation (2.12) is a rational function, which can be rewritten as a rational Bézier curve and, secondly, the function $\sin(\cdot)$ is a strictly monotonically increasing function with the appropriate sign convention on the interval $[-\frac{1}{2}\pi, \frac{1}{2}\pi]$. Here, the assumption is made that $-\frac{1}{2}\pi < \gamma_{\min} < \gamma_{\max} < \frac{1}{2}\pi$, which is a valid assumption, considering the fact that constraints on the flight-path angles are not required when $|\gamma_i| \geq \frac{1}{2}\pi$ are allowed. Hence, Equation (2.12) can be transformed into a standard rational Bézier curve:

$$\sin\left(\gamma_i(\zeta_i)\right) = \frac{\langle\, \mathbf{p}_{d,i}'(\zeta_i), \hat{\mathbf{k}} \,\rangle}{\sigma_i(\zeta_i)} = \frac{\sum_{k=0}^{4} w_{\gamma_{\mathrm{s}i},k} \bar{\gamma}_{\mathrm{s}_i,k} b_k^4(\zeta_i)}{\sum_{k=0}^{4} w_{\gamma_{\mathrm{s}i},k} b_k^4(\zeta_i)} \, , \tag{4.10}$$

where $\bar{\gamma}_{\mathrm{s}_i,k}$ and $w_{\gamma_{\mathrm{s}i},k}$ are the control points and weights of the rational Bézier curve, respectively. Therefore, an equivalent constraint for the flight-path angle $\gamma_i(\zeta_i)$ can be expressed as follows

$$\sin\left(\gamma_{\min}\right) \leq \sin\left(\gamma_i(\zeta_i)\right) \leq \sin\left(\gamma_{\max}\right) \, .$$

Given the relation between $\hat{t}_{d,i}$ and t_d in (4.2), the desired speed profile $v_{d,i}$ in Equation (2.13) can be easily expressed as a Bézier polynomial, by first finding the composition of functions $(\mathbf{p}_{d,i}' \circ \zeta_i)(\hat{t}_{d,i})$ as outlined

in [27,32,33] and, subsequently, using the multiplication property of Bernstein basis polynomials to determine the control points of $v_{d,i}(\hat{t}_{d,i})$:

$$\begin{aligned} v_{d,i}(\hat{t}_{d,i}) &= \tfrac{1}{t^{\mathrm{f}}_{d,i}}\sigma_i(\zeta_i(\hat{t}_{d,i}))\theta_i(\hat{t}_{d,i}) \\ &= \sum_{k=0}^{14} \bar{v}_{d_i,k} b_k^{14}(\hat{t}_{d,i})\,. \end{aligned} \tag{4.11}$$

The acceleration profile along the path $a^{\mathrm{p}}_{d,i}(t_d)$ given in Equation (2.14a) is defined as the first derivative of $v_{d,i}(t_d)$ with respect to t_d. Hence, using Equation (4.11), we obtain the following Bézier polynomial for the acceleration profile as function of $\hat{t}_{d,i}$:

$$\begin{aligned} a^{\mathrm{p}}_{d,i}(\hat{t}_{d,i}) &= \frac{14}{t^{\mathrm{f}}_{d,i}} \sum_{k=0}^{13} \left(\bar{v}_{d_i,k+1} - \bar{v}_{d_i,k}\right) b_k^{13}(\hat{t}_{d,i}) \\ &= \sum_{k=0}^{13} \bar{a}_{d_i,k} b_k^{13}(\hat{t}_{d,i})\,. \end{aligned} \tag{4.12a}$$

In general, the equation for the acceleration profile $a^{\mathrm{t}}_{d,i}(t_d)$, as defined in Equation (2.14b), does not permit a Bézier polynomial form, due to the square root arising from the 2-norm. Hence, in order to obtain a Bézier polynomial for the framework to work with, the expression for $(a^{\mathrm{t}}_{d,i}(\hat{t}_{d,i}))^2$ is derived and evaluated against an *equivalent* bound. By doing so, the following Bézier polynomial is obtained:

$$\begin{aligned} (a^{\mathrm{t}}_{d,i}(\hat{t}_{d,i}))^2 &= \frac{1}{(t^{\mathrm{f}}_{d,i})^4} \left\| \mathbf{p}'_{d,i}(\zeta_i(\hat{t}_{d,i}))\dot{\theta}_i(\hat{t}_{d,i}) + \mathbf{p}''_{d,i}(\zeta_i(\hat{t}_{d,i}))\theta_i^2(\hat{t}_{d,i}) \right\|^2 \\ &= \sum_{k=0}^{26} \bar{a}_{d_i,k} b_k^{26}(\hat{t}_{d,i})\,, \end{aligned} \tag{4.12b}$$

where $\dot{\theta}_i(\hat{t}_{d,i}) = \mathrm{d}\theta_i(\hat{t}_{d,i})/\mathrm{d}\hat{t}_{d,i}$. The equations for $\dot{\gamma}_i(t_d)$ and $\dot{\psi}_i(t_d)$ in Equation (2.13) can be transformed into rational Bézier polynomials as well. This can be performed straightforwardly for the turn rate, since $\dot{\psi}_i(t_d)$ in Equation (2.13) is already a rational function. To evaluate the bounds on the rate of change of the flight-path angle, we derive the expression for $\dot{\gamma}_i^2(t_d)$ in terms of Bézier polynomials, instead of the rate of change of the flight-path angle $\dot{\gamma}_i(t_d)$, due the square root in the denominator of $\dot{\gamma}_i(t_d)$ in Equation (2.13). Equivalence of the constraint is achieved, if we assume that the bounds on $\dot{\gamma}_i(t_d)$ are of symmetric nature.

We obtain the following expression for $\dot{\gamma}_i^2(\hat{t}_{d,i})$:

$$\dot{\gamma}_i^2(\hat{t}_{d,i}) = \left[\frac{\sigma_i(\zeta_i(\hat{t}_{d,i}))\langle \mathbf{p}''_{d,i}(\zeta_i(\hat{t}_{d,i})), \hat{\mathbf{k}} \rangle - \sigma'_i(\zeta_i(\hat{t}_{d,i}))\langle \mathbf{p}'_{d,i}(\zeta_i(\hat{t}_{d,i})), \hat{\mathbf{k}} \rangle}{\sigma_i(\zeta_i(\hat{t}_{d,i})) \left(\langle \mathbf{p}'_{d,i}(\zeta_i(\hat{t}_{d,i})), \hat{\mathbf{i}} \rangle^2 + \langle \mathbf{p}'_{d,i}(\zeta_i(\hat{t}_{d,i})), \hat{\mathbf{j}} \rangle^2 \right)^{\frac{1}{2}}} \frac{\theta_i(\hat{t}_{d,i})}{t^{\mathrm{f}}_{d,i}} \right]^2$$

$$= \frac{\sum_{k=0}^{48} w_{\gamma_{\mathrm{d}i},k} \bar{\gamma}_{\mathrm{d}_i,k} b_k^{48}(\hat{t}_{d,i})}{\sum_{k=0}^{48} w_{\gamma_{\mathrm{d}i},k} b_k^{48}(\hat{t}_{d,i})}, \tag{4.13}$$

while $\dot{\psi}_i(\hat{t}_{d,i})$ is given by

$$\dot{\psi}_i(\hat{t}_{d,i}) = \frac{\langle \mathbf{p}'_{d,i}(\zeta_i(\hat{t}_{d,i})) \times \mathbf{p}''_{d,i}(\zeta_i(\hat{t}_{d,i})), \hat{\mathbf{k}} \rangle}{\langle \mathbf{p}'_{d,i}(\zeta_i(\hat{t}_{d,i})), \hat{\mathbf{i}} \rangle^2 + \langle \mathbf{p}'_{d,i}(\zeta_i(\hat{t}_{d,i})), \hat{\mathbf{j}} \rangle^2} \frac{\theta_i(\hat{t}_{d,i})}{t^{\mathrm{f}}_{d,i}}$$

$$= \frac{\sum_{k=0}^{24} w_{\psi_{\mathrm{d}i},k} \bar{\psi}_{\mathrm{d}_i,k} b_k^{24}(\hat{t}_{d,i})}{\sum_{k=0}^{24} w_{\psi_{\mathrm{d}i},k} b_k^{24}(\hat{t}_{d,i})}. \tag{4.14}$$

In order for the generated trajectories to meet the dynamic constraints of the vehicles, Equations (4.10)-(4.14) are required to satisfy the following bounds, which are *equivalent* to the bounds given in (2.15):

$$\begin{gathered} v_{d,\min} \leq v_{d,i}(\hat{t}_{d,i}) \leq v_{d,\max}, \quad \sin(\gamma_{\min}) \leq \sin(\gamma_i(\zeta_i)) \leq \sin(\gamma_{\max}), \\ \dot{\gamma}_i^2(\hat{t}_{d,i}) \leq \dot{\gamma}_{\max}^2, \quad |\dot{\psi}_i(\hat{t}_{d,i})| \leq \dot{\psi}_{\max}, \\ |a^{\mathrm{p}}_{d,i}(\hat{t}_{d,i})| \leq a_{d,\max} \quad \text{or} \quad (a^{\mathrm{t}}_{d,i}(\hat{t}_{d,i}))^2 \leq a_{d,\max}^2, \end{gathered} \tag{4.15}$$

for all $\zeta_i, \hat{t}_{d,i} \in [0,1]$.

Lastly, a minimum spatial clearance E_d has to be ensured for the generated trajectories to be collision-free. In Section 2.3, two strategies were presented to guarantee spatial deconfliction between the trajectories. Spatial separation is guaranteed if and only if the inequality in Equation (2.16) is satisfied. Since the spatial paths are PH Bézier curves, this inequality can be expressed as:

$$\min_{\substack{i,j=1,\ldots,N \\ i \neq j}} \left\| \sum_{k=0}^{5} \bar{\mathbf{p}}_{i,k} b_k^5(\zeta_i) - \sum_{k=0}^{5} \bar{\mathbf{p}}_{j,k} b_k^5(\zeta_j) \right\|^2 \geq E_d^2, \qquad \forall \zeta_i, \zeta_j \in [0,1]. \tag{4.16}$$

In the case when temporal separation is preferred over spatial separation, the path $\mathbf{p}_{d,i}(\zeta_i)$ has to be re-parameterized by the time variable $\hat{t}_{d,i}$. We have seen earlier that this can be performed efficiently since we are working with Bézier polynomials, and the composition of two Bézier polynomials is a Bézier polynomial.

It can be easily verified that $\mathbf{p}_{d,i}(\hat{t}_{d,i})$ is a degree 15 spatial PH Bézier curve. Hence, let

$$\mathbf{p}_{d,i}(\hat{t}_{d,i}) = \sum_{k=0}^{15} \tilde{\mathbf{p}}_{i,k} b_k^{15}(\hat{t}_{d,i}), \qquad \forall \hat{t}_{d,i} \in [0,1], \tag{4.17}$$

where $\tilde{\mathbf{p}}_{i,k}$ are the control points, which can be determined using the recursive algorithm for computing the control points of the composition of two Bézier polynomials. First, we consider the case that $t_{d,i}^{\mathrm{f}} = t_{d,j}^{\mathrm{f}} = T_d$ for $i, j = 1, \ldots, N$, i.e. missions where the vehicles arrive simultaneously at their desired destination. Then, temporal separation between the ith and jth vehicle is defined as

$$\min_{\substack{i,j=1,\ldots,N \\ i \neq j}} \left\| \sum_{k=0}^{15} (\tilde{\mathbf{p}}_{i,k} - \tilde{\mathbf{p}}_{j,k})\, b_k^{15}(\hat{t}_d) \right\|^2 \geq E_d^2, \quad \forall \hat{t}_d \in [0,1], \tag{4.18}$$

where now $\hat{t}_d = t_d / T_d$.

In the general case, when $t_{d,i}^{\mathrm{f}} \neq t_{d,j}^{\mathrm{f}}$ for $i, j = 1, \ldots, N$ and $i \neq j$, the issue arises that $\hat{t}_{d,i} \neq \hat{t}_{d,j}$ at time t_d, since the variables $\hat{t}_{d,i}$ and $\hat{t}_{d,j}$ are normalized by $t_{d,i}^{\mathrm{f}}$ and $t_{d,j}^{\mathrm{f}}$, respectively. Consider two trajectories $\mathbf{p}_{d,i}(\hat{t}_{d,i})$ and $\mathbf{p}_{d,j}(\hat{t}_{d,j})$ for vehicles i and j. Without loss of generality, we assume that $t_{d,i}^{\mathrm{f}} > t_{d,j}^{\mathrm{f}}$ and, hence, $t_{d_{ij}}^{\mathrm{f}} := \min\{t_{d,i}^{\mathrm{f}}, t_{d,j}^{\mathrm{f}}\} = t_{d,j}^{\mathrm{f}}$. Note that the mission for vehicle j has terminated for all $t_d > t_{d_{ij}}^{\mathrm{f}}$ and, thus, to guarantee temporal separation between vehicles i and j, it is sufficient to satisfy

$$\left\| \mathbf{p}_{d,i}(\hat{t}_{d,i}) - \mathbf{p}_{d,j}(\hat{t}_{d,j}) \right\|^2 \geq E_d^2, \quad \forall\, t_d \in [0, t_{d_{ij}}^{\mathrm{f}}]. \tag{4.19}$$

Since $\hat{t}_{d,i} \neq \hat{t}_{d,j}$ at time t_d, Equation (4.19) cannot be evaluated straightforwardly with the available algorithms. However, this can be addressed by subdividing one of the trajectories using the *de Casteljau* algorithm. A description of the *de Casteljau* algorithm can be found in Chapter 3. Because $t_{d,i}^{\mathrm{f}} > t_{d,j}^{\mathrm{f}}$, we use the *de Casteljau* algorithm to subdivide the trajectory of vehicle i at $\alpha = t_{d_{ij}}^{\mathrm{f}} / t_{d,i}^{\mathrm{f}}$. From the *de Casteljau* algorithm we obtain a Bézier curve $\mathbf{p}_{d,i}^{\alpha}(\hat{t}_{d,i}^{\alpha})$ defined on $\hat{t}_{d,i}^{\alpha} \in [0,1]$, which corresponds to the trajectory segment $\mathbf{p}_{d,i}(\hat{t}_{d,i})$ for $t_d \in [0, t_{d_{ij}}^{\mathrm{f}}]$. If we let $\hat{t}_{d,i}^{\alpha}$ evolve at the same pace as $\hat{t}_{d,j}$, that is $\hat{t}_{d,i}^{\alpha} = \hat{t}_{d,j}$, then it can be shown that

$$\mathbf{p}_{d,i}(\hat{t}_{d,i}) = \mathbf{p}_{d,i}^{\alpha}(\hat{t}_{d,j}), \qquad \forall\, t_d \in [0, t_{d_{ij}}^{\mathrm{f}}].$$

This result is proven in Lemma 1 below. First, we state the following Proposition, which is instrumental in the proof of the Lemma.

Proposition 1 *Given a degree n Bézier polynomial $r(\zeta) = \sum_{k=0}^{n} \bar{r}_k b_k^n(\zeta)$ and constant $\alpha \in (0,1)$, let $r^{\alpha}(\zeta^{\alpha}) = \sum_{k=0}^{n} \bar{r}_k^{\alpha} b_k^n(\zeta^{\alpha})$ be the Bézier polynomial that identically describes the subsegment of $r(\zeta)$ defined*

over $\zeta \in [0, \alpha]$*, resulting from the subdivision of the curve at* $\zeta = \alpha$ *using the* de Casteljau *algorithm. The following relation holds*

$$r^{\alpha}(\zeta^{\alpha}) = r(\alpha\zeta^{\alpha}), \qquad \forall \zeta^{\alpha} \in [0, 1].$$

Proof: The proof is given in Appendix B.1. □

Lemma 1 *Consider the trajectories* $\mathbf{p}_{d,i}(\hat{t}_{d,i})$ *and* $\mathbf{p}_{d,j}(\hat{t}_{d,j})$ *for vehicle* i *and* j*, respectively, where* $\hat{t}_{d,i}, \hat{t}_{d,j} \in [0, 1]$ *and, without loss of generality, assume that* $t^{\mathrm{f}}_{d,i} > t^{\mathrm{f}}_{d,j}$*. Letting* $\alpha = t^{\mathrm{f}}_{d_{ij}}/t^{\mathrm{f}}_{d,i}$*, define* $\mathbf{p}^{\alpha}_{d,i}(\hat{t}^{\alpha}_{d,i})$ *to be the Bézier curve that identically describes the subsegment of* $\mathbf{p}_{d,i}(\hat{t}_{d,i})$ *defined over* $\hat{t}_{d,i} \in [0, \alpha]$*, resulting from the subdivision of the curve at* $\hat{t}_{d,i} = \alpha$ *using the* de Casteljau *algorithm. The following relation holds*

$$\mathbf{p}_{d,i}(\hat{t}_{d,i}) = \mathbf{p}^{\alpha}_{d,i}(\hat{t}_{d,j}), \qquad \forall\, t_d \in [0, t^{\mathrm{f}}_{d_{ij}}],$$

where $t^{\mathrm{f}}_{d_{ij}} := \min\{t^{\mathrm{f}}_{d,i}, t^{\mathrm{f}}_{d,j}\} = t^{\mathrm{f}}_{d,j}$.

Proof: The proof is given in Appendix B.2. □

Hence, using Lemma 1, Equation (4.19) can be rewritten as a function of a single parameterizing variable as

$$\left\|\mathbf{p}^{\alpha}_{d,i}(\hat{t}_{d,j}) - \mathbf{p}_{d,j}(\hat{t}_{d,j})\right\|^2 \geq E_d^2, \quad \forall\, \hat{t}_{d,j} \in [0, 1].$$

With this result, and noting that the trajectories $\mathbf{p}_{d,i}(\hat{t}_{d,i})$ are given by Equation (4.17), the temporal separation constraint for the case of N vehicles can now be expressed as

$$\min_{\substack{i,j=1,\ldots,N \\ i \neq j}} \left\| \sum_{k=0}^{15} \tilde{\mathbf{p}}_{ij,k} b_k^{15}(\hat{t}_{d_{ij}}) \right\|^2 \geq E_d^2, \quad \forall\, \hat{t}_{d_{ij}} \in [0, 1], \tag{4.20}$$

where $\hat{t}_{d_{ij}} = t_d/t^{\mathrm{f}}_{d_{ij}}$ with $t^{\mathrm{f}}_{d_{ij}} := \min\{t^{\mathrm{f}}_{d,i}, t^{\mathrm{f}}_{d,j}\}$, and

$$\tilde{\mathbf{p}}_{ij,k} = \begin{cases} \tilde{\mathbf{p}}^{\alpha}_{i,k} - \tilde{\mathbf{p}}_{j,k}, & \text{if } \; t^{\mathrm{f}}_{d,i} > t^{\mathrm{f}}_{d,j} \\ \tilde{\mathbf{p}}_{i,k} - \tilde{\mathbf{p}}^{\alpha}_{j,k}, & \text{if } \; t^{\mathrm{f}}_{d,i} < t^{\mathrm{f}}_{d,j} \end{cases}.$$

In the expressions above, $\tilde{\mathbf{p}}^{\alpha}_{i,k}$ and $\tilde{\mathbf{p}}^{\alpha}_{j,k}$ are the control points that are obtained from curve subdivision with the *de Casteljau* algorithm. Finally, we note that there exist extremely computationally efficient algorithms, associated with Bézier curves, to evaluate Equations (4.16) and (4.18) (or (4.20)).

4.4 Trajectory Generation: Constrained Optimization

The constrained optimization problem presented in Equation (2.3) can now be re-formulated for the framework that specifically uses quintic PH Bézier curves to represent the spatial paths, and quadratic Bézier polynomials for the timing laws:

$$\begin{aligned} &\min_{\boldsymbol{\Xi}_i \times \ldots \times \boldsymbol{\Xi}_N} \quad J(\cdot) \\ &\text{subject to} \quad \text{dynamic constraints of the vehicles (4.15)}, \\ &\qquad\qquad\ \ \text{spatial deconfliction ((4.16), (4.18) or (4.20))}, \\ &\qquad\qquad\ \ \text{mission-specific constraints}, \end{aligned} \tag{4.21}$$

where $\boldsymbol{\Xi}_i = \left[\, |\mathbf{d}_i^{\mathrm{i}}|, |\mathbf{d}_i^{\mathrm{f}}|, \phi_{i,0}, \phi_{i,2}, t_{d,i}^{\mathrm{f}} \,\right]^\top$ represents the vector of optimization parameters for the ith vehicle, and $J(\cdot)$ is a given objective function. Note that the boundary conditions in (4.7) and the boundary conditions on the function $\zeta_i(\hat{t}_{d,i})$ are automatically satisfied and, hence, do not impose extra constraints. Lastly, if simultaneous time-of-arrival is required, then the additional constraints would be given by Equation (2.10). However, this can be implemented simpler, by substituting for all $t_{d,i}^{\mathrm{f}}$ with T_d in the equations of Sections 4.2 and 4.3.

The resulting constrained optimization problem is nonlinear and nonconvex and, hence, the solutions are in general suboptimal [13]. The problem belongs to the class of generalized semi-infinite programming (SIP) problems, which by nature are nonsmooth, since the gradients of the objective function or constraints may not exist. However, the *generalized directional derivative* $f_i^\circ(x; d)$ and the *subdifferential* $\partial f_i(x)$ of each of the nonsmooth functions $f_i(x)$, as defined in [24], have specific structures for this type of problems. Especially, the subdifferential $\partial f_i(x)$ plays an important role in algorithms solving nonsmooth optimization problems. Algorithms for solving SIP problems can be found in the literature, such as in [76]. However, they are mostly based on discretization and approximation of the functions.

In our formulation, by virtue of using Bézier curves, we are able to solve the semi-infinite programming problem as a (finite) nonsmooth problem, for which we can compute the *whole* subdifferential $\partial f_i(x)$ of each of the nonsmooth functions $f_i(x)$. The latter is in general not possible, and existing solvers for nonsmooth optimization problems rely on approximating the whole subdifferential, such as the *bundle methods* [4]. In Chapter 5 we will discuss in more detail the theory of nonsmooth optimization, and in Chapter 6 we formulate a *distributed* algorithm for our nonsmooth cooperative trajectory-generation problem, where we exploit the knowledge of the whole subdifferential.

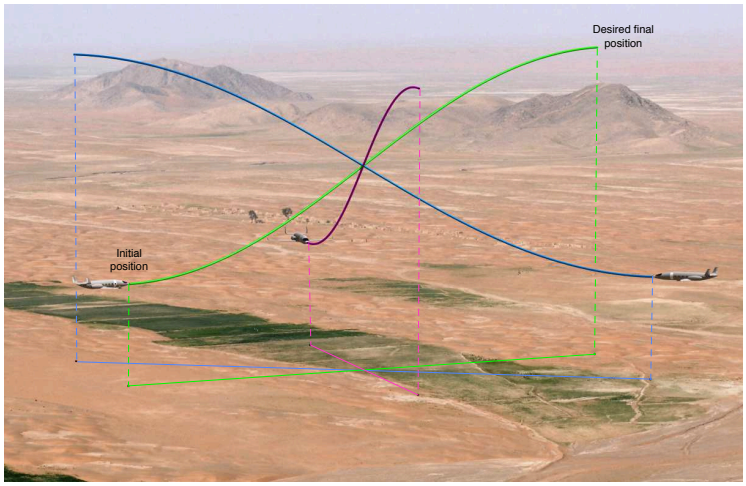


Figure 4.1: Artist's impression of the simulation scenario for a team of three fixed-wing UAVs.

4.5 Simulation Examples

In this section, we demonstrate the efficacy of the cooperative trajectory-generation framework by considering two simulation examples of teams of multiple UAVs executing a cooperative mission. The cooperative trajectory-generation framework, as outlined in Sections 4.1-4.3, is implemented in Matlab® on a desktop computer with Intel® Core™ i5-3470 CPU 3.20GHz, 8GB of RAM and running 64-bit Windows 7. Standard Matlab® routines are used to solve the constrained nonlinear programming.

4.5.1 Fixed-Wing UAVs Example

In this example, a simulation scenario will be presented where three fixed-wing UAVs are tasked to converge to and follow three spatially deconflicted paths and arrive at their final destinations at the same time. Representative examples of such missions are simultaneous monitoring of multiple targets located at different positions, and time efficient retrieval of assets after a survey mission. Note that these missions impose only *relative* temporal constraints on the arrival of the UAVs.

The simulation scenario is depicted in Figure 4.1. Three fixed-wing UAVs are tasked to arrive at their respective final positions simultaneously. The initial and final flight conditions (see Table 4.1) are chosen such that, if the trajectory generation was performed individually rather than cooperatively, the paths would intersect at a single point where the UAVs collide with each other. We demonstrate that the proposed cooperative trajectory-generation framework generates collision-free trajectories by enforcing spatial deconfliction through spatial or temporal separation of the vehicles.

In the planning phase, we generate a desired trajectory for each UAV so as to cooperatively execute the

Table 4.1: Flight conditions and dynamic constraints of the fixed-wing UAVs.

		UAV 1	UAV 2	UAV 3
$\mathbf{p}_d^{\mathrm{i}}$	[km]	(0 , 3.00 , 3.00)	(2.60 , −1.50 , 3.00)	(−2.60 , −1.50 , 3.00)
v_d^{i}	[m/s]	25	25	25
γ^{i}	[deg]	0	0	0
ψ^{i}	[deg]	−90	150	30
$\mathbf{p}_d^{\mathrm{f}}$	[km]	(0 , −3.00 , 4.00)	(−2.60 , 1.50 , 4.00)	(2.60 , 1.50 , 4.00)
v_d^{f}	[m/s]	25	25	25
γ^{f}	[deg]	0	0	0
ψ^{f}	[deg]	−90	150	30
$v_{d,\min}$	[m/s]	18	18	18
$v_{d,\max}$	[m/s]	32	32	32
$a_{d,\max}$	[m/s^2]	10	10	10
$\gamma_{\min}$	[deg]	−20	−20	−20
$\gamma_{\max}$	[deg]	30	30	30
$\dot{\gamma}_{\max}$	[deg/s]	11.46	11.46	11.46
$\dot{\psi}_{\max}$	[deg/s]	11.46	11.46	11.46

mission shown in Figure 4.1. The values for the boundary conditions are given in Table 4.1, along with the dynamic constraints for each individual aircraft. To show the efficacy of the framework in generating desired deconflicted trajectories, three different cases are considered. In Case I, no minimum separation constraints are imposed; Case II guarantees a minimum separation between the vehicles via spatial separation; while Case III ensures collision-free trajectories through temporal separation. In all three cases, the generated desired trajectories satisfy the dynamic constraints of the vehicles and also the temporal specifications of the mission.

Figure 4.2 shows the flight paths for Case I. The separation between the vehicles is presented in Figure 4.2b and separations between the paths are given in Figures 4.2c-4.2e. Since no deconfliction constraints are imposed here, by construction of the example, the three generated paths intersect at one point as expected, and the UAVs arrive at the intersection simultaneously. The collision occurs around $t_d = 150$ s (see Figure 4.2b). Hence, although the generated trajectories are *flyable*, they are under no circumstances *feasible*.

Case II demonstrates the generation of collision-free trajectories through spatial separation, where the constraints are given in Equation (4.16). The results are presented in Figure 4.3. It can be observed that the three trajectories are separated both in space as well as in time, since the temporal separation is always greater than or equal to the spatial separation.

In the third and final case, spatial deconfliction is ensured via temporal separation (Equation (4.20)). The generated trajectories for this case are given in Figures 4.4 and 4.5. Comparison of Figure 4.4 with Figure 4.2 shows that the generated spatial paths in both cases do not differ much. However, deconfliction through temporal separation between the aircraft is ensured in Case III by adjusting the speed profiles of

Table 4.2: Flight conditions and dynamic constraints of the multirotor UAVs.

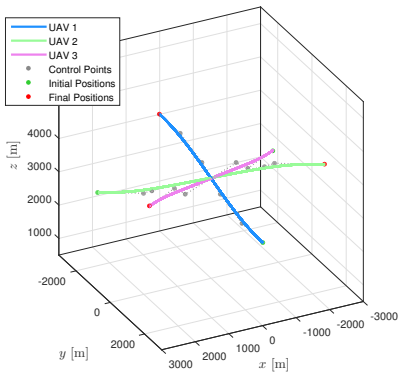
		UAV 1	UAV 2	UAV 3
$\mathbf{p}_d^{\mathrm{i}}$	[m]	$(0, 2.50, 1.00)$	$(5.00, 5.00, 1.00)$	$(3.00, 1.00, 1.00)$
v_d^{i}	[m/s]	0	0	0
$\mathbf{p}_d^{\mathrm{f}}$	[m]	$(45.00, 60.00, 10.00)$	$(45.00, 63.00, 10.00)$	$(45.00, 66.00, 10.00)$
v_d^{f}	[m/s]	4.00	4.00	4.00
$v_{d,\max}$	[m/s]	6.76	6.76	6.76
$a_{d,\max}$	[m/s^2]	4.28	4.28	4.28

the UAVs along their respective paths. This is an example where the benefit of introducing the timing law comes to its full right. Here, independent adjustment of the speed profiles from the spatial paths is achieved through the decoupling of the temporal element from the spatial element of the trajectory by the timing law. Lastly, from Figure 4.5 it is clear that the generated desired trajectories are flyable, as all dynamic constraints are satisfied.

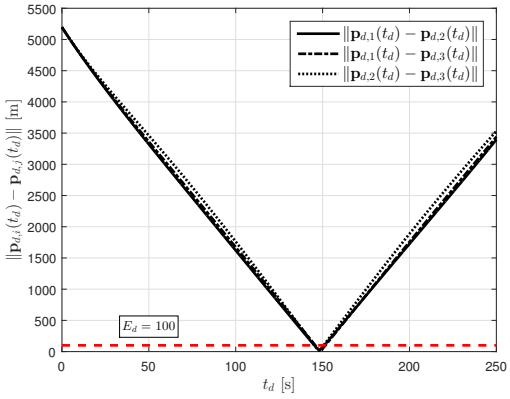
4.5.2 Multirotor UAVs Example

In this simulation example, a team of three multirotors, equipped with cameras, are tasked to inspect a road. The complete simulation can be found in [21]. During the planning phase, a desired trajectory for each UAV is generated so as to cooperatively execute the transition from vehicle launch to the start of the road search mission. At their initial holding areas, the UAVs are hovering stationary, and they are required to arrive at the desired starting point of the road simultaneously. This transition phase is an example of a typical multi-vehicle cooperative mission. The numerical values for the boundary conditions for this part of the mission are given in Table 4.2, along with the dynamic constraints for each individual multirotor. The number of optimization variables is $4N + 1$, while the number of inequality constraints is $\frac{1}{2}N(N-1) + 2N$. The latter includes the evaluation of the minimum distance between $\frac{1}{2}N(N-1)$ pairs of trajectories. The computation time taken to generate the set of trajectories for this particular example, where $N = 3$, is 12.4 s.

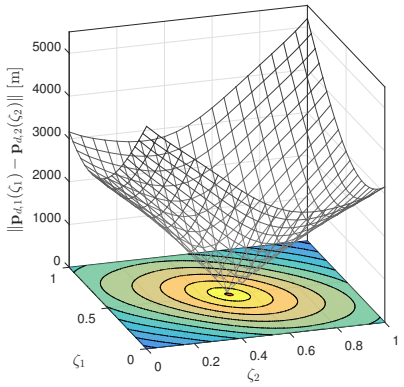
Figure 4.6 shows the flight paths for transition phase. The separation between the vehicles is presented in Figure 4.6b and the separations between the paths are given in Figures 4.6c-4.6e. Recall, that deconfliction is enforced in this mission by temporal separation. Therefore, although the minimum spatial separations between the paths are less than the required $E_d = 2$ meters during the planning phase, as shown in Figures 4.6c-4.6e, the algorithm ensures that the vehicles are sufficiently separated from each other at any point in time t_d (Figure 4.6b). From Figure 4.7 it is clear that the generated desired trajectories do not violate the maximum permissible speed and total acceleration. Lastly, note that the trajectory-generation framework, by introducing the timing law, can cope with zero-speed conditions that are characteristic to multirotors.



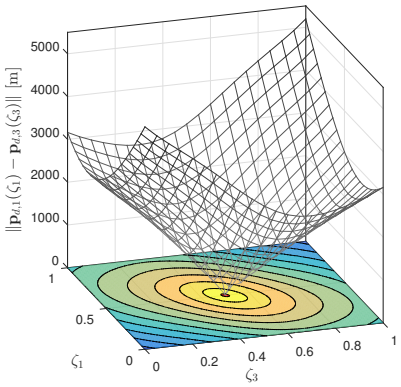
(a) Three-dimensional flight paths



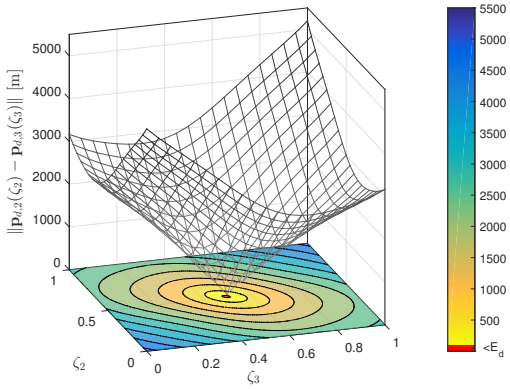
(b) Vehicle separation



(c) Path separation UAV 1 and 2

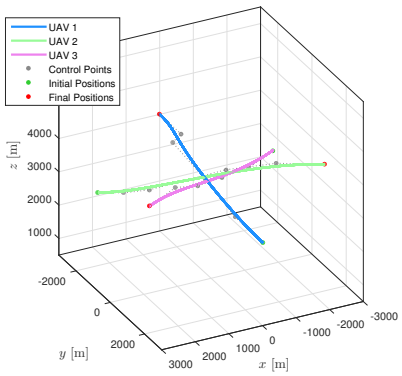


(d) Path separation UAV 1 and 3

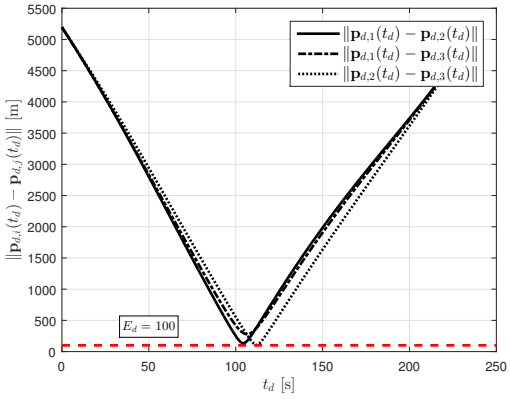


(e) Path separation UAV 2 and 3

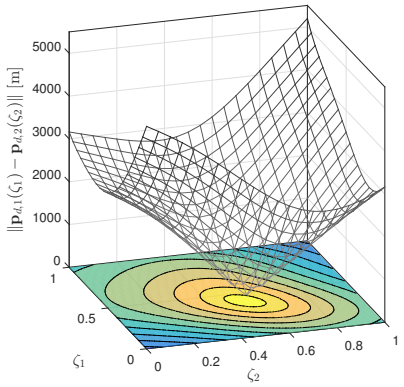
Figure 4.2: Three-dimensional trajectories for a team of three cooperating UAVs. Case I: no spatial deconfliction is imposed. Computation time is 5.6 s.



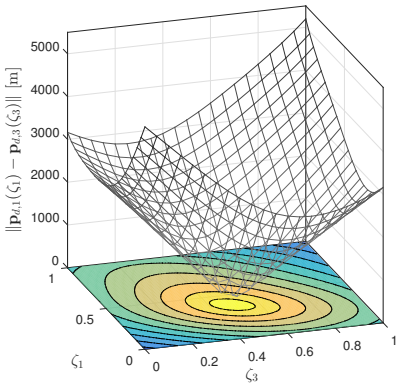
(a) Three-dimensional flight paths



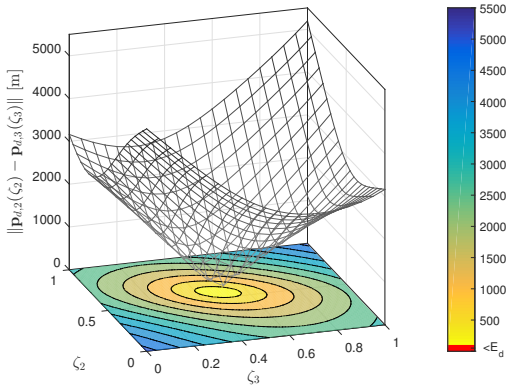
(b) Vehicle separation



(c) Path separation UAV 1 and 2

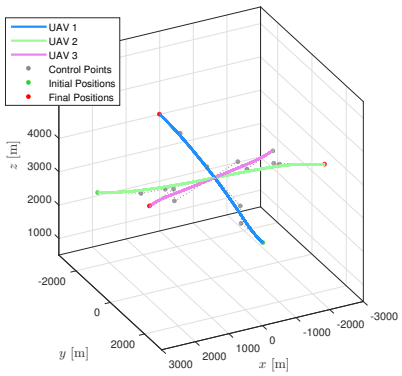


(d) Path separation UAV 1 and 3

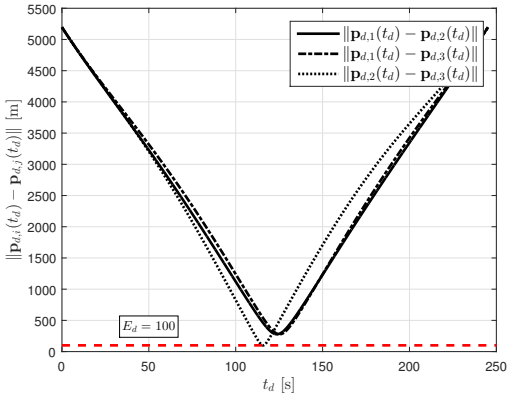


(e) Path separation UAV 2 and 3

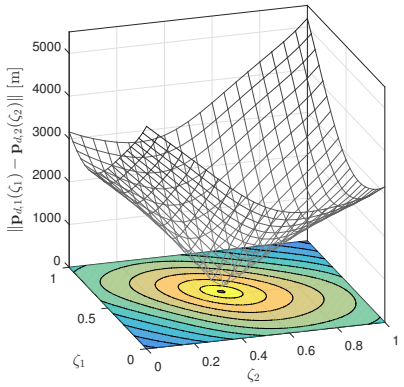
Figure 4.3: Three-dimensional trajectories for a team of three cooperating UAVs. Case II: spatial deconfliction is ensured through spatial separation. Computation time is 16.2 s.



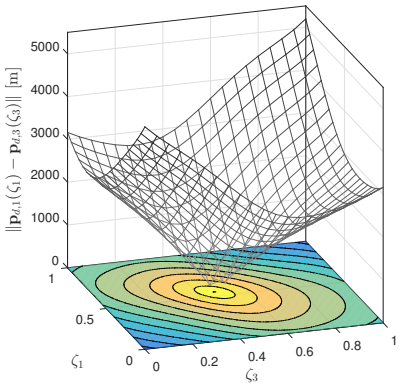
(a) Three-dimensional flight paths



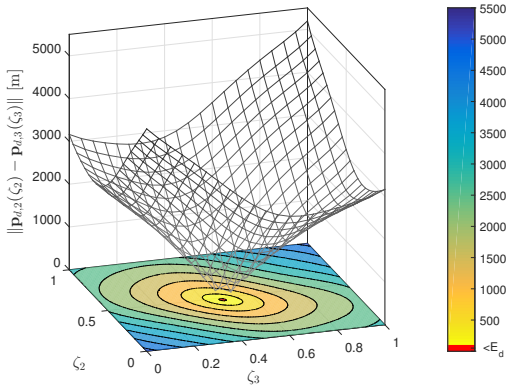
(b) Vehicle separation



(c) Path separation UAV 1 and 2

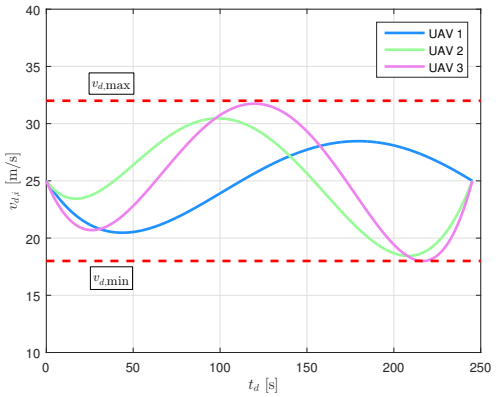


(d) Path separation UAV 1 and 3

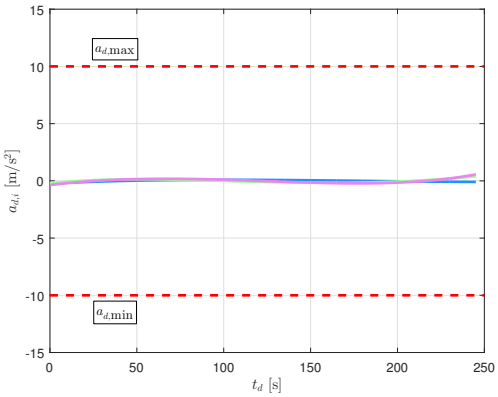


(e) Path separation UAV 2 and 3

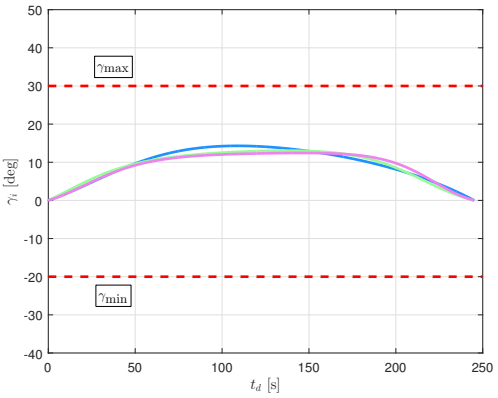
Figure 4.4: Three-dimensional trajectories for a team of three cooperating UAVs. Case III: spatial deconfliction is ensured through temporal separation. Computation time is 14.5 s.



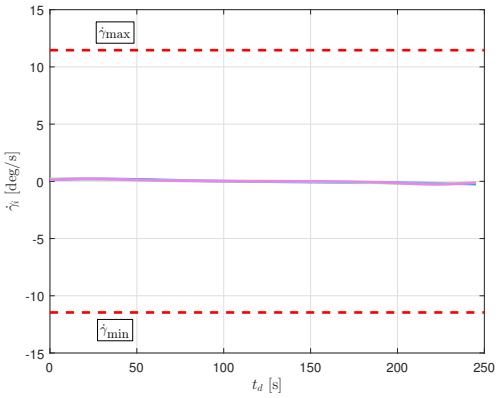
(a) Speed profile $v_{d,i}(t_d)$



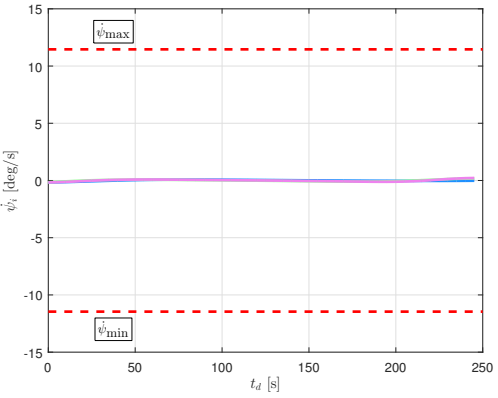
(b) Acceleration profile $a_{d,i}(t_d)$



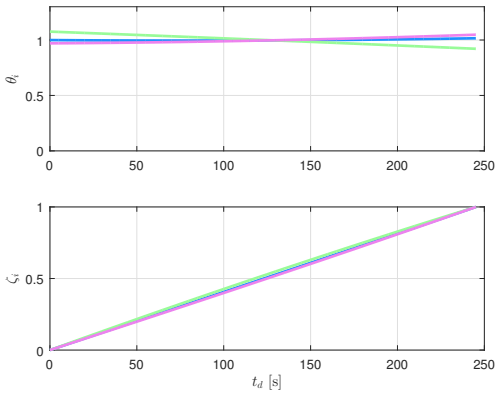
(c) Flight-path angle $\gamma_i(t_d)$



(d) Rate of change of flight-path angle $\dot{\gamma}_i(t_d)$

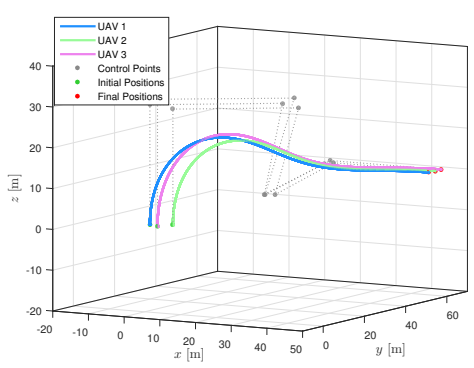


(e) Turn rate $\dot{\psi}_i(t_d)$

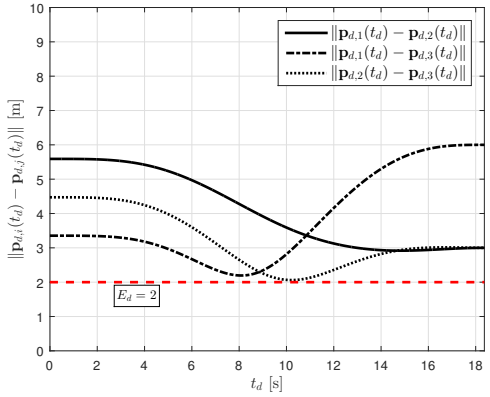


(f) Timing law θ_i and the function $\zeta_i(t_d)$

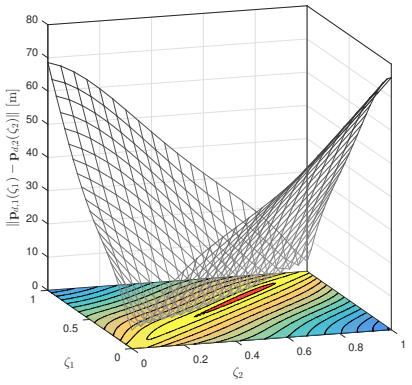
Figure 4.5: Dynamic constraints and timing laws for a team of three cooperating UAVs. Case III: spatial deconfliction is ensured through temporal separation.



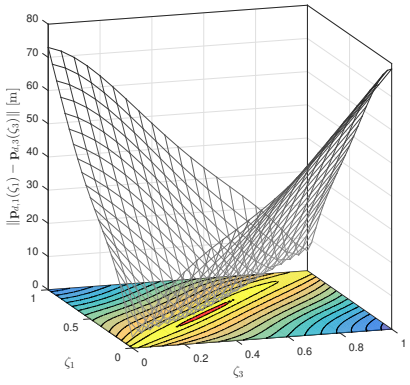
(a) Three-dimensional flight paths



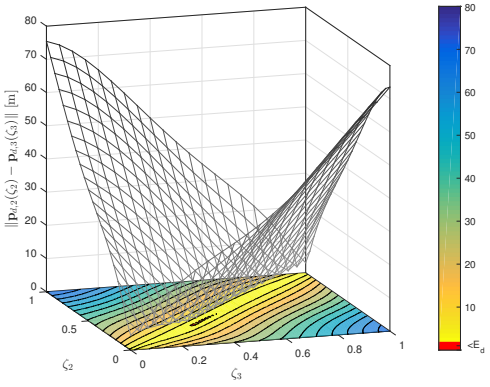
(b) Vehicle separation



(c) Path separation UAV 1 and 2

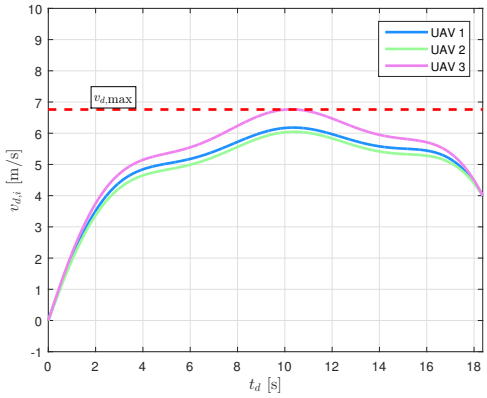


(d) Path separation UAV 1 and 3

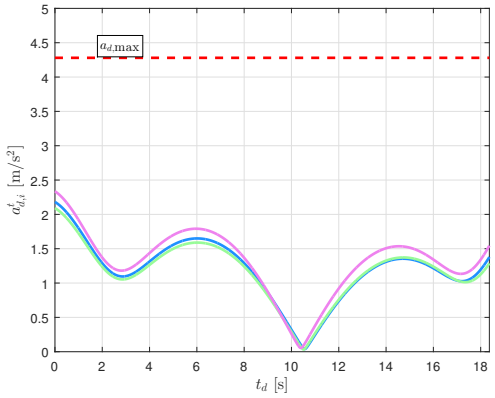


(e) Path separation UAV 2 and 3

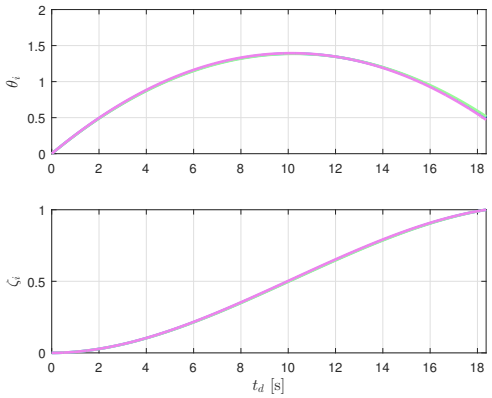
Figure 4.6: Three-dimensional trajectories for a team of three cooperating multirotors. Spatial deconfliction is ensured through temporal separation. The computation time is 12.4 s.



(a) Speed profile $v_{d,i}(t_d)$



(b) Acceleration profile $a^t_{d,i}(t_d)$



(c) Timing law θ_i and the function $\zeta_i(t_d)$

Figure 4.7: Dynamic constraints and timing laws for a team of three cooperating multirotors. Spatial deconfliction is ensured through temporal separation.

Chapter 5

Nonsmooth Optimization

While the previous chapters discussed in detail the formulation of a cooperative trajectory-generation framework, this chapter characterizes the resulting constrained nonlinear optimization problem that is given by Equation (4.21). The reasons for a deeper understanding of the optimization problem at hand are twofold. First, it is well-known that every optimization problem can be solved much more efficiently with tailor-made algorithms than one-size-fits-all solutions. In fact, the standard Matlab® routines that are used in Chapter 4 are not suitable and, therefore, inefficient for our class of optimization problems. Second, in view of our quest to distribute the cooperative trajectory-generation framework over the team of vehicles, readily available optimization software and algorithms cannot be easily used, since they are only designed to be executed in a centralized way. Relevant work on distributed optimization has been reported in the literature; see, for example, [46, 70, 71, 89] and the references therein. In general, these distributed algorithms are also only applicable to certain classes of optimization problems and, more than often, our trajectory-generation framework does not satisfy the underlying assumptions.

Hence, this chapter will focus on the characterization of the *centralized* optimization problem first. We shall see in Section 5.1 that the trajectory-generation framework belongs to the class of *semi-infinite programming* (SIP) problems, which are nonsmooth, i.e. non-differentiable, by nature. Since in this case, we cannot apply smooth optimization techniques that are vastly based on the existence of gradients, Sections 5.2 and 5.3 subsequently present the mathematical notions and tools commonly used in nonsmooth analysis and optimization theory, respectively. The next chapter will propose a distributed algorithm that is suitable for application to our nonsmooth optimization problem.

5.1 Nonsmooth Cooperative Trajectory Generation

In this section, we discuss the cooperative trajectory-generation problem in more detail from an optimization point of view. The aim is to determine the mathematical properties, so that the right apparatus can be used in search of a viable solution to the distributed cooperative trajectory-generation problem. We start

by restating the (centralized) cooperative trajectory-generation framework as formulated in Chapter 4 and given by Equation (4.21):

$$\min_{\boldsymbol{\Xi}_i \times \ldots \times \boldsymbol{\Xi}_N} \; J(\cdot)$$

$$\begin{aligned} \text{subject to} \;& \text{dynamic constraints of the vehicles (4.15)}, \\ & \text{spatial deconfliction ((4.16), (4.18) or (4.20))}, \\ & \text{mission-specific constraints}, \end{aligned}$$

where the vector of optimization variables $\boldsymbol{\Xi}_i$ completely determines the spatial trajectory for vehicle i, that is decomposed into a spatial path and a timing law, represented by quintic PH Bézier curves and quadratic Bézier polynomials, respectively. Now let us take a closer look at the functions determining the constraints. An example of such a function is the speed profile $v_{d,i}(\hat{t}_{d,i})$ for vehicle i. We have seen that the speed profile $v_{d,i}(\hat{t}_{d,i})$ was expressed as a Bézier polynomial and presented in Equation (4.11) as

$$v_{d,i}(\hat{t}_{d,i}) = \sum_{k=0}^{14} \bar{v}_{d_i,k} b_k^{14}(\hat{t}_{d,i}) \,.$$

Note that the above notation of the speed profile $v_{d,i}(\hat{t}_{d,i})$ reflects the speed profile for a *given* vector $\boldsymbol{\Xi}$. In fact, the control points of the cost and constraint Bézier polynomials are functions of the vector $\boldsymbol{\Xi}$. This is in essence the objective of the trajectory-generation problem: find an optimal vector $\boldsymbol{\Xi}$, such that the cost function $J(\cdot)$ is minimized while the constraints are satisfied. Hence, to gain more insight in the behavior of these functions, they should be written explicitly as a function of the optimization variable $\boldsymbol{\Xi}$. As an example, consider the constraint on the maximum speed shown in Equation (4.15). Then this particular constraint can be written as

$$v_{d,i}(\boldsymbol{\Xi}, \hat{t}_{d,i}) - v_{d,\max} \leq 0 \,.$$

It is obvious that the above inequality is equivalent to

$$g(\boldsymbol{\Xi}) - v_{d,\max} \leq 0 \,,$$

where $g(\boldsymbol{\Xi})$ is defined as

$$g(\boldsymbol{\Xi}) := \max_{\hat{t}_{d,i} \in [0,1]} v_{d,i}(\boldsymbol{\Xi}, \hat{t}_{d,i}) \,.$$

In this representation, it is immediately clear that the optimization problem is not as benign as it initially appeared to be. First, the maximum function sets off all alarm bells as it is non-differentiable and turns the cooperative trajectory-generation framework into a *nonsmooth optimization problem*. Second, to make matters worse, the dimensionless variable $\hat{t}_{d,i}$ is defined on the compact (continuous) interval $[0, 1]$. Therefore, in contrast to a case where $\hat{t}_{d,i}$ takes values from a discrete set, the constraint $g(\boldsymbol{\Xi}) - v_{d,\max} \leq 0$ actually consists of an infinite number of inequality constraints. These optimization problems are termed *semi-infinite programming* (SIP) problems since finitely many variables are subject to infinitely many inequality constraints [43, 75, 76, 92, 93, 101].

SIP problems commonly arise in optimization-based engineering, for example in worst-case structural or control systems designs [75]. Therefore, there is an extensive amount of research available in the literature. Early works are summarized in [43, 93], whereas present day approaches are described in [92, 101]. SIP problems are hard to solve and, in fact, dealing with nonsmooth functions is not the main difficulty. In order for a point $\boldsymbol{\Xi}$ to be *feasible*, each constraint has to be satisfied for all $\hat{t}_{d,i} \in [0, 1]$ at $\boldsymbol{\Xi}$. In other words, obtaining a local maximum is not sufficient; it is necessary to find the *global* maximum over $\hat{t}_{d,i} \in [0, 1]$ at the point $\boldsymbol{\Xi}$. Results in global optimization show that finding a global maximum for nonlinear functions is far from a trivial task and is definitely computationally expensive. Additionally, the fact that the functions $g(\boldsymbol{\Xi})$ are non-differentiable is not making it any easier either. In general, existing algorithms for solving SIP problems resort to discretizing the set over which the global maxima of the constraint functions have to be computed. By doing so, the infinitely constrained optimization problem is transformed into a finitely constrained optimization problem. The latter is known as a finite min-max constrained optimization problem. An infinite sequence of approximating problems is obtained by refining the discretization at each step and consistency conditions are derived for which the solutions of the approximating problems converge to the solution of the original problem. Several works based on this approach can be found in [74, 76, 80–82, 87]. It must be noted, that these algorithms are formulated such that they are to be executed in a centralized way. To the best of our knowledge, distributed algorithms for solving SIP problems are not reported in the literature.

In what follows, we will define a generic SIP problem without specifying any particular function from Chapter 4, for example the speed profile $v_{d,i}$. Instead we will consider generic (constraint) functions of the same form and, therefore, impose the same mathematical properties. As a consequence, at least for now, we will abandon the naming convention for the variables and functions from the previous chapters, and switch to a more commonly used notation in the field of optimization. For example, the vector of optimization variables $\boldsymbol{\Xi}$ will be denoted by x, and should not be confused with the x-coordinates of vehicles. Thus,

consider the following constrained optimization problem:

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && g(x) \leq 0\,, \end{aligned} \tag{5.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is locally Lipschitz continuous at $x \in \mathbb{R}^n$, that is there exists some $\epsilon > 0$ such that

$$|f(y) - f(z)| \leq K\|y - z\|\,, \qquad \text{for all } y, z \in \mathcal{B}(x;\epsilon)\,,$$

with K being the Lipschitz constant and $\mathcal{B}(x;\epsilon)$ being defined as the open ball with center x and radius ϵ. The vector-valued function $g : \mathbb{R}^n \to \mathbb{R}^m$ is defined as

$$g(x) := [\, g_1(x), \ldots, g_m(x)\,]^\top\,,$$

with $g_i : \mathbb{R}^n \to \mathbb{R}$ locally Lipschitz continuous. Moreover, the functions $g_i(x)$ are of the form

$$g_i(x) = \max_{\zeta \in \hat{\mathcal{Z}}_i} \gamma_i(x, \zeta)\,,$$

where $\hat{\mathcal{Z}}_i \subset \mathbb{R}^{l_i}$. The functions $\gamma_i : \mathbb{R}^n \times \mathbb{R}^{l_i} \to \mathbb{R}$ are continuous and their gradients $\nabla_x \gamma_i(\cdot,\cdot)$ exist and are continuous. It can be verified that the cooperative trajectory-generation framework, presented in Chapter 4, falls under the class of SIP optimization problems as described by (5.1). In particular, the functions $\gamma_i(x, \zeta)$ in our trajectory-generation framework are expressed as Bézier polynomials. In Chapter 3 we have seen that the global minima and maxima of Bézier polynomials can be efficiently computed, without resorting to discretization, using an iterative algorithm that exploits the properties of Bézier polynomials. Therefore, not only can we evaluate the values of constraints g_i at a point x, due to the fact that the *global maxima* of each Bézier polynomial $\gamma_i(x, \zeta)$ are found, we can also perform these computations very efficiently. This allows us to take a different approach than existing SIP algorithms, that are necessarily based on solving the nested optimization problems simultaneously. Consider the constraints $g_i(x)$ given by

$$g_i(x) = \max_{\zeta \in \hat{\mathcal{Z}}_i} \gamma_i(x, \zeta)\,,$$

where we now assume that all $\bar{\zeta} \in \hat{\mathcal{Z}}_i$ for which $g_i(x) = \gamma_i(x, \bar{\zeta})$, i.e. the maximizers $\bar{\zeta}$ of $\gamma_i(x, \zeta)$, are known. Then the constraints $g_i(x)$ can be regarded as 'ordinary' nonsmooth functions in x, that are non-differentiable at those points x where the global maximum is attained at multiple distinct values $\bar{\zeta}$. As a

result, if at each x the set of maximizers for the functions $\gamma_i(x,\cdot)$ are assumed to be known, then the SIP Problem (5.1) reduces to a *finite-dimensional* nonsmooth optimization problem. Note that, by assumption the gradients $\nabla_x\gamma_i(x,\zeta)$ for all $\zeta \in \hat{\mathcal{Z}}_i$ exist and are continuous in x.

Although less complex compared to SIP problems, (finite-dimensional) nonsmooth optimization problems are still hard to solve because the notion of gradient, and with that a clear and unique direction of descent, simply does not exist. Definitely, nonsmooth optimization problems are not new, and many novel algorithms have been proposed in the literature. They can roughly be categorized as *derivative-free* algorithms, and approaches that analytically gather some sort of information regarding the behavior of the function in the neighborhood of x at which it is non-differentiable. The latter methods are largely based on the *generalized directional derivative*, introduced by Clarke [24]. By *Danskin's Theorem* [25], it then follows that the generalized directional derivative for the type of nonsmooth functions such as the constraints $g_i(x)$, can be analytically determined if the gradients $\nabla_x\gamma_i(\cdot,\cdot)$ exist and are continuous.

Encouraged by Danskin's result, we follow the approaches based on the differential properties of nonsmooth functions, and subsequently extend existing (centralized) optimization algorithms to distributed algorithms that are applicable to the class of problems where the cooperative trajectory-generation framework belongs to. Hence, in the next section we give a brief overview of important results found in differential calculus of nonsmooth functions, followed by a succinct summary on nonsmooth optimization theory.

5.2 Nonsmooth Analysis

In this section, we will examine the differential properties of a nonconvex function $f : \mathbb{R}^n \to \mathbb{R}$ that is locally Lipschitz continuous and possibly non-differentiable at $x \in \mathbb{R}^n$. We present several notions and concepts that are required for the understanding and development of algorithms for nonsmooth optimization problems. The present section and the next are by no means meant to be a thorough and complete discussion, but merely a brief excerpt from references on this topic, such as [24,56,78]. For this reason, the reader is referred to these references for the proofs to the theorems given in the following.

5.2.1 Subdifferentials and Subgradients

The classical definition of a directional derivative of a function f at x in the direction $d \in \mathbb{R}^n$ is defined by

$$f'(x;d) = \lim_{\tau\to 0} \frac{f(x+\tau d) - f(x)}{\tau}. \tag{5.2}$$

Note that this is a two-sided directional derivative and if f is differentiable at x, we obtain

$$f'(x;d) = \langle \nabla f, d \rangle, \qquad \text{for all } d \in \mathbb{R}^n .$$

For a locally Lipschitz continuous function f, this classical directional derivative does not necessarily exist at some $x \in \mathbb{R}^n$. If the Lipschitz continuous function f is convex, then a *one-sided* directional derivative exists in every direction $d \in \mathbb{R}^n$.

Theorem 1 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function. Then the directional derivative $f'(x;d)$ exists in every direction $d \in \mathbb{R}^n$ and it satisfies*

$$f'(x;d) = \inf_{\tau > 0} \frac{f(x + \tau d) - f(x)}{\tau} . \tag{5.3}$$

The proof can be found in [56, 78] and is based on the convexity property of the function f. Related to the directional derivative are the concepts of the *subdifferential* and *subgradient* of a convex function f at x, which play important roles in nonsmooth convex optimization theories.

Definition 2 *The subdifferential of a convex function $f : \mathbb{R}^n \to \mathbb{R}$ at $x \in \mathbb{R}^n$ is the set*

$$\partial_c f(x) = \{\xi \in \mathbb{R}^n \mid f(y) \geq f(x) + \xi^\top (y - x) \quad \textit{for all} \quad y \in \mathbb{R}^n\} . \tag{5.4}$$

Each vector $\xi \in \partial_c f(x)$ is called a subgradient of f at x.

The next theorem gives the relation between the subdifferential $\partial_c f(x)$ and the directional derivative $f'(x;d)$, and summarizes the properties of the subdifferential. The proof of the theorem is given in [56].

Theorem 2 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be convex. Then at every x we have*

(i) $f'(x;d) = \max\{\xi^\top d \mid \xi \in \partial_c f(x)\}$ *for all* $d \in \mathbb{R}^n$,

(ii) $\partial_c f(x) = \{\xi \in \mathbb{R}^n \mid f'(x;d) \geq \xi^\top d\}$ *for all* $d \in \mathbb{R}^n$,

(iii) $\partial_c f(x)$ *is a nonempty, convex and compact set such that* $\partial_c f(x) \subset \mathcal{B}(0;K)$, *where K is the Lipschitz constant of f at x,*

(iv) *the point-to-set mapping* $\partial_c f(\cdot) : \mathbb{R}^n \to \mathcal{P}(\mathbb{R}^n)$ *is upper semicontinuous, i.e., if* $y_i \to x$ *and* $\xi_i \in \partial_c f(x)$ *for each i, then each accumulation point ξ of (ξ_i) is in $\partial_c f(x)$.*

Theorem 2 shows that, for convex functions, either the subdifferential or the directional derivative is sufficient to compute the other.

Nonsmooth optimization theories for nonconvex Lipschitz continuous functions generally find their origin in their convex counterparts. We will see in the next section, that nonsmooth convex optimization theories are based on the notion of the subgradient and subdifferential of a convex function f, as defined by Definition 2 and closely related to the directional derivative $f'(x;d)$. Unfortunately, the classical directional derivative, given by Equation (5.2), does not necessarily exist for a locally Lipschitz continuous function, with the exception of convex nonsmooth functions. The existence of the classical directional derivative is guaranteed, because the convexity property holds *globally* for a nonsmooth convex function. This is no longer true for a nonconvex Lipschitz continuous function, hence, we need an alternative definition of a directional derivative such that it holds only in the *local* neighborhood of x. For this reason, the (Clarke) *generalized directional derivative*, as proposed by Clarke in [24], is widely used in the field of nonsmooth optimization of nonconvex functions. The generalized directional derivative is defined as follows.

Definition 3 ***(Clarke)***. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz continuous function at $x \in \mathbb{R}^n$. The generalized directional derivative of f at x in the direction of $d \in \mathbb{R}^n$ is defined by*

$$f^\circ(x;d) = \limsup_{\substack{y \to x \\ \tau \downarrow 0}} \frac{f(y+\tau\, d) - f(y)}{\tau}\,. \tag{5.5}$$

In [24] it is shown that for a function f that is locally Lipschitz at x with constant K, the generalized directional derivative satisfies

$$|f^\circ(x;d)| \leq K\,\|d\|\,.$$

This means that the generalized directional derivative is well-defined and finite for locally Lipschitz continuous functions, and, analogous to Theorem 2, allows for a generalization of the subdifferential $\partial f(x)$ at x for nonconvex locally Lipschitz continuous functions. The following definition of the subdifferential is given in [24], where it is termed as *generalized gradient*.

Definition 4 ***(Clarke)***. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz continuous function at $x \in \mathbb{R}^n$. Then the subdifferential of f at x is the set $\partial f(x)$ of vectors $\xi \in \mathbb{R}^n$ such that*

$$\partial f(x) = \{\xi \in \mathbb{R}^n \,|\, f^\circ(x;d) \geq \xi^\top d \quad \textit{for all} \quad d \in \mathbb{R}^n\}\,. \tag{5.6}$$

Each vector $\xi \in \partial f(x)$ is called a subgradient of f at x.

The reason why $\partial f(x)$ is called the generalized gradient in [24], becomes clear if f is smooth, i.e. continuously differentiable. In that case, the subdifferential $\partial f(x)$ reduces to the singleton set $\{\nabla f(x)\}$. Moreover, the

subdifferential for Lipschitz continuous functions is in fact a generalization of the subdifferential for convex functions, as shown by the following theorem [56].

Theorem 3 *If the function $f : \mathbb{R}^n \to \mathbb{R}$ is convex, then*

(i) $f'(x;d) = f^\circ(x;d)$ *for all* $d \in \mathbb{R}^n$, *and*

(ii) $\partial_c f(x) = \partial f(x)$.

The next theorem presents some properties of the subdifferential for nonconvex Lipschitz continuous functions. Again, the proof of the theorem can be found in [56].

Theorem 4 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz continuous function at $x \in \mathbb{R}^n$ with a Lipschitz constant K. Then*

(i) $\partial f(x)$ *is a nonempty, convex and compact set such that* $\partial f(x) \subset \mathcal{B}(0;K)$,

(ii) $f^\circ(x;d) = \max\{\xi^\top d \,|\, \xi \in \partial f(x)\}$ *for all* $d \in \mathbb{R}^n$,

(iii) *the point-to-set mapping* $\partial f(\cdot) : \mathbb{R}^n \to \mathcal{P}(\mathbb{R}^n)$ *is upper semicontinuous.*

A Lipschitz function is differentiable almost everywhere by Rademacher's Theorem and, hence, the gradient exists almost everywhere. Let the set Ω_f denote the set of points where the function f is not differentiable. Then the following theorem allows us to compute the subdifferential $\partial f(x)$ of a locally Lipschitz continuous function f at $x \in \mathbb{R}^n$. The proof is presented in [56].

Theorem 5 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz continuous function at $x \in \mathbb{R}^n$. Then*

$$\partial f(x) = \operatorname{conv}\{\xi \in \mathbb{R}^n \,|\, \textit{there exists } (x_i) \subset \mathbb{R}^n \backslash \Omega_f \textit{ such that } x_i \to x \textit{ and } \nabla f(x_i) \to \xi\}.$$

Figure 5.1 presents an example of a subdifferential of a locally Lipschitz continuous function $f(x)$. The figure shows the level set $f(x) = c$ and the local minimizer is indicated with x^*. At the point x_1, the function is differentiable and, therefore, the subdifferential $\partial f(x_1)$ consists of the single element $\nabla f(x_1)$. At x_2 the function is non-differentiable and we can use Theorem 5 to find the two subgradients ξ_1 and ξ_2, and obtain the subdifferential $\partial f(x_2) = \operatorname{conv}\{\xi_1, \xi_2\}$. Definitely, Theorem 5 is rather cumbersome to compute the subdifferential. Rules have been derived to compute the subdifferential of basic standard functions and we refer, for example, to [24] for a more in-depth overview. Nevertheless, in the following we will extensively use the maximum function and, hence, we present a theorem that states how to compute the subdifferential of such a function. First we need the definition of a *regular* function.

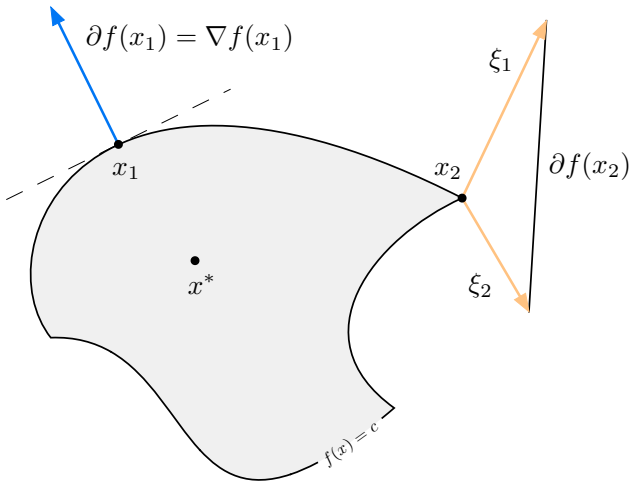


Figure 5.1: Example of a subdifferential $\partial f(x)$. The level set c for a nonconvex function $f : \mathbb{R}^2 \to \mathbb{R}$ is drawn here. The local minimizer is indicated with x^*. The function f is continuously differentiable at x_1, while the gradient fails to exist at x_2. Hence, the subdifferential of f at x_2 is given by: $\partial f(x_2) = \text{conv}\{\xi_1, \xi_2\}$.

Definition 5 *The function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be regular at $x \in \mathbb{R}^n$ if for all $d \in \mathbb{R}^n$ the directional derivative $f'(x; d)$ exists, and*

$$f'(x; d) = f^\circ(x; d)\,.$$

Theorem 6 *Let $f_i : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz continuous function at $x \in \mathbb{R}^n$ for each $i = 1, \ldots, m$. Then the function $f : \mathbb{R}^n \to \mathbb{R}$ defined by*

$$f(x) := \max_{i=1,\ldots,m} f_i(x)$$

is locally Lipschitz at x and

$$\partial f(x) \subset \operatorname*{conv}_{i \in I(x)} \partial f_i(x)\,,$$

where $I(x) = \{i \in \{1, \ldots, m\} \,|\, f_i(x) = f(x)\}$. In addition, if f_i is regular at x for each $i = 1, \ldots, m$, then f is regular at x and

$$\partial f(x) = \operatorname*{conv}_{i \in I(x)} \partial f_i(x)\,.$$

The proof of this theorem can be found in [56].

5.2.2 Continuous Maximum Functions

Let us recall that nonsmooth constraints $g_i(x)$ of the cooperative trajectory-generation framework are locally Lipschitz continuous functions of the form

$$g_i(x) = \max_{\zeta \in \bar{\mathcal{Z}}_i} \gamma_i(x, \zeta)\,,$$

where the functions $\gamma_i : \mathbb{R}^n \times \mathbb{R}^{l_i} \to \mathbb{R}$ are continuous and their gradients $\nabla_x \gamma_i(\cdot, \cdot)$ exist and are continuous. For these type of functions, Danskin's Theorem, proven in [25], states that the directional derivative in the classical sense exists in every direction $d \in \mathbb{R}^n$.

Theorem 7 ***(Danskin)***. *Let $F : \mathbb{R}^n \times \hat{Y} \to \mathbb{R}$, where $\hat{Y}$ is a compact topological space, be continuously differentiable with respect to the first variable. Also, let the function $\phi(x)$ and point-to-set map $Y(x)$ be given as*

$$\phi(x) = \max_{y \in \hat{Y}} F(x, y)\,,$$

and

$$Y(x) = \{y \in \hat{Y} \,|\, F(x, y) = \phi(x)\}\,,$$

respectively, where $Y(x)$ is the set of maximizers of $F(x, y)$ over y. Then the function $\phi(x)$ has, for every x and d in $\mathbb{R}^n$, a directional derivative at x in the direction of d given by

$$\phi'(x; d) = \max_{y \in Y(x)} \langle\, d, \nabla_x F(x, y) \,\rangle\,.$$

Theorem 2.1 in [23] generalizes the results for locally Lipschitz functions $F(x, y)$ by using the definition of the generalized directional derivative and, additionally, gives the subdifferential of the function ϕ at x. In the case when $F(x, y)$ is continuously differentiable, we obtain the following corollary to Theorem 2.1 in [23].

Corollary 1 *Let the functions $\phi(x)$ and $F(x, y)$ be given as in Theorem 7. Then*

(i) *$\phi(x)$ is locally Lipschitz continuous,*

(ii) *The generalized directional derivative of ϕ at x in the direction of $d \in \mathbb{R}^n$ is given by*

$$\phi^\circ(x; d) = \max\{\xi^\top d \,|\, \xi = \nabla_x F(x, y)\,, y \in Y(x)\}\,, \qquad \text{for all } d \in \mathbb{R}^n\,,$$

where $Y(x) = \{y \in \hat{Y} \,|\, F(x, y) = \phi(x)\}$,

(iii) *$\phi'(x;d)$ exists and $\phi'(x;d) = \phi^\circ(x;d)$,*

(iv) *The subdifferential of ϕ at x is found as*

$$\partial\phi(x) = \operatorname*{conv}_{y\in Y(x)} \{\nabla_x F(x,y)\}\,.$$

Note that Danskin's Theorem is recovered and that the subdifferential of ϕ at x can be easily computed by Corollary 1(iv).

We conclude this section by emphasizing the importance of Corollary 1 in the context of the cooperative trajectory-generation framework as formulated in Chapter 4. It is shown that with the developed algorithms for Bézier curves, we can easily determine the set of maximizers $\mathcal{Z}_i(x)$ for the functions $\gamma_i(x,\zeta)$, defined as

$$\mathcal{Z}_i(x) = \{\zeta \in \hat{\mathcal{Z}}_i \,|\, \gamma_i(x,\zeta) = g_i(x)\}\,.$$

Basically, these algorithms, in combination with Corollary 1, allow us to *efficiently* compute the *whole* subdifferential of the nonsmooth constraint functions $g_i(x)$. In general, determining the entire subdifferential of a locally Lipschitz continuous function is hard, if not impossible, and existing optimization algorithms, such as the bundle method, necessarily relax this requirement at the expense of a more complex algorithm. In fact, the extra steps that are needed to overcome the lack of knowledge over the exact subdifferential, prevent a distributed formulation of these algorithms. We shall see that by incorporating the knowledge of the whole subdifferential, we are able to formulate a distributed algorithm to solve the cooperative trajectory-generation problem.

5.3 Nonsmooth Optimization Theory

The optimization theory of nonsmooth functions is based on the differential properties that were discussed in the previous section. The subdifferential $\partial f(x)$ of a locally Lipschitz continuous function f plays an important role in the first-order optimality conditions, while the subgradient $\xi \in \partial f(x)$ allows us to construct a linear approximation of f at x. In fact, the notions of necessary conditions and linear approximations given hereafter, are analogous to those defined for smooth optimization problems and, hence, they are generalizations of their well-known counterparts in classical optimization theory. First, we derive results for the unconstrained nonsmooth optimization problem and, subsequently, discuss the *bundle method* which is a popular method for solving nonsmooth optimization problems. The theoretical results for the constrained case will be presented in the next chapter, where we develop a distributed algorithm to solve our constrained

nonsmooth optimization problem.

5.3.1 Unconstrained Optimization

Consider an unconstrained nonsmooth optimization problem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad f(x)\,,$$

where the objective function $f:\mathbb{R}^n\to\mathbb{R}$ is locally Lipschitz continuous at x for all $x\in\mathbb{R}^n$. First of all, let us formally give the definition of minimizers for the above unconstrained optimization problem. Note that the definitions apply to smooth as well as nonsmooth objective functions f.

Definition 6 *A point $x^*\in\mathbb{R}^n$ is a* global minimum *of f if it satisfies*

$$f(x^*)\leq f(x)\,, \qquad \textit{for all } x\in\mathbb{R}^n\,.$$

Definition 7 *A point $x^*\in\mathbb{R}^n$ is a* local minimum *of f if there exists $\epsilon>0$ such that*

$$f(x^*)\leq f(x)\,, \qquad \textit{for all } x\in\mathcal{B}(x^*;\epsilon)\,.$$

The next theorem gives the necessary conditions for an unconstrained optimization problem to attain a local minimum at x^*. The proof, given in [56,78], follows from the definitions of the subdifferential ∂f and a local minimum for a locally Lipschitz continuous function f at x^* (Definitions 4 and 7), and Theorem 4.

Theorem 8 *Let $f:\mathbb{R}^n\to\mathbb{R}$ be a locally Lipschitz continuous function at $x\in\mathbb{R}^n$. If f attains its local minimum at x^*, then*

(i) $0\in\partial f(x^*)$, *and*

(ii) $f^\circ(x^*;d)\geq 0$ *for all* $d\in\mathbb{R}^n$.

It is clear that Theorem 8 is indeed a generalization of the results known from the classical optimization theory, where the necessary conditions for a local minimizer of a nonconvex function f at x^* is given by $\nabla f(x^*)=0$. If the function f is convex, the conditions are sufficient and f attains a global minimum at x^*, as shown by the following theorem [56].

Theorem 9 *If $f:\mathbb{R}^n\to\mathbb{R}$ is a convex function, then the following conditions are equivalent:*

(i) *f attains its global minimum at x^*,*

(ii) $0 \in \partial_c f(x^*)$, *and*

(iii) $f'(x^*; d) \geq 0$ *for all* $d \in \mathbb{R}^n$.

Many classical optimization algorithms are based on local approximations of the function f at x. For example, the steepest descent (gradient) method is derived from the first-order Taylor expansion, while Newton's method is a refinement by using second-order information. For a nonsmooth function, these approximations cannot be defined from the Taylor expansion, simply because the gradient may not exist at x. Moreover, we have seen that the function behaves differently in the neighborhood of x, depending on the direction $d \in \mathbb{R}^n$ in which we are moving away from x. The following definitions capture these difficulties and define a piecewise linear local approximation of a locally Lipschitz continuous function at x.

Definition 8 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz continuous function at $y \in \mathbb{R}^n$ and let $\xi \in \partial f(y)$ be an aribitrary subgradient. Then the ξ-linearization of f at y is the function $\tilde{f} : \mathbb{R}^n \to \mathbb{R}$ defined by*

$$\tilde{f}(x) = f(y) + \langle\, \xi, x - y \,\rangle\,, \qquad \text{for all } x \in \mathbb{R}^n\,, \tag{5.7}$$

and the linearization of f at y is the function $\bar{f} : \mathbb{R}^n \to \mathbb{R}$ such that

$$\bar{f}(x) = \max_{\xi \in \partial f(y)} \tilde{f}(x)\,, \qquad \text{for all } x \in \mathbb{R}^n\,. \tag{5.8}$$

Optimization methods are fundamentally based on the principle of moving from a point x, along a direction $d \in \mathbb{R}^n$, such that the function value decreases at $x + \tau\, d$ for some $\tau > 0$. These are so-called *descent methods*, i.e for the unconstrained optimization problem they generate a sequence of points $\{x_k\}$ such that

$$x_k \in \mathbb{R}^n \qquad \text{and} \qquad f(x_{k+1}) < f(x_k) \qquad \text{for all } \; k = 1, 2, \ldots$$

In order to define how x_k is updated, we need to first give the notion of a *descent direction* d.

Definition 9 *The direction $d \in \mathbb{R}^n$ is called a descent direction for $f : \mathbb{R}^n \to \mathbb{R}$ at x_k, if there exists $\epsilon > 0$ such that*

$$f(x_k + \tau d) < f(x_k)\,, \qquad \text{for all } \; \tau \in (0, \epsilon\,]\,.$$

Hence, it is a natural choice to update the current iteration point x_k according to

$$x_{k+1} = x_k + \tau d\,.$$

Hence, it remains to determine the direction d along which the function f decreases. The next theorem states conditions for which a direction d is a descent direction for a locally Lipschitz continuous function $f(x)$ and, more importantly, that a descent direction d for the linearization $\bar{f}(x)$, as given by Definition 8, is in fact a descent direction for the function f at x_k. This latter result allows us to formulate the descent direction finding problem in terms of the linearization $\bar{f}(x)$ at x_k.

Theorem 10 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be locally Lipschitz at x. The direction $d \in \mathbb{R}^n$ is a descent direction for f at x if any of the following hold:*

(i) $f^\circ(x; d) < 0$,

(ii) $\xi^\top d < 0$ *for all* $\xi \in \partial f(x)$, *and*

(iii) d *is a descent direction for* $\bar{f}$ *at* x.

Proof: The proof is given in Appendix B.3. □

The following theorem tells us how to find a descent direction d for the linearization of locally Lipschitz continuous function f at x. From Theorem 10(iii) we know that this is also a descent direction for the locally Lipschitz continuous function f.

Theorem 11 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz continuous function at $x \in \mathbb{R}^n$ and let $\xi^* \in \partial f(x)$ exist such that $\xi^* = \arg\min\{\|\xi\| \,|\, \xi \in \partial f(x)\}$. Consider the problem*

$$\underset{d \in \mathbb{R}^n}{\text{minimize}} \quad \bar{f}(x + d) + \frac{1}{2}\|d\|^2 . \tag{5.9}$$

Then

(i) *Problem (5.9) has a unique solution $d^* \in \mathbb{R}^n$ such that $d^* = -\xi^*$,*

(ii) $f^\circ(x; d^*) = -\|d^*\|^2$,

(iii) $\bar{f}(x + \tau\, d^*) = \bar{f}(x) - \tau\, \|\xi^*\|^2$ *for all* $\tau \in [0, 1]$,

(iv) $0 \notin \partial f(x)$ *if and only if* $d^* \neq 0$, *and*

(v) $0 \in \partial f(x)$ *if and only if* $\bar{f}$ *attains its global minimum at* x.

The term $\frac{1}{2}\|d\|^2$ is added to the objective function, in order to convexify the problem and guarantee a unique solution for Problem (5.9). Theorem 11 shows that a descent direction d of $\bar{f}$ (and thus of f) at x, is in fact the (negative) subgradient $\xi \in \partial f(x)$ with the minimum norm. Therefore, if one has knowledge over the whole subdifferential ∂f at x, one could find a feasible descent direction d by minimizing the norm over all subgradients ξ contained in the subdifferential. Practically, there are two main issues with this approach:

1. In general, one does not know the whole subdifferential of a nonsmooth function and, hence, Theorem 11 demands a strict requirement to satisfy. Instead, existing methods for solving nonsmooth optimization problems require the knowledge of only one arbitrary subgradient ξ of the subdifferential ∂f at x.

2. A natural attempt is to extend gradient-based methods, commonly applied in smooth optimization theories, to nonsmooth problems by employing the descent direction found through Theorem 11. Unfortunately, counter examples in [102, 103] show that an algorithm based on this concept may take infinitely many steps without significantly decreasing the cost function and, hence, converge to a non-stationary point. The reason for this phenomenon is due to the fact that the descent direction d as function of x, generated by such an algorithm, is discontinuous in x [11, 44, 45]. For example, the absolute function $f(x) = |x|$ is not differentiable at $x = 0$ (Figure 5.2). For all $x > 0$ the subdifferential $\partial_c f(x) = \{1\}$ and for all $x < 0$ the subdifferential $\partial_c f(x) = \{-1\}$. At $x = 0$, the subdifferential is given by:

$$\partial_c f(0) = \operatorname{conv}\{-1, 1\}.$$

   Note that $x^* = 0$ is the global minimizer for $f(x)$ by Theorem 9, since $0 \in \operatorname{conv}\{-1, 1\}$. Clearly, a sequence of descent directions $\{d_k\}$ generated by Theorem 11 is discontinuous at $x = 0$.

   To tackle this problem, it is necessary to build a certain degree of "foresight" in the subdifferential, so that it does not change abruptly in the neighborhood of x^*. This can be achieved by considering the ϵ-subdifferential $\partial_\epsilon f(x)$ instead of the subdifferential $\partial f(x)$. A definition of the ϵ-subdifferential can be found, for example, in [11]. Nevertheless, it is still too big of a demand, or even bigger, to require the knowledge of the entire ϵ-subdifferential of a non-differentiable function. Hence, one approach taken in nonsmooth optimization algorithms, is to approximate this ϵ-subdifferential by collecting a *bundle* of subgradients that are computed at each iteration point, the so-called ϵ-steepest descent method [51]. Another variation of the bundle method was introduced in [48]. The approach is based on the classical cutting plane method, where instead of approximating the ϵ-subdifferential, a convex piecewise linear approximation of the nonsmooth function is constructed from the linearizations generated with the subgradients.

Nevertheless, Theorem 11 forms the basis of the many nonsmooth optimization methods. The methods mainly differ in the approach the direction finding problem is implemented.

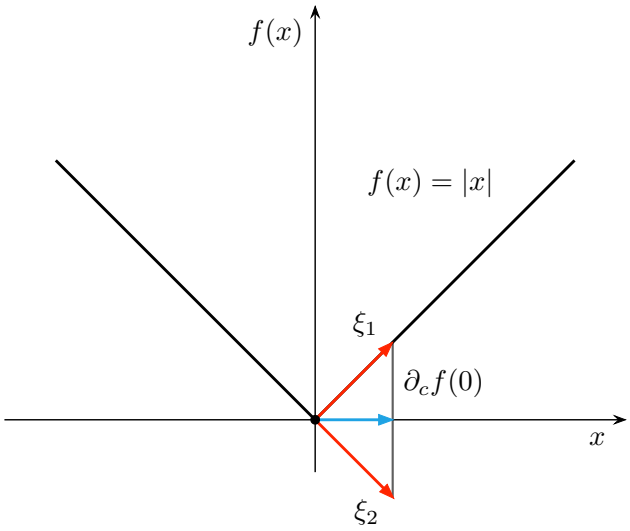


Figure 5.2: Subdifferential of the absolute function $f(x) = |x|$ at $x = 0$.

5.3.2 Bundle Methods

Nonsmooth optimization methods that rely on differential information of the functions, can be divided into subgradient methods and bundle methods. Both classes of methods require that, at each iteration point, only the objective function value and *one* subgradient are available. Definitely, these methods are preferred for problems for which one cannot (easily) determine the whole subdifferential of the nonsmooth functions, but instead only one subgradient. For example, finite-difference approximations can be used to determine one subgradient. Subgradient methods, first introduced in [88], are in essence generalizations of gradient-based methods for smooth optimization problems, where an arbitrary subgradient replaces the gradient. Convergence is only guaranteed for convex optimization problems and, in combination with its simplicity, the subgradient method is widely used for these types of problems. Nevertheless, the subgradient method suffers from poor convergence rates and, unlike in gradient-based methods, the direction opposite an arbitrary subgradient may not necessarily be a descent direction.

Bundle methods, on the other hand, are developed for convex as well as nonconvex nonsmooth optimization problems. Additionally, as we shall see in the next chapter, the method lends itself to a distributed formulation. Therefore, in the remainder of this section, we give a brief background on the general bundle method by considering a convex unconstrained nonsmooth optimization problem. The intention is to present a high-level overview of the methods, without cluttering the big picture with too many details. In the next chapter, we will further discuss the technical details concerning constrained and nonconvex nonsmooth optimization problems in conjunction with the development of the distributed algorithms.

The bundle method finds its origin in the ϵ-steepest descent method that was introduced in [51]. Many

variants of the bundle method have been developed ever since. While some are refinements to the original idea as introduced in [51], others are based on a different approach in applying the common guiding principle of bundle methods: exploiting the subgradient information from previous iterations by gathering them into a *bundle*. The common objective is to obtain a good approximation of the nonsmooth function, especially *in the neighborhood* of its local minimum, despite only having limited information on the behavior of the function at each iteration point. To better understand the idea behind bundle methods, let us consider the following unconstrained nonsmooth optimization problem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad f(x)\,,$$

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is convex. Bundle methods assume that at every point $x \in \mathbb{R}^n$, the function value $f(x)$ and an arbitrary subgradient ξ from the subdifferential $\partial_c f(x)$ can be computed. The definitions of the subdifferential and subgradient are given in Definition 2. The necessary and sufficient optimality condition is given in Theorem 9, that is, $x^* \in \mathbb{R}^n$ is a global minimizer for the function f, if and only if

$$0 \in \partial_c f(x^*)\,.$$

It is clear that, in order to verify the necessary and sufficient condition, we need to have knowledge over the subdifferential $\partial_c f$ at a point x. However, since at a point x we only have one arbitrary subgradient of the subdifferential, this condition cannot be checked directly. Bundle methods aim to construct an approximation of the subdifferential $\partial_c f$ at the minimizer x^* from the bundle of subgradients gathered from previous iterations. The various bundle methods differ in the way they compute the descent direction d at each iteration point in order to converge to x^*. The ϵ-steepest descent method [51] computes this descent direction using an approximation of the ϵ-subdifferential directly and let the parameter $\epsilon \to 0$, so that when x converges to x^*, the ϵ-subdifferential converges to the subdifferential of f at x^*. Other bundle methods obtain the descent directions from an approximation of the nonsmooth function, based on the cutting plane model and, therefore, avoid the complexity of choosing an appropriate ϵ at each iteration. Intuitively, when the iteration points converge to the global minimizer x^*, the resulting cutting plane model of the nonsmooth function exhibits a close approximation of the ϵ-subdifferential at x^*. Therefore, despite the different approaches, there is a clear connection between both methods [56].

The distributed algorithms that will be presented in the next chapter, borrow the ideas from the bundle methods that are based on the cutting plane approach. Hence, we will focus our following discussion on this branch of bundle methods. Suppose that, in addition to the current iteration point x_k, we have a collection

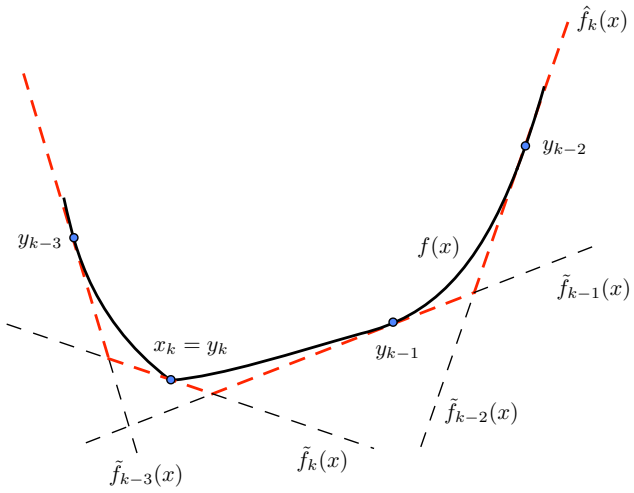


Figure 5.3: Cutting-plane model of a convex function $f : \mathbb{R} \to \mathbb{R}$. The black solid line is the function $f(x)$, the black-dashed lines are the ξ_j-linearizations $\tilde{f}_j(x)$ at y_j with $j \in J_k$, and the red-dashed line is the piecewise linear approximation $\hat{f}_k(x)$ at the kth iteration.

of previous trial points $y_j \in \mathbb{R}^n$ (from previous iterations) and corresponding subgradients $\xi_j \in \partial_c f(y_j)$ for $j \in J_k$, where the index set J_k is a nonempty subset of $\{1, \ldots, k\}$. Then, at the kth iteration, we construct a piecewise linear approximation $\hat{f}_k$ for all $x \in \mathbb{R}^n$ as

$$\hat{f}_k(x) = \max_{j \in J_k} \tilde{f}_j(x)\,,$$

where $\tilde{f}_j$ is the ξ_j-linearization of f at y_j as given in Equation (5.7)

$$\tilde{f}_j(x) = f(y_j) + \langle\, \xi_j, x - y_j \,\rangle\,.$$

The piecewise linear approximation $\hat{f}_k(x)$ approximates the function from below due to the convexity of f, i.e. $\hat{f}_k(x)$ is a cutting-plane model of $f(x)$ at the kth iteration. Note that we can rewrite $\tilde{f}_j$ as follows

$$\begin{aligned}\tilde{f}_j(x) &= f(y_j) + \langle\, \xi_j, x - y_j \,\rangle \pm f(x_k) \pm \langle\, \xi_j, y_j - x_k \,\rangle\,,\\ &= f(x_k) + \langle\, \xi_j, x - x_k \,\rangle - \alpha_j^k\,,\end{aligned}$$

where $\alpha_j^k := f(x_k) - f(y_j) - \langle\, \xi_j, x_k - y_j \,\rangle$ and represents the *linearization error* at x_k of the ξ_j-linearization obtained at a previous iteration point y_j. Since $f(x)$ is convex, we have that the linearization error $\alpha_j^k \geq 0$ for all $j \in J_k$. Figure 5.3 shows an example of a cutting-plane model of a convex function $f : \mathbb{R} \to \mathbb{R}$ at the kth iteration, consisting of four trial points y_j, where $j \in \{k-3, \ldots, k\}$. The function is non-differentiable at y_k and, hence, the ξ_k-linearization is constructed using an arbitrary subgradient $\xi_k \in \partial_c f(y_k)$. As a result,

the linearization $\tilde{f}_k(x)$ poorly approximates the function $f(x)$ at y_k. Lastly, it can be observed that the piecewise linear approximation $\hat{f}_k(x)$ approximates the function $f(x)$ from below.

Next, a descent direction $d_k \in \mathbb{R}^n$ for $\hat{f}_k$ at x_k is computed by solving the following direction finding problem

$$\underset{d_k \in \mathbb{R}^n}{\text{minimize}} \quad \hat{f}(x_k + d_k) + \frac{1}{2} d_k^\top M_k d_k \,,$$

where again a stabilizing term $1/2\, d_k^\top M_k d_k$ is added to guarantee the existence of a unique solution d_k and to keep the approximation local. Different approaches exist for the choice of M_k, resulting in various forms of bundle methods. The direction finding problem will be discussed in more detail in the next chapter. At this point, suppose we have determined a descent direction d_k for the piecewise linear approximation $\hat{f}_k(x)$ at the iteration point x_k. We might tend to simply set the next iteration point x_{k+1} as

$$x_{k+1} = x_k + \tau\, d_k \,, \tag{5.10}$$

for a sufficiently small step size $\tau > 0$. Unfortunately, this may result in a non-converging sequence $\{x_k\}$, as a descent direction for $\hat{f}_k(x)$ might not necessarily be a descent direction for the convex objective function $f(x)$ at x_k. Recall that at each iteration point, only one arbitrary subgradient is used for the construction of the cutting plane model $\hat{f}_k(x)$ and, hence, at a certain iteration k this might be a very poor approximation of the function $f(x)$. For example, similar to the subgradient method, the direction opposite to any arbitrary subgradient $\xi_j \in \partial_c f(y_j)$ might even fail to be a descent direction. Therefore, although bundle methods do not require the knowledge of the whole subdifferential, it comes at the expense of a more complicated line search procedure, in order to guarantee global convergence.

The line search procedure is designed such that a next iteration point x_{k+1} is accepted if the function value $f(x_{k+1})$ decreases sufficiently. We give a simplified version of this special line search procedure, applicable to convex functions f. First, a trial point y_{k+1} is computed using the descent direction d_k found for $\hat{f}_k$, according to

$$y_{k+1} = x_k + d_k \,.$$

Then the procedure proceeds by checking whether

$$f(y_{k+1}) \leq f(x_k) + m_L \nu_k$$

is satisfied. The parameter $m_L \in (0, \frac{1}{2})$ is the line search parameter, while $\nu_k = \hat{f}(y_{k+1}) - f(x_k)$ represents the *predicted descent* of f at x_k. Hence, the parameter m_L specifies a minimum amount of desired descent.

If the desired descent is met, the line search procedure takes a *serious step* by setting

$$x_{k+1} := y_{k+1}\,,$$

otherwise a so-called *null step* is taken, that is

$$x_{k+1} := x_k\,.$$

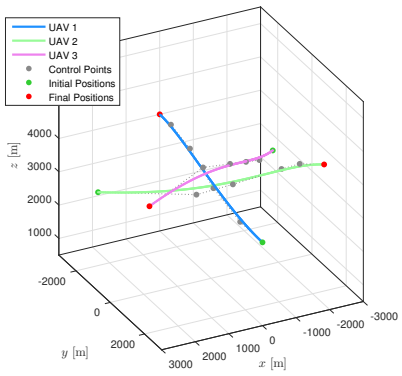
In both cases, the subgradient $\xi_{y_{k+1}} \in \partial_c f(y_{k+1})$ is added to the bundle to improve the cutting plane model. Effectively, if a null step is taken, a new descent direction d for $\hat{f}$ at x_k is determined using an improved cutting plane model, obtained due to the additional trial point y_{k+1}. The algorithm is terminated if

$$\nu_k \geq -\epsilon_s\,,$$

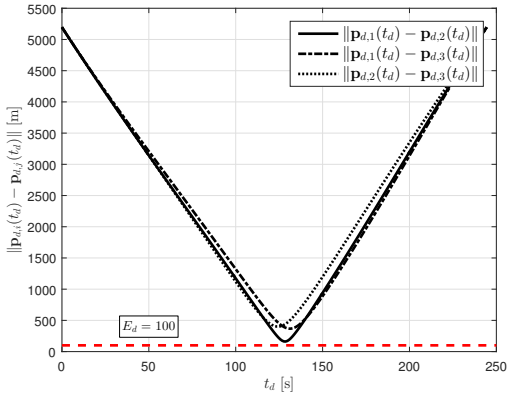
where $\epsilon_s > 0$ denotes the desired final accuracy tolerance. For a nonconvex locally Lipschitz continuous function f, the line search procedure becomes more complicated and involves an iteration to determine the step sizes for updating y_{k+1} and x_{k+1}. Especially this line search procedure makes it hard to formulate a distributed bundle method suitable for a team of vehicles. Intuitively, the line search procedure will require the vehicles to exchange their objective function values and, secondly, will most likely result in a distributed line search procedure at each iteration k. A more comprehensive overview of the bundle method algorithm is given in [48, 56].

To end this chapter, Figures 5.4 and 5.5 show the results for Case III of the fixed-wing UAVs simulation example from Section 4.5.1, obtained by using a bundle method solver instead of the standard Matlab® optimization routines. The bundle method solver is a Fortran implementation of the Multiobjective Proximal Bundle Methods (MPBNGC 2.0)[1]. In order to integrate this Fortran code with the existing Matlab® cooperative trajectory-generation routines, the main routine had to be rewritten in Matlab® , while wrapper functions were created for the remaining Fortran subroutines. The results were found to be encouraging and, in the next chapter, we present a distributed algorithm that is suitable for our nonsmooth optimization problem, by combining a modified version of the bundle method with existing work in the area of distributed smooth optimization problems. We also show that we can avoid the line search procedure as, by using the whole subdifferential at x_k, it can be proven that the descent direction for the cutting plane model is also a descent direction for the function f.

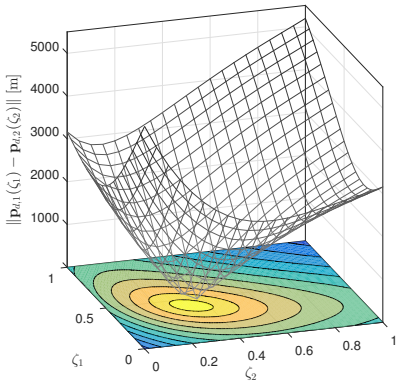
[1]The Fortran code was kindly provided by Prof. Marko M. Mäkelä of the University of Turku, Finland.



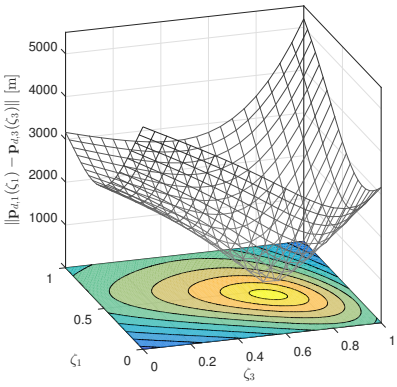
(a) Three-dimensional flight paths



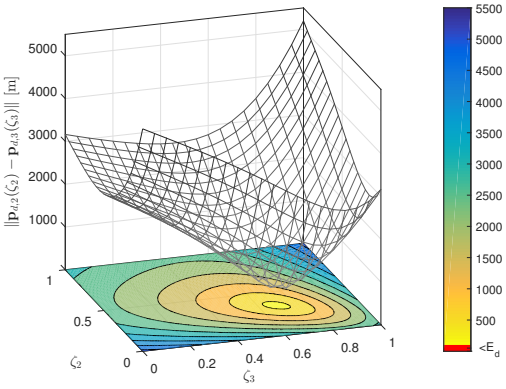
(b) Vehicle separation



(c) Path separation UAV 1 and 2

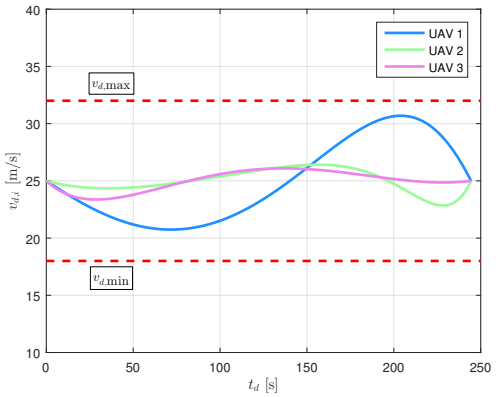


(d) Path separation UAV 1 and 3

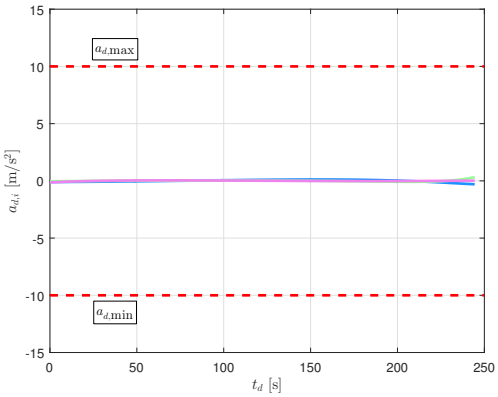


(e) Path separation UAV 2 and 3

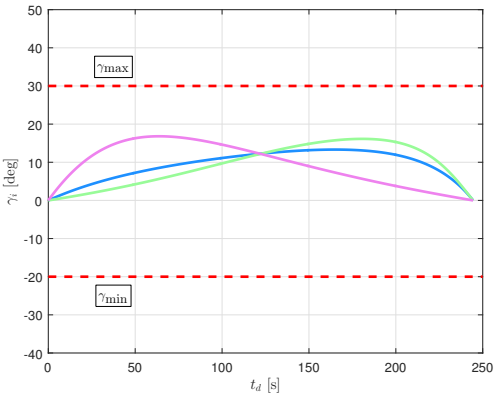
Figure 5.4: Three-dimensional trajectories for a team of three cooperating UAVs. Case III: spatial deconfliction is ensured through temporal separation. A bundle method solver is used to obtain the solution.



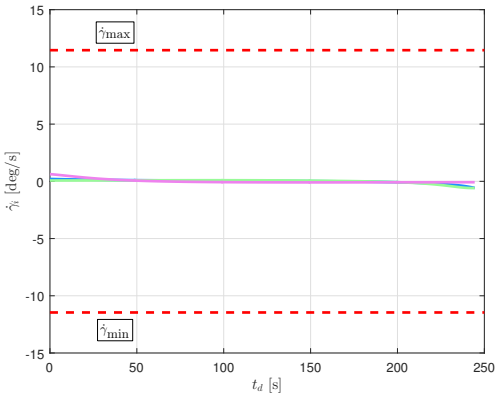
(a) Speed profile $v_{d,i}(t_d)$



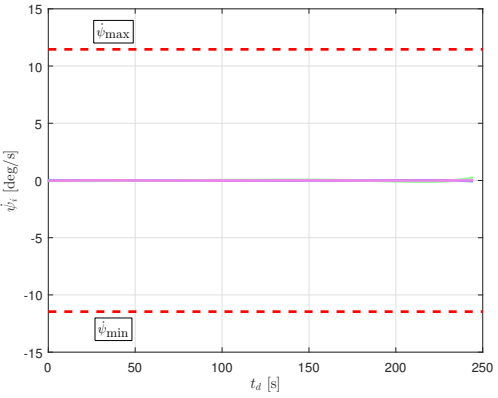
(b) Acceleration profile $a_{d,i}(t_d)$



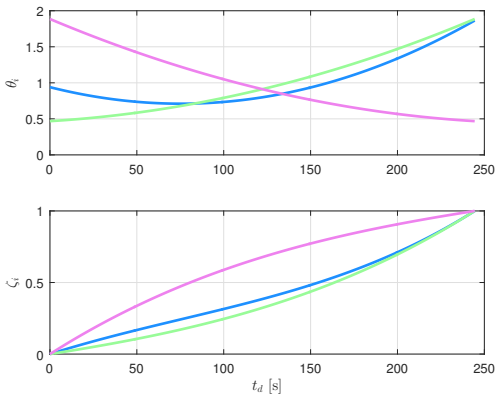
(c) Flight-path angle $\gamma_i(t_d)$



(d) Rate of change of flight-path angle $\dot{\gamma}_i(t_d)$



(e) Turn rate $\dot{\psi}_i(t_d)$



(f) Timing law θ_i and the function $\zeta_i(t_d)$

Figure 5.5: Dynamic constraints and timing laws for a team of three cooperating UAVs. Case III: spatial deconfliction is ensured through temporal separation. A bundle method solver is used to obtain the solution.

Chapter 6

Distributed Nonsmooth Optimization

Chapter 5 presented a short overview of important notions of nonsmooth analysis, and optimality conditions for unconstrained nonsmooth optimization problems. A popular method for solving nonsmooth optimization problems in a centralized way, namely the bundle method, was also discussed briefly. In this chapter, we develop a distributed algorithm for solving nonsmooth optimization problems such as the cooperative trajectory-generation framework from Chapter 4. In Section 6.1, we discuss a distributed nonlinear programming method that is applicable to a broad class of smooth optimization problems. Subsequently, in Section 6.2, a distributed bundle method is derived by combining the distributed nonlinear programming method with the (centralized) bundle method for nonsmooth optimization problems.

6.1 Distributed Optimization

In this section, we summarize the distributed nonlinear programming methods that are presented in [57–62]. Results on equality constrained optimization problems are presented in [58, 59, 62], whereas [60, 61] consider optimization problems with inequality constraints. The approach is based on classical centralized algorithms that result in a distributed formulation of the algorithm. This renders the approach versatile and applicable to a broad class of optimization problems, including our nonsmooth trajectory-generation framework. To illustrate the approach, we consider an equality constrained optimization problem. A more detailed discussion can be found in [58, 59, 62], including the proofs that are omitted in this section.

6.1.1 Problem Formulation

Consider a team of N vehicles solving the following equality constrained problem

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & h(x) = 0\,. \end{aligned} \tag{6.1}$$

The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is defined as

$$f(x) := \sum_{i=1}^{N} f_i(x)\,,$$

while the vector-valued function $h : \mathbb{R}^n \to \mathbb{R}^N$ is given by

$$h(x) := [h_1(x), \ldots, h_N(x)]^\top\,,$$

where $h_i : \mathbb{R}^n \to \mathbb{R}$ and $N \leq n$. The following assumptions are made on the functions $f(x)$ and $h(x)$.

Assumption 1 *It is assumed that*

(i) *the functions $f_i(x)$ and $h_i(x)$ for all $i = 1, \ldots, N$ are twice continuously differentiable, and*

(ii) *vehicle i has knowledge of only functions $f_i(x)$ and $h_i(x)$.*

The communication topology over which the set of N vehicles exchange information, is modeled as an undirected communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, N\}$ is the set of nodes and $\mathcal{E} = \{e_{ij}\}$ is the set of edges. An edge between two nodes i and j means that vehicles i and j can exchange information. The set of neighbors of agent i is defined as $\mathcal{N}_i := \{\, j \,|\, e_{ij} \in \mathcal{E}\}$, and we assume that at each time instant k the vehicles can synchronously exchange information with their neighbors. The (weighted) Laplacian $L \in \mathbb{R}^{N \times N}$ of graph $\mathcal{G}$ is defined as:

$$L_{ij} = \begin{cases} -\,l_{ij} & j \in \mathcal{N}_i\,, \\ \sum\limits_{j \in \mathcal{N}_i} l_{ij} & j = i\,, \\ 0 & \text{otherwise}\,, \end{cases} \tag{6.2}$$

where l_{ij} are given positive scalars. Next, the communication model satisfies the following assumptions.

Assumption 2 *It is assumed that*

(i) *vehicle i has knowledge of only the scalars l_{ij} for $j \in \mathcal{N}_i$,*

(ii) *vehicle i can exchange information only with the vehicles in the set of neighbors defined by $\mathcal{N}_i$, and*

(iii) *the communication graph $\mathcal{G}$ is connected and the Laplacian L is symmetric.*

The central idea of the distributed algorithms as described in [57–62], hinges on the properties of the Laplacian matrix L. Especially the nullspace of the matrix L is the key component of the approach. The characteristics of the nullspace are summarized in the following proposition.

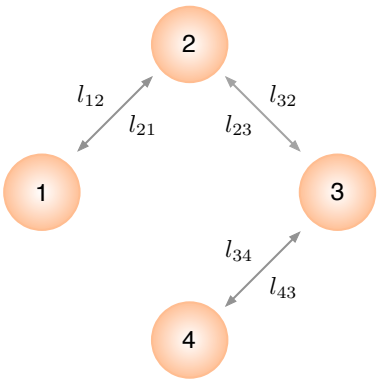


Figure 6.1: Example of a communication topology between four agents. The communication graph is connected and undirected.

Proposition 2 *The matrix L defined with respect to a connected graph $\mathcal{G}$ satisfies the following properties:*

(i) *The nullspace of L is given by* $\text{Null}(L) = \{\gamma \mathbb{1}_N \,|\, \gamma \in \mathbb{R}\}$, *where $\mathbb{1}_N \in \mathbb{R}^N$ whose components are all* 1*;*

(ii) *Let $\tilde{L} = L \otimes \mathbb{I}_n$, where $\mathbb{I}_n$ is the n-dimensional identity matrix. Then the nullspace of $\tilde{L}$ is given by* $\text{Null}(\tilde{L}) = \{\mathbb{1}_N \otimes x \,|\, x \in \mathbb{R}^n\}$.

An example of a communication topology is given in Figure 6.1. The graph $\mathcal{G}$ is undirected and connected, and the scalars l_{ij} are assumed to be all equal to 1. Then the Laplacian L associated with the communication topology from Figure 6.1, is given by

$$L = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}, \tag{6.3}$$

and it can be verified that Proposition 2 holds.

The next assumption on a local minimizer x^* of Problem (6.1), ensures the existence of unique Lagrange multipliers associated with the equality constraints $h(x)$.

Assumption 3 *The local minimizer x^* of Problem (6.1) is a regular point of the constraints $h(x) = 0$, that is, the gradient vectors $\nabla h_1(x^*), \nabla h_2(x^*), \ldots, \nabla h_N(x^*)$ are linearly independent.*

Using Assumption (3) we can state the first-order necessary conditions for Problem (6.1).

Lemma 2 *Let Assumption 3 hold and let $\hat{x}^*$ be the local minimizer for Problem (6.1). There exists a unique vector μ^* such that*

$$\nabla f^\top(\hat{x}^*) + \nabla h^\top(\hat{x}^*)\mu^* = 0\,. \tag{6.4}$$

6.1.2 An Equivalent Optimization Problem

The key step in the novel distributed algorithm that the authors of [57–62] propose, lies in the formulation of an *equivalent* optimization problem, from which the solution to the original Problem (6.1) can be obtained. Moreover, the distributed algorithms are in fact a result of using familiar (classical) centralized algorithms to solve this equivalent problem. The idea is to introduce *local estimates* $\hat{x}^i$ of the optimization vector x, where these local estimates are updated onboard each individual vehicle, and force an alignment of these local estimates towards the local minimizer x^*, i.e. the vehicles together agree on the common solution x^*.

Hence, we first define the vector $\hat{\boldsymbol{x}} := [\hat{x}^{1^\top}, \dots, \hat{x}^{N^\top}]^\top$, where $\hat{x}^i \in \mathbb{R}^n$. Next, the objective function $F : \mathbb{R}^{nN} \to \mathbb{R}$ is defined as

$$F(\hat{\boldsymbol{x}}) := \sum_{i=1}^{N} f_i(\hat{x}^i)\,.$$

The vector-valued function $H : \mathbb{R}^{nN} \to \mathbb{R}^N$ is given by

$$H(\hat{\boldsymbol{x}}) := [h_1(\hat{x}^1), \dots, h_N(\hat{x}^N)]^\top\,.$$

The functions $f_i(x)$ and $h_i(x)$ for all $i = 1, \dots, N$ are given as per Problem (6.1). Lastly, the matrix $\tilde{L}$ is defined as $\tilde{L} := L \otimes \mathbb{I}_n$, with $\mathbb{I}_n$ the n-dimensional identity matrix and L given in Equation (6.2). The nullspace of the matrix $\tilde{L}$ is given in Proposition 2. We formulate the following constrained optimization problem

$$\begin{aligned} \underset{\hat{\boldsymbol{x}} \in \mathbb{R}^{nN}}{\text{minimize}} \quad & F(\hat{\boldsymbol{x}}) \\ \text{subject to} \quad & H(\hat{\boldsymbol{x}}) = 0\,, \\ & \tilde{L}\hat{\boldsymbol{x}} = 0\,. \end{aligned} \tag{6.5}$$

The following result shows the equivalence between Problem (6.1) and Problem (6.5) under Assumptions 1 and 2. Especially, the assumption of a connected graph is fundamental, so that Proposition 2 holds and, therefore, the equivalence between the two problems follows.

Proposition 3 *Let Assumptions 1 and 2 hold. The vector $x^* \in \mathbb{R}^n$ is a local minimizer of Problem (6.1) if and only if $\hat{\boldsymbol{x}}^* = \mathbb{1}_N \otimes \hat{x}^*$ is a local minimizer of Problem (6.5).*

The proof of Proposition 3 can be found, for example, in [59] and [61]. The equivalence between the two problems is a result from the fact that the local minimizer $\hat{\boldsymbol{x}}^*$ for Problem (6.5) has a specific structure that is imposed by the equality constraint $\tilde{L}\hat{\boldsymbol{x}} = 0$. For the solution to exhibit this particular structure, it is important that the assumption of a connected communication topology holds. For example, if there is no

communication between vehicle 1 and 2 in Figure 6.1, i.e. the communication graph is disconnected, then $[0, \gamma, \gamma, \gamma]^\top$ with $\gamma \in \mathbb{R}$ also belongs to the nullspace Null(L) and we lose the fundamental structure that a local minimizer $\hat{\boldsymbol{x}}^*$ requires to ensure that both problems are equivalent.

Since the local minimizer $\hat{x}^*$ for Problem (6.1) can be extracted from the local minimizer $\hat{\boldsymbol{x}}^*$, the aim is to solve Problem (6.5) instead. Hence, before we present the first-order necessary conditions for Problem (6.5), we first derive the gradient $\nabla F(\hat{\boldsymbol{x}})$ and Jacobian $\nabla H(\hat{\boldsymbol{x}})$. Due to the introduction of the local estimates $\hat{x}^i$, $\nabla F(\hat{\boldsymbol{x}})$ and $\nabla H(\hat{\boldsymbol{x}})$ have a particular structure, and can be expressed in terms of the gradients $\nabla f_i(x)$ and $\nabla h_i(x)$:

$$\nabla F(\hat{\boldsymbol{x}}) = \left[\nabla f_1(\hat{x}^1), \dots, \nabla f_N(\hat{x}^N)\right], \qquad \nabla H(\hat{\boldsymbol{x}}) = \begin{bmatrix} \nabla h_1(\hat{x}^1) & \mathbf{0}_n^\top & \cdots & \mathbf{0}_n^\top \\ \mathbf{0}_n^\top & \nabla h_2(\hat{x}^2) & & \vdots \\ \vdots & & \ddots & \vdots \\ \mathbf{0}_n^\top & \mathbf{0}_n^\top & \cdots & \nabla h_N(\hat{x}^N) \end{bmatrix}, \tag{6.6}$$

where $\mathbf{0}_n \in \mathbb{R}^n$ whose components are all 0. Next, we proceed by deriving the first-order necessary conditions for Problem (6.5).

Lemma 3 *Let Assumptions 1, 2, and 3 hold let and $\hat{\boldsymbol{x}}^* = \mathbb{1}_N \otimes \hat{x}^*$ be the local minimizer for Problem (6.5). There exist unique vectors $\boldsymbol{\mu}^*$ and $\boldsymbol{\lambda}^* \in$ Range$(\tilde{L})$ such that*

$$\nabla F^\top(\hat{\boldsymbol{x}}^*) + \nabla H^\top(\hat{\boldsymbol{x}}^*)\boldsymbol{\mu}^* + \tilde{L}^\top \boldsymbol{\lambda} = 0\,, \tag{6.7}$$

for all $\boldsymbol{\lambda} \in \{\boldsymbol{\lambda}^* + \boldsymbol{\lambda}_\perp \,|\, \boldsymbol{\lambda}_\perp \in \text{Null}(\tilde{L}^\top)\}$.

Note that Equation (6.7) holds for any vector $\boldsymbol{\lambda}_\perp \in$ Null$(\tilde{L}^\top)$, since $\tilde{L}^\top \boldsymbol{\lambda}_\perp = 0$ by definition of the nullspace. Therefore, although $\boldsymbol{\lambda}^* \in$ Range$(\tilde{L})$ is unique, the vectors $\boldsymbol{\lambda}$ are not. The Lagrangian function of (6.5) is defined as

$$l(\hat{\boldsymbol{x}}, \boldsymbol{\mu}, \boldsymbol{\lambda}) := F(\hat{\boldsymbol{x}}) + \boldsymbol{\mu}^\top H(\hat{\boldsymbol{x}}) + \boldsymbol{\lambda}^\top \tilde{L}\, \hat{\boldsymbol{x}}\,. \tag{6.8}$$

6.1.3 Distributed Algorithm

The set of primal-dual equations for Problem (6.5) can be obtained from the Lagrangian function by setting $\nabla_{\hat{\boldsymbol{x}}}\, l(\hat{\boldsymbol{x}}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = 0$, $\nabla_{\boldsymbol{\mu}}\, l(\hat{\boldsymbol{x}}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = 0$ and $\nabla_{\boldsymbol{\lambda}}\, l(\hat{\boldsymbol{x}}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = 0$:

$$\begin{aligned} \nabla F^{\top}(\hat{\boldsymbol{x}}) + \nabla H^{\top}(\hat{\boldsymbol{x}})\boldsymbol{\mu} + \tilde{L}^{\top}\boldsymbol{\lambda} &= 0\,, \\ H(\hat{\boldsymbol{x}}) &= 0\,, \\ \tilde{L}\,\hat{\boldsymbol{x}} &= 0\,. \end{aligned} \tag{6.9}$$

Since Lemma 3 only provides the necessary conditions, solutions to the primal-dual equations are stationary points that are not necessarily local minimizers for Problem (6.5). Sufficient conditions must also be checked to determine whether a stationary point found from Equation (6.9) is indeed a local minimizer.

We can use the *first-order Lagrangian method* [8, 54], to solve the set of primal-dual equations. The algorithm to solve Equation (6.9) is given by

$$\begin{aligned} \hat{\boldsymbol{x}}_{k+1} &= \hat{\boldsymbol{x}}_k - \tau \left[\nabla F^{\top}(\hat{\boldsymbol{x}}_k) + \nabla H^{\top}(\hat{\boldsymbol{x}}_k)\boldsymbol{\mu}_{\boldsymbol{k}} + \tilde{L}^{\top}\boldsymbol{\lambda}_{\boldsymbol{k}}\right]\,, \\ \boldsymbol{\mu}_{k+1} &= \boldsymbol{\mu}_k + \tau\, H(\hat{\boldsymbol{x}}_k)\,, \\ \boldsymbol{\lambda}_{k+1} &= \boldsymbol{\lambda}_k + \tau\, \tilde{L}\,\hat{\boldsymbol{x}}_k\,, \end{aligned} \tag{6.10}$$

for some sufficiently small step size $\tau > 0$. The following theorem states the local convergence properties of algorithm (6.10). If, under some assumptions on the functions $f_i(\hat{x}^i)$ and $h_i(\hat{x}^i)$, the initial values are close enough to a solution of the primal-dual equations (6.9) and a sufficiently small step size τ is used, then the sequence $\{\hat{\boldsymbol{x}}_k, \boldsymbol{\mu}_k, \boldsymbol{\lambda}_k\}$ converges to this solution.

Theorem 12 *Let Assumptions 1, 2, and 3 hold and let $(\hat{\boldsymbol{x}}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*)$ be a local minimizer-Lagrange multipliers pair of Problem (6.5), where $\boldsymbol{\lambda}^* \in \text{Range}(\tilde{L})$. Assume also that $\nabla_{\hat{\boldsymbol{x}}\hat{\boldsymbol{x}}}\, l(\hat{\boldsymbol{x}}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*)$ is positive definite. Then there exists $\bar{\tau}$, such that for all $\tau \in (0, \bar{\tau}]$, the set $(\hat{\boldsymbol{x}}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^* + \text{Null}(\tilde{L}^{\top}))$ is an attractor of iteration (6.10) and if the sequence $\{\hat{\boldsymbol{x}}_k, \boldsymbol{\mu}_k, \boldsymbol{\lambda}_k\}$ converges to the set $(\hat{\boldsymbol{x}}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^* + \text{Null}(\tilde{L}^{\top}))$, the rate of convergence of $\|\hat{\boldsymbol{x}}_k - \hat{\boldsymbol{x}}^*\|$, $\|\boldsymbol{\mu}_k - \boldsymbol{\mu}^*\|$ and $\|\boldsymbol{\lambda}_k - [\boldsymbol{\lambda}^* + \text{Null}(\tilde{L}^{\top})]\|$ is linear.*

Since the matrix $\tilde{L}$ is not full rank, it cannot be guaranteed that the sequence $\boldsymbol{\lambda}_k$ converges to the unique $\boldsymbol{\lambda}^* \in \text{Range}(\tilde{L})$ but rather to a point in the set $\{\boldsymbol{\lambda}^* + \text{Null}(\tilde{L}^{\top})\}$. Although algorithm (6.10) is obtained from a classical *centralized* solver, it in fact results in a *distributed* formulation. This becomes clear if we extract the ith n-dimensional component of $\hat{\boldsymbol{x}}_k$ and the associated multipliers. Recalling the particular structures of the gradient $\nabla F(\hat{\boldsymbol{x}})$ and Jacobian $\nabla H(\hat{\boldsymbol{x}})$ from Equation (6.6), it is easy to verify that we

obtain the following iteration for the ith n-dimensional component of $\hat{\boldsymbol{x}}_k$

$$\hat{x}^i_{k+1} = \hat{x}^i_k - \tau \left[\nabla f_i^\top(\hat{x}^i_k) + \nabla h_i^\top(\hat{x}^i_k)\mu_{i,k} + \sum_{j \in \mathcal{N}_i} (l_{ij}\lambda_{i,k} - l_{ji}\lambda_{j,k}) \right], \tag{6.11a}$$

$$\mu_{i,k+1} = \mu_{i,k} + \tau\, h_i(\hat{x}^i_k)\,, \tag{6.11b}$$

$$\lambda_{i,k+1} = \lambda_{i,k} + \tau \sum_{j \in \mathcal{N}_i} l_{ij}(\hat{x}^i_k - \hat{x}^j_k)\,. \tag{6.11c}$$

If a priori the step size τ is known to all vehicles, then from Assumptions 1 and 2 it is clear that algorithm (6.11) can be executed by vehicle i, since it has access to all required information, that is either computed onboard or exchanged with its neighbors $j \in \mathcal{N}_i$ over the communication network. Note that the assumption on a symmetric Laplacian L (Assumption 2(iii)) is necessary, as vehicle i requires the knowledge over the scalars l_{ji} for $j \in \mathcal{N}_i$. Lastly, the following corollary gives the condition under which the iteration (6.11a) and (6.11b) ensures convergence to a local minimizer of Problem (6.1).

Corollary 2 *Let Assumptions 1, 2, and 3 hold and let $(\hat{x}^*, \mu^*)$ be a local minimizer-Lagrange multiplier pair of Problem (6.1). Assume also that $\nabla^2 f_i(\hat{x}^*) + \mu_i^* \nabla^2 h_i(\hat{x}^*)$ is positive definite for all $i = 1, \ldots, N$. Then there exists $\bar{\tau}$, such that for all $\tau \in (0, \bar{\tau}]$, $(\hat{x}^*, \mu^*)$ is a point of attraction for iteration (6.11a) and (6.11b), for all $i = 1, \ldots, N$, and if the sequence $\{\hat{x}^i_k, \mu_{i,k}\}$ converges to $(\hat{x}^*, \mu^*)$, the rate of convergence of $\|\hat{x}^i_k - \hat{x}^*\|$ and $\|\mu_{i,k} - \mu^*\|$ is linear.*

In the next section, we use these results to formulate a distributed algorithm, based on the bundle method, that is suitable for application to the nonsmooth cooperative trajectory-generation problem.

6.2 A Distributed Bundle Method

In this section, we present a *distributed bundle method*. We derive a distributed algorithm for solving nonsmooth constrained optimization problems by combining the distributed nonlinear programming method that was presented in the previous section, with the bundle method discussed in Chapter 5. Our approach differs from the previously discussed and existing work in the following ways:

1. Instead of using one arbitrary subgradient from the subdifferential ∂f at x, when constructing the piecewise linear approximation of the locally Lipschitz continuous function f, we employ the entire subdifferential ∂f at x;

2. The complicated line search procedure, characteristic of bundle methods, is avoided, since a feasible descent direction can be determined for the objective function;

3. The stopping criterion is more intuitive compared to the one used for classical bundle methods;

4. The distributed algorithms are derived for the *dual* problem, while the distributed nonlinear programming method from Section 6.1 was developed for the *primal* problem.

Building upon the unconstrained nonsmooth optimization theory from Section 5.3.1, we will expand on the theory with results obtained for constrained nonsmooth optimization problems. The presentation will be interwoven with the derivation of the proposed distributed bundle method. In what follows, multiple optimization problems are presented that eventually culminate in the formulation of a set of distributed algorithms for solving the nonsmooth problem at hand. We summarize the different formulations here to ease the understanding of the discussion presented in the following sections.

1. Problem (P1) is the original *centralized* problem at hand;

2. Using the techniques discussed in the previous section, we formulate Problem (P2) where essentially the gradients and Jacobians of the functions and vector-valued functions, respectively, are *decoupled*.

3. Next, in order to handle the inequality constraints, an improvement function is introduced. Problem (P3), that is formulated in terms of this improvement function, and in fact not implementable, allows to analyze the problem for its necessary optimality conditions.

4. Subsequently, an approach based on the feasible point descent method is presented, where the direction finding problem is formulated in terms of a *linearization* of the improvement function at the *current* iteration point. This approach will not give satisfactory results, as it generates discontinuous descent directions with respect to the optimization variable.

5. A direction finding problem (DFP1) is reformulated using an alternative linearization of the improvement function, the so-called cutting plane model. It is shown that a feasible descent direction for this piecewise linear approximation of the improvement function, is in fact a feasible descent direction for the original objective function.

6. Problem (DFP1) is still a nonsmooth optimization problem and, hence, it is converted into a *smooth* direction finding problem (DFP2). This causes coupling of the constraints and the same technique from Section 6.1 is used to decouple the system. This results in Problem (DFP3).

7. Lastly, the dual problem of (DFP3) is derived and denoted by (DFP4). This is the optimization problem that is solved using the first-order Lagrangian method, which results in a *distributed* formulation of a centralized algorithm.

6.2.1 Problem Formulation

We have seen that for a team of N vehicles, we obtain the following nonlinear constrained optimization problem

$$\begin{aligned} \text{(P1)} \quad & \text{minimize} \quad f(x) \\ & \text{subject to} \quad g(x) \leq 0\,, \end{aligned} \tag{6.12}$$

where $g(x) = [\, g_1(x), \ldots, g_m(x) \,]^\top$ and $g_i(x)$ are of the form

$$g_i(x) = \max_{\zeta} \gamma_i(x, \zeta)\,, \qquad \text{for} \quad i = 1, \ldots, m\,. \tag{6.13}$$

Recall that the functions $\gamma_i(x, \zeta)$ are Bézier polynomials in ζ and that $g_i(x)$ represent the various constraints, such as the dynamic and deconfliction constraints. Without loss of generality, and for the sake of simplicity, we assume that each vehicle has one optimization parameter and one inequality constraint and, hence, $x \in \mathbb{R}^N$ and $m = N$, respectively. In what follows, the optimization problem (P1) is considered to be a *multi-objective* optimization problem, i.e. the objective function $f(x)$ is of the form

$$f(x) = [\, f_1(x), \ldots, f_N(x) \,]^\top\,,$$

where each $f_i(x)$ corresponds to the cost function of vehicle i. Lastly, we define the *feasible* set

$$\Omega := \{x \in \mathbb{R}^N \,|\, g(x) \leq 0\}\,,$$

and assume that Ω is nonempty. Additionally, we make the following assumptions on the functions $f_i(x)$ and $g_i(x)$.

Assumption 4 *We assume that*

(i) *the objective functions $f_i : \mathbb{R}^N \to \mathbb{R}$ are of the form*

$$f_i(x) = \max_{\zeta} \phi_i(x, \zeta)\,,$$

where the functions $\phi_i(\cdot, \cdot)$ are continuous and their gradients $\nabla_x \phi_i(\cdot, \cdot)$ exist and are continuous,

(ii) *the functions $\gamma_i(\cdot, \cdot)$ given in Equation (6.13) are continuous and their gradients $\nabla_x \gamma_i(\cdot, \cdot)$ exist and are continuous, and*

(iii) *vehicle i has knowledge of only functions $f_i(x)$ and $g_i(x)$.*

Then, from Assumption 4 and Corollary 1 in Chapter 5, we know that the functions f_i and g_i are locally Lipschitz continuous.

Remark 3 *The choice to consider a multi-objective optimization problem, is made in order to formulate a distributed algorithm based on the bundle method. A collective objective function $f(x)$, defined as the sum of the individual objective functions $f_i(x)$ of the vehicles, causes a coupling between the individual objective functions $f_i(x)$ in the construction of the piecewise linear approximation of the collective objective function $f(x)$. This coupling can be avoided by constructing individual piecewise linear approximations, a natural approach of multi-objective bundle methods.*

Remark 4 *Although Equation (6.13) and Assumption 4(ii) appear to be reasonable if we consider our cooperative trajectory-generation framework, Assumption 4(i) on the objective functions $f_i(x)$ seems to be overly restrictive. In fact, continuously differentiable functions and locally Lipschitz continuous $f_i(x)$, for which we can obtain the whole subdifferential (efficiently), are admissible classes of functions within our framework and, hence, the functions $f_i(x)$ do not necessarily need to be of the form as prescribed by Assumption 4(i).*

First, we need to define the notion of optimality for the multi-objective optimization problem. The following definitions characterize different *Pareto* optima for multi-objective optimization problems and are given in [55].

Definition 10 *A vector x^* is said to be a global Pareto optimum of (P1), if there does not exist $x \in \Omega$ such, that $f_i(x) \leq f_i(x^*)$ for all $i = 1, \dots, N$ and $f_j(x) < f_j(x^*)$ for some j. Vector x^* is said to be a global weak Pareto optimum of (P1), if there does not exist $x \in \Omega$ such that $f_i(x) < f_i(x^*)$ for all $i = 1, \dots, N$. Vector x^* is a local (weak) Pareto optimum of (P1), if there exists $\rho > 0$ such that x^* is a global (weak) Pareto optimum on $\mathcal{B}(x^*; \rho) \cap \Omega$.*

Next, as discussed in Section 6.1, an equivalent optimization problem is formulated, which facilitates the derivation of a distributed optimization algorithm. To this end, we introduce *local estimates* $\hat{x}^i$ of the vector x that are computed onboard each vehicle i and let $\hat{\boldsymbol{x}} := [\hat{x}^{1\top}, \dots, \hat{x}^{N\top}]^\top$. Then, we define the vector-valued functions $F : \mathbb{R}^{N^2} \to \mathbb{R}^N$ and $G : \mathbb{R}^{N^2} \to \mathbb{R}^N$ as

$$F(\hat{\boldsymbol{x}}) := \left[f_1(\hat{x}^1), \dots, f_N(\hat{x}^N)\right]^\top \quad \text{and}$$
$$G(\hat{\boldsymbol{x}}) := \left[g_1(\hat{x}^1), \dots, g_N(\hat{x}^N)\right]^\top,$$

respectively, where Assumption 4 holds on the functions f_i and g_i. Now, let us consider the following

constrained multi-objective optimization problem

$$\begin{aligned} \text{(P2)} \quad \text{minimize} \quad & F(\hat{\boldsymbol{x}}) \\ \text{subject to} \quad & G(\hat{\boldsymbol{x}}) \leq 0\,, \\ & \tilde{L}\hat{\boldsymbol{x}} = 0\,, \end{aligned}$$

with $\tilde{L} = L \otimes \mathbb{I}_N$. The Laplacian L characterizes the communication graph $\mathcal{G}$, which satisfies Assumption 2 in Section 6.1.1. Finally, the *feasible set* Ω is defined as $\Omega := \Omega_g \cap \Omega_{\tilde{L}}$, where

$$\begin{aligned} \Omega_g &:= \{\hat{\boldsymbol{x}} \in \mathbb{R}^{N^2} \,|\, G(\hat{\boldsymbol{x}}) \leq 0\}\,, \\ \Omega_{\tilde{L}} &:= \{\hat{\boldsymbol{x}} \in \mathbb{R}^{N^2} \,|\, \tilde{L}\hat{\boldsymbol{x}} = 0\}\,. \end{aligned}$$

Using a similar proposition, given in [60] for inequality constrained optimization problems, as Proposition 3, we conclude that the multi-objective optimization problem (P2) is equivalent to the optimization problem at hand and, therefore, if $\hat{\boldsymbol{x}}^* = \mathbb{1}_N \otimes \hat{x}^*$ is a Pareto optimal solution for (P2), then $\hat{x}^*$ is a Pareto optimal solution to Problem (P1). Hence, we consider optimization Problem (P2) in the derivation of a distributed optimization algorithm for solving Problem (P1).

Unlike equality constraints, in general, inequality constraints complicate solving constrained optimization problems. Only *active* inequality constraints, i.e. $g_i(x) = 0$ at a feasible point x, restrict the domain of feasibility in the neighborhood of x, while *inactive* inequality constraints, i.e. $g_i(x) < 0$ at a feasible point x, have no influence in the neighborhood of x. Therefore, if the set of active constraints at the optimal solution x^* is known, then the simple approach would be to append the equality constraints with this set of active constraints (and discard the remaining inequality constraints), and treat the problem as an equality constrained optimization problem. Unfortunately, the set of active inequality constraints at the optimal solution is, in general, unknown beforehand. One approach is to make an initial guess of the set of active inequality constraints and, as the algorithm progresses, update this set by adding potentially active inequality constraints while discarding those that are in fact inactive. This is the so-called *Active Set Method* [54, 72]. Other popular approaches augment the objective function with terms containing these inequality constraints and treat the constrained optimization problem as an unconstrained problem. An example of such methods is the *Interior Point Method* (or *Barrier Method*) [54, 72]. Every method has its advantages and disadvantages and whether one approach is efficient and outperforms the other depends on the functions $g_i(x)$.

Here, we construct an objective function that captures the effects of the inequality constraints of the

original problem. Let us define the *improvement function* $H : \mathbb{R}^{N^2} \times \mathbb{R}^{N^2} \to \mathbb{R}$ as

$$H(\hat{\boldsymbol{x}}; \hat{\boldsymbol{y}}) := \max \{ f_i(\hat{x}^i) - f_i(\hat{y}^i) \, , \, g_i(\hat{x}^i) \} \, , \qquad \text{for} \quad i = 1, \ldots, N \, , \tag{6.14}$$

where $\hat{\boldsymbol{y}} := [\, \hat{y}^{1\top}, \ldots, \hat{y}^{N\top}]^\top$. Note that $H(\hat{\boldsymbol{x}}; \hat{\boldsymbol{y}})$ is locally Lipschitz continuous by Theorem 6, since the functions f_i and g_i are locally Lipschitz continuous.

Theorem 13 *Consider the problem (P2). Suppose that $\hat{\boldsymbol{x}}^* \in \mathbb{R}^{N^2}$ is a local weak Pareto optimal solution for (P2). Then $\hat{\boldsymbol{x}}^*$ is a local minimizer of $H(\cdot\,; \hat{\boldsymbol{x}}^*)$ over $\Omega_{\tilde{L}}$.*

Proof: The proof is given in Appendix B.4. □

Note that Theorem 13 gives a necessary condition for $\hat{\boldsymbol{x}}^*$ to be a local weak Pareto optimum for (P1) and, hence, the converse may not hold. Suppose that $\hat{\boldsymbol{x}}^*$ is a local minimizer of $H(\cdot\,; \hat{\boldsymbol{x}}^*)$ over $\Omega_{\tilde{L}}$, and that $g_i(\hat{x}^*) = 0$ for some $i = 1, \ldots, N$. Then, we may have that $H(\hat{\boldsymbol{x}}; \hat{\boldsymbol{x}}^*) = 0$, while $g_i(\hat{x}^i) = 0$ for some $i = 1, \ldots, N$ and $\hat{\boldsymbol{x}} \in \mathcal{B}(\hat{\boldsymbol{x}}^*; \rho^*) \cap \Omega_{\tilde{L}}$, where ρ^* is the radius associated with $\hat{\boldsymbol{x}}^*$. Hence, this does not necessarily exclude that $f_i(\hat{x}^i) - f_i(\hat{x}^*) < 0$ for all $i = 1, \ldots, N$ and, therefore, $\hat{\boldsymbol{x}}^*$ is not a local weak Pareto optimum for Problem (P2). The next Assumption defines a so-called constraint qualification, so that the converse of Theorem 13 also holds.

Assumption 5 *Suppose that $\hat{\boldsymbol{x}}^* \in \mathbb{R}^{N^2}$ is a local minimizer of $H(\cdot\,; \hat{\boldsymbol{x}}^*)$ over $\Omega_{\tilde{L}}$. Then we assume that either $g_i(\hat{x}^*) < 0$ for all $i = 1, \ldots, N$ or $g_i(\hat{x}^*) = 0$ and $0 \notin \partial g_i(\hat{x}^*)$ for some $i = 1, \ldots, N$.*

Theorem 14 *Let Assumption 5 hold and let $\hat{\boldsymbol{x}}^* \in \mathbb{R}^{N^2}$ be a local minimizer of $H(\cdot\,; \hat{\boldsymbol{x}}^*)$ over $\Omega_{\tilde{L}}$. Then $\hat{\boldsymbol{x}}^*$ is a local weak Pareto optimal solution for Problem (P2).*

The theorem can be proved along the same lines as Theorem 3.2.3 in [76] and by using Definition 10 for a local weak Pareto optimum. The constraint qualification prevents that the constraint $g_i(\hat{x}^i)$ for all $i = 1, \ldots, N$, achieves a maximum that is equal to 0 at a local minimizer $\hat{\boldsymbol{x}}^*$ of $H(\cdot\,; \hat{\boldsymbol{x}}^*)$ over $\Omega_{\tilde{L}}$. Theorem 14 allows us to reformulate the constrained optimization problem (P2) into an *equality* constrained optimization problem using the improvement function $H(\hat{\boldsymbol{x}}; \hat{\boldsymbol{y}})$:

$$\begin{aligned} \text{(P3)} \quad & \text{minimize} \quad && H(\hat{\boldsymbol{x}}; \hat{\boldsymbol{x}}^*) \\ & \text{subject to} \quad && \tilde{L}\, \hat{\boldsymbol{x}} = 0 \, . \end{aligned}$$

Problem (P3), as formulated above, is definitely not implementable as the optimal solution $\hat{\boldsymbol{x}}^*$ is assumed to be known. However, it serves as a basis for the development of iterative optimization algorithms and allows

to derive the necessary optimality conditions for constrained nonsmooth optimization problems. Before we formulate these necessary conditions, we need the next theorem, which states the necessary conditions for an optimization problem, subject to a constraint set Ω, to attain a local minimum at x^*. The proof is given in [56], as well as the definitions of *tangent cone* $\mathrm{T}_\Omega(x^*)$ and *normal cone* $\mathrm{N}_\Omega(x^*)$.

Theorem 15 *If f is locally Lipschitz at x and attains its local minimum over the set $\Omega \subset \mathbb{R}^n$ at $x^* \in \Omega$, then*

$$0 \in \partial f(x^*) + \mathrm{N}_\Omega(x^*)\,.$$

For the constraint set $\Omega_{\tilde{L}} := \{\hat{\boldsymbol{x}} \in \mathbb{R}^{N^2} \,|\, \tilde{L}\,\hat{\boldsymbol{x}} = 0\}$, we can characterize its tangent plane at a local minimizer $\hat{\boldsymbol{x}}^*$ of Problem (P3).

Proposition 4 *The tangent cone to $\Omega_{\tilde{L}}$ at $\hat{\boldsymbol{x}}^*$, denoted by $\mathrm{T}_{\Omega_{\tilde{L}}}(\hat{\boldsymbol{x}}^*)$, is given by*

$$\mathrm{T}_{\Omega_{\tilde{L}}}(\hat{\boldsymbol{x}}^*) = \mathrm{Null}(\tilde{L})\,.$$

Proof: The proof is given in Appendix B.5. □

Using Theorem 15 and Proposition 4, we can formally state the necessary conditions for Problem (P3) to attain a local minimum at $\hat{\boldsymbol{x}}^*$.

Theorem 16 *Suppose Problem (P3) satisfies Assumption 5 and let $\hat{\boldsymbol{x}}^* = \mathbb{1}_N \otimes \hat{x}^*$ be a local minimizer of (P3). Then there exists a vector $\nu \in \mathbb{R}^{N^2}$ such that*

$$0 \in \partial H(\hat{\boldsymbol{x}}^*; \hat{\boldsymbol{x}}^*) + \tilde{L}^\top \nu\,. \tag{6.15}$$

Proof: The proof is given in Appendix B.6. □

6.2.2 Feasible Point Descent Method

Section 5.3.1 discussed the descent method, on which many optimization solvers for unconstrained problems are based. However, for a constrained optimization problem, finding a descent direction d as defined by Definition 9 is not sufficient, as d may not necessarily be *feasible* with respect to the feasible set Ω. Hence, we give a formal definition of a *feasible descent direction*.

Definition 11 *The direction $d \in \mathbb{R}^n$ is called a feasible descent direction subject to Ω for $f : \mathbb{R}^n \to \mathbb{R}$ at x_k,*

if there exists $\epsilon > 0$ *such that*

$$f(x_k + \tau\, d) < f(x_k) \qquad \textit{and} \qquad x_k + \tau\, d \in \Omega\,, \qquad \textit{for all} \;\; \tau \in (0, \epsilon\,]\,.$$

The key step in *feasible point descent methods* is to determine the feasible descent direction d as defined by Definition 11. The optimization problem that is formulated to obtain such a direction is termed the direction finding problem (DFP). Suppose that $\boldsymbol{d} := [{d^1}^\top, \ldots, {d^N}^\top]^\top$ is a feasible descent direction subject to $\Omega_{\tilde{L}}$ for the improvement function H, given by Equation (6.14), at the current *non-optimal* iteration point $\hat{\boldsymbol{x}}_k$. From Definition 11, we know that there exists $\epsilon > 0$ such that

$$H(\hat{\boldsymbol{x}}_k + \tau\, \boldsymbol{d}; \hat{\boldsymbol{x}}_k) < H(\hat{\boldsymbol{x}}_k; \hat{\boldsymbol{x}}_k) \qquad \text{and} \qquad \hat{\boldsymbol{x}}_k + \tau\, \boldsymbol{d} \in \Omega_{\tilde{L}}\,, \qquad \text{for all} \;\; \tau \in (0, \epsilon\,]\,,$$

and note that $H(\hat{\boldsymbol{x}}_k; \hat{\boldsymbol{x}}_k) = 0$, since $\hat{\boldsymbol{x}}_k$ is feasible subject to Ω. Hence, it is easily verified from the definition of $H(\hat{\boldsymbol{x}}; \hat{\boldsymbol{y}})$, that for $i = 1, \ldots, N$

$$f_i(\hat{x}_k^i + \tau d^i) < f_i(\hat{x}_k^i)\,, \qquad \text{for all} \;\; \tau \in (0, \epsilon\,]\,,$$

where $\hat{\boldsymbol{x}} + \tau\, \boldsymbol{d} \in \Omega$. In other words, a feasible descent direction $\boldsymbol{d}$ subject to $\Omega_{\tilde{L}}$ of H at $\hat{\boldsymbol{x}}_k$ is also a feasible descent direction subject to Ω of F at $\hat{\boldsymbol{x}}_k$. Thus, we can formulate the DFP in terms of the improvement function $H(\hat{\boldsymbol{x}}; \hat{\boldsymbol{y}})$. Recall that Problem (P1) belongs to the class of nonsmooth optimization problems; in particular it belongs to the class of semi-infinite programming due the fact that the objective functions $f_i(x)$ and inequality constraints $g_i(x)$ consist of continuous maximum functions. Hence, the improvement function $H(\hat{\boldsymbol{x}}; \hat{\boldsymbol{y}})$ is too intractable to formulate an optimization problem with. Instead, we find a piecewise linear approximation of $H(\hat{\boldsymbol{x}}; \hat{\boldsymbol{y}})$ at the current non-optimal iteration point $\hat{\boldsymbol{x}}_k$, and show that if a direction $\boldsymbol{d}$ is a feasible descent direction for the approximation, then it is a feasible descent direction for the multi-objective function F at $\hat{\boldsymbol{x}}_k$.

To this end, using Definition 8 from Chapter 5, we first obtain the linearizations for $f_i(\hat{x}^i)$ and $g_i(\hat{x}^i)$. We start by defining the ξ^i-linearization of f_i and g_i at $\hat{y}^i$:

$$\begin{aligned} \tilde{f}_i(\hat{x}^i) &:= f_i(\hat{y}^i) + \langle\, \xi^i, \hat{x}^i - \hat{y}^i \,\rangle\,, & \xi^i &\in \partial f_i(\hat{y}^i)\,, \\ \tilde{g}_i(\hat{x}^i) &:= g_i(\hat{y}^i) + \langle\, \rho^i, \hat{x}^i - \hat{y}^i \,\rangle\,, & \rho^i &\in \partial g_i(\hat{y}^i)\,. \end{aligned}$$

The linearizations of f_i and g_i at $\hat{y}^i$ are then, respectively, given by:

$$\begin{aligned} \bar{f}_i(\hat{x}^i) &:= \max_{\xi^i \in \partial f_i(\hat{y}^i)} \tilde{f}_i(\hat{x}^i)\,, \\ \bar{g}_i(\hat{x}^i) &:= \max_{\rho^i \in \partial g_i(\hat{y}^i)} \tilde{g}_i(\hat{x}^i)\,. \end{aligned} \tag{6.16}$$

Finally, we define the following linear approximation of $H(\hat{\boldsymbol{x}}; \hat{\boldsymbol{y}})$ at $\hat{\boldsymbol{y}}$:

$$\hat{H}(\hat{\boldsymbol{x}}) := \max_{i=1,\ldots,N} \{\, \bar{f}_i(\hat{x}^i) - f_i(\hat{y}^i)\,,\, \bar{g}_i(\hat{x}^i) \,\}\,. \tag{6.17}$$

The next theorem states that a feasible descent direction $\boldsymbol{d}$ for the approximation $\hat{H}$ at $\hat{\boldsymbol{x}}_k$ is indeed a feasible descent direction for the multi-objective function F.

Theorem 17 *If the direction $\boldsymbol{d} \in \mathbb{R}^{N^2}$ is a descent direction for $\hat{H}$ at $\hat{\boldsymbol{x}}_k \in \Omega$ and feasible subject to $\Omega_{\tilde{L}}$, then $\boldsymbol{d}$ is a descent direction for F at $\hat{\boldsymbol{x}}_k$ and feasible subject to Ω.*

Proof: The proof is given in Appendix B.7. □

Theorem 17 permits us to find a feasible descent direction $\boldsymbol{d}$ at iteration point $\hat{\boldsymbol{x}}_k$ for the nonsmooth optimization problem (P2) in terms of the linear approximation $\hat{H}(\hat{\boldsymbol{x}})$ of the improvement function H at $\hat{\boldsymbol{x}}_k$. Using this result, the next theorem shows how to obtain such a feasible descent direction by formulating an optimization problem using this linear approximation. The proof of the theorem can be found in [56].

Theorem 18 *At iteration point $\hat{\boldsymbol{x}}_k$, consider the problem*

$$\begin{aligned} \underset{\boldsymbol{d}}{\text{minimize}} \quad & \hat{H}(\hat{\boldsymbol{x}}_k + \boldsymbol{d}) + \frac{1}{2}\|\boldsymbol{d}\|^2 \\ \text{subject to} \quad & \hat{\boldsymbol{x}}_k + \boldsymbol{d} \in \Omega_{\tilde{L}}\,. \end{aligned} \tag{6.18}$$

The Problem (6.18) has a unique solution $\boldsymbol{d}^* = -\xi^*$, where

$$\xi^* = \arg\min\{\|\xi\| \,|\, \xi \in \partial \hat{H}(\hat{\boldsymbol{x}}_k) \;\; \textit{and} \;\; \hat{\boldsymbol{x}}_k - \xi \in \Omega_{\tilde{L}}\}\,.$$

The term $\frac{1}{2}\|\boldsymbol{d}\|^2$ is added to the cost function, in order to convexify the problem and guarantee a unique solution for Problem (6.18). We now show that the unique solution $\boldsymbol{d}^*$ is indeed a descent direction for $\hat{H}$ at $\hat{\boldsymbol{x}}_k$. Let $\boldsymbol{d}^*$, not equal to 0, be the global minimizer to Problem (6.18). Therefore, we have that

$$\hat{H}(\hat{\boldsymbol{x}}_k + \boldsymbol{d}^*) + \|\boldsymbol{d}^*\|^2 < \hat{H}(\hat{\boldsymbol{x}}_k + \boldsymbol{d}) + \|\boldsymbol{d}\|^2\,,$$

for all $\boldsymbol{d} \in \mathbb{R}^{N^2}$ such that $\hat{\boldsymbol{x}}_k + \boldsymbol{d} \in \Omega_{\tilde{L}}$. Suppose that $\boldsymbol{d}^*$ is not a descent direction, i.e. $\hat{H}(\hat{\boldsymbol{x}}_k + \boldsymbol{d}^*) \geq \hat{H}(\hat{\boldsymbol{x}}_k)$. Then, for $\boldsymbol{d} = 0$ we have

$$\hat{H}(\hat{\boldsymbol{x}}_k + 0) + \|0\|^2 < \hat{H}(\hat{\boldsymbol{x}}_k + \boldsymbol{d}^*) + \|\boldsymbol{d}^*\|^2 ,$$

which contradicts the fact that $\boldsymbol{d}^*$ is the global minimizer to Problem (6.18). Therefore, the unique solution $\boldsymbol{d}^*$ must be a descent direction of $\hat{H}$ at $\hat{\boldsymbol{x}}_k$.

For the same reasons given for Theorem 11, we cannot directly apply Theorem 18 to obtain the feasible descent direction. Although we can determine the whole subdifferentials of the nonsmooth objective functions $f_i(\hat{x}^i)$ and the inequality constraints $g_i(\hat{x}^i)$, see Section 5.2.2, we cannot directly use Theorem 18, as it will render discontinuous feasible descent directions. In what follows, we present a modification to the bundle method, as described in Section 5.3.2, that uses the whole subdifferential information to construct a piecewise linear approximation of the improvement function, and that eventually allows for a distributed formulation.

6.2.3 Distributed Direction Finding Problem

The idea behind the bundle method [48] is to construct a piecewise linear approximation of the improvement function $H(x; y)$. At iteration point x_k, the method assumes that for each nonsmooth function $f(x)$, one can only determine one subgradient $\xi \in \partial f(x_k)$ and the function value $f(x_k)$. This information about the current iteration point is stored and added to the bundle that was collected at previous iteration points. Therefore, this bundle of subgradients grows as the algorithm continues, and information from earlier iteration points are taken into account in computing the current feasible descent direction d_k.

Since we are able to compute the whole subdifferential, we will exploit this extra information and take a different approach than the classical bundle method as described in Section 5.3.2. We start by defining the linear approximations of the non-differentiable functions. Again, let $\hat{\boldsymbol{x}}_k$ be the current (non-optimal) iteration point at the kth iteration of the algorithm and suppose a *bundle of subdifferentials* at (previous) iteration points $\hat{\boldsymbol{x}}_\ell$ has been collected and stored for $\ell \in \mathcal{I}_k = \{1, \ldots, k\}$. First, recall that the functions f_i and g_i are given as

$$f_i(\hat{x}^i) = \max_{\zeta} \phi_i(\hat{x}^i, \zeta) ,$$
$$g_i(\hat{x}^i) = \max_{\zeta} \gamma_i(\hat{x}^i, \zeta) .$$

Then, let the sets of maximizers for $\phi_i(\hat{x}^i, \zeta)$ and $\gamma_i(\hat{x}^i, \zeta)$ at $\hat{\boldsymbol{x}}_\ell$ be defined as

$$\mathcal{Z}_\ell^{f_i} := \{\bar{\zeta} \in [0,1] \,|\, \phi_i(\hat{x}_\ell^i, \bar{\zeta}) = f_i(\hat{x}_\ell^i)\}\,,$$
$$\mathcal{Z}_\ell^{g_i} := \{\bar{\zeta} \in [0,1] \,|\, \gamma_i(\hat{x}_\ell^i, \bar{\zeta}) = g_i(\hat{x}_\ell^i)\}\,,$$

and their respective index sets as $\mathcal{I}_\ell^{f_i} := \{1, \ldots, n_{\zeta,\ell}^{f_i}\}$ and $\mathcal{I}_\ell^{g_i} := \{1, \ldots, n_{\zeta,\ell}^{g_i}\}$, where $n_{\zeta,\ell}^{f_i} := \mathrm{card}(\mathcal{Z}_\ell^{f_i})$ and $n_{\zeta,\ell}^{g_i} := \mathrm{card}(\mathcal{Z}_\ell^{g_i})$, respectively. We collect and store the subdifferentials ∂f_i and ∂g_i at $\hat{x}_\ell^i$ for $\ell \in \mathcal{I}_k$, where we know from Corollary 1 that the subdifferentials can be determined as

$$\partial f_i(\hat{x}_\ell^i) = \operatorname*{conv}_{m \in \mathcal{I}_\ell^{f_i}} \{\xi_{\ell,m}^i\}\,, \qquad i = 1, \ldots, N\,, \quad \ell \in \mathcal{I}_k\,,$$
$$\partial g_i(\hat{x}_\ell^i) = \operatorname*{conv}_{m \in \mathcal{I}_\ell^{g_i}} \{\rho_{\ell,m}^i\}\,, \qquad i = 1, \ldots, N\,, \quad \ell \in \mathcal{I}_k\,,$$

where $\xi_{\ell,m}^i$ and $\rho_{\ell,m}^i$, by virtue of introducing the local estimates $\hat{x}^i$, have the following structure

$$\begin{aligned}
\xi_{\ell,m}^i &= [\mathbf{0}_N^\top, \ldots, \underbrace{\nabla\phi_i(\hat{x}_\ell^i, \bar{\zeta}_m)}_{\substack{i^{\text{th}}\ N-\text{dimensional} \\ \text{component}}}, \ldots, \mathbf{0}_N^\top]\,, \qquad \text{with } \bar{\zeta}_m \in \mathcal{Z}_\ell^{f_i}\,, \\
\rho_{\ell,m}^i &= [\mathbf{0}_N^\top, \ldots, \underbrace{\nabla\gamma_i(\hat{x}_\ell^i, \bar{\zeta}_m)}_{\substack{i^{\text{th}}\ N-\text{dimensional} \\ \text{component}}}, \ldots, \mathbf{0}_N^\top]\,, \qquad \text{with } \bar{\zeta}_m \in \mathcal{Z}_\ell^{g_i}\,.
\end{aligned} \tag{6.19}$$

Note that the subdifferentials $\partial f_i(\hat{x}_\ell^i)$ and $\partial g_i(\hat{x}_\ell^i)$ are taken with respect to $\hat{\boldsymbol{x}}$, whereas the gradients $\nabla\phi_i(\hat{x}_\ell^i, \bar{\zeta}_m)$ and $\nabla\gamma_i(\hat{x}_\ell^i, \bar{\zeta}_m)$ are taken with respect to $\hat{x}^i$.

Next, we define the linearizations of f_i and g_i at $\hat{x}_\ell^i$ as

$$\bar{f}_{i,\ell}(\hat{x}^i) := f_i(\hat{x}_\ell^i) + f_i^\circ(\hat{x}_\ell^i\,; \hat{x}^i - \hat{x}_\ell^i)\,, \tag{6.20a}$$
$$\bar{g}_{i,\ell}(\hat{x}^i) := g_i(\hat{x}_\ell^i) + g_i^\circ(\hat{x}_\ell^i\,; \hat{x}^i - \hat{x}_\ell^i)\,, \tag{6.20b}$$

where, due to the specific structure of the max-functions f_i and g_i and using Corollary 1, the generalized directional derivatives can be found as

$$f_i^\circ(\hat{x}_\ell^i\,; \hat{x}^i - \hat{x}_\ell^i) = \max_{m \in \mathcal{I}_\ell^{f_i}} \langle\, \xi_{\ell,m}^i\,,\, \hat{x}^i - \hat{x}_\ell^i \,\rangle\,,$$
$$g_i^\circ(\hat{x}_\ell^i\,; \hat{x}^i - \hat{x}_\ell^i) = \max_{m \in \mathcal{I}_\ell^{g_i}} \langle\, \rho_{\ell,m}^i\,,\, \hat{x}^i - \hat{x}_\ell^i \,\rangle\,.$$

Let us define the maps $\alpha^i_{\ell,m} : \mathbb{R}^N \to \mathbb{R}$ and $\beta^i_{\ell,m} : \mathbb{R}^N \to \mathbb{R}$ as

$$\alpha^i_{\ell,m}(\hat{x}^i) = f_i(\hat{x}^i) - f_i(\hat{x}^i_\ell) - \langle\, \xi^i_{\ell,m}, \hat{x}^i - \hat{x}^i_\ell \,\rangle\,, \tag{6.21a}$$

$$\beta^i_{\ell,m}(\hat{x}^i) = g_i(\hat{x}^i) - g_i(\hat{x}^i_\ell) - \langle\, \rho^i_{\ell,m}, \hat{x}^i - \hat{x}^i_\ell \,\rangle\,, \tag{6.21b}$$

and denote the *linearization errors* at $\hat{x}^i$ for the $\xi^i_{\ell,m}$-and $\rho^i_{\ell,m}$-linearization of f_i and g_i, respectively. They will be discussed in more detail in Section 6.2.5. Using these linearization errors, we can rewrite Equations (6.20) as

$$\bar{f}_{i,\ell}(\hat{x}^i) = \max_{m \in \mathcal{I}^{f_i}_\ell} \{ f_i(\hat{x}^i_k) + \langle\, \xi^i_{\ell,m}, \hat{x}^i - \hat{x}^i_k \,\rangle - \alpha^i_{\ell,m}(\hat{x}^i_k) \}\,,$$

$$\bar{g}_{i,\ell}(\hat{x}^i) = \max_{m \in \mathcal{I}^{g_i}_\ell} \{ g_i(\hat{x}^i_k) + \langle\, \rho^i_{\ell,m}, \hat{x}^i - \hat{x}^i_k \,\rangle - \beta^i_{\ell,m}(\hat{x}^i_k) \}\,.$$

Finally, we construct the piecewise linear approximations at the kth iteration as

$$\hat{f}_{i,k}(\hat{x}^i) := \max_{\ell \in \mathcal{I}_k} \bar{f}_{i,\ell}(\hat{x}^i)\,, \tag{6.23a}$$

$$\hat{g}_{i,k}(\hat{x}^i) := \max_{\ell \in \mathcal{I}_k} \bar{g}_{i,\ell}(\hat{x}^i)\,, \tag{6.23b}$$

and, similarly to Equation (6.17), define the *polyhedral approximation* of the improvement function $H(\hat{\boldsymbol{x}}; \hat{\boldsymbol{y}})$ as

$$\hat{H}_k(\hat{\boldsymbol{x}}) := \max_{i=1,\dots,N} \{\, \hat{f}_{i,k}(\hat{x}^i) - f_i(\hat{x}^i_k)\,,\, \hat{g}_{i,k}(\hat{x}^i) \,\}\,, \qquad \text{for all } \hat{\boldsymbol{x}} \in \mathbb{R}^{N^2}\,. \tag{6.24}$$

The following two lemmas summarize some properties of the piecewise linear approximations $\hat{f}_{i,k}(\hat{x}^i)$, $\hat{g}_{i,k}(\hat{x}^i)$, and $\hat{H}_k(\hat{\boldsymbol{x}})$ at the kth iteration.

Lemma 4 *Let the functions $f_i(\hat{x}^i)$ and $g_i(\hat{x}^i)$ be convex, and the piecewise linear approximations $\hat{f}_{i,k}(\hat{x}^i)$ and $\hat{g}_{i,k}(\hat{x}^i)$ at the kth iteration be defined as in Equation (6.23). Then*

$$\hat{f}_{i,k}(\hat{x}^i_\ell) = f_i(\hat{x}^i_\ell)\,,$$

$$\hat{g}_{i,k}(\hat{x}^i_\ell) = g_i(\hat{x}^i_\ell)\,,$$

for each $i = 1, \dots, N$ and all $\ell \in \mathcal{I}_k$.

Proof: The proof is given in Appendix B.8. □

Lemma 5 *Let the piecewise linear approximation $\hat{H}_k(\hat{\boldsymbol{x}})$ at $\hat{\boldsymbol{x}}_k$ be defined as in Equation (6.24). Then the*

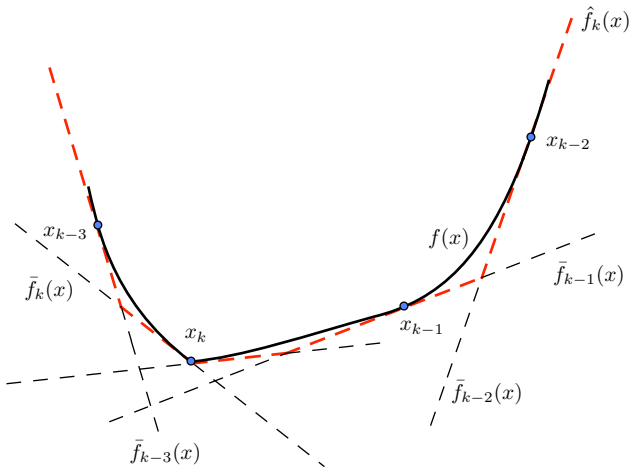


Figure 6.2: Cutting-plane model of a convex function $f : \mathbb{R} \to \mathbb{R}$ using the whole subdifferential $\partial_c f$ at each x_ℓ. The black solid line is the function $f(x)$, the black-dashed lines are the piecewise linear approximations $\bar{f}_\ell(x)$ at x_ℓ with $\ell \in \mathcal{I}_k$, and the red-dashed line is the piecewise linear approximation $\hat{f}_k(x)$ at the kth iteration.

function $\hat{H}_k(\hat{\boldsymbol{x}})$ is convex. If additionally, the functions f_i and g_i are convex, then the following holds for the subdifferential $\partial\hat{H}_k$ at $\hat{\boldsymbol{x}}_k$

$$\partial\hat{H}_k(\hat{\boldsymbol{x}}_k) = \partial H(\hat{\boldsymbol{x}}_k; \hat{\boldsymbol{x}}_k)\,.$$

Proof: The proof is given in Appendix B.9. □

To understand the difference in constructing the approximations between our approach and the bundle method, as described in Section 5.3.2, we show in Figure 6.2 the piecewise linear approximation $\hat{f}_k(x)$ for the same convex function $f : \mathbb{R} \to \mathbb{R}$ from Figure 5.3. At the current iteration point x_k, the function f is non-differentiable. In contrast to the bundle method, we use the whole subdifferential $\partial_c f(x_k)$ to construct the piecewise linear approximation $\hat{f}_k(x)$, that is shown by the red-dashed line. It is apparent, that in Figure 6.2 the approximation lies closer to the function $f(x)$ (black solid line), especially near the neighborhood of x_k, where $f(x)$ is non-differentiable. Hence, a clear benefit over the classical bundle method is that our approach obtains a more accurate approximation.

The next theorem shows that if the nonsmooth functions f_i and g_i are convex, then a feasible descent direction for F can be found through the piecewise linear approximation $\hat{H}_k$.

Theorem 19 *Consider the constrained optimization problem (P2). Let the functions f_i and g_i be convex. If the direction $\boldsymbol{d}_k \in \mathbb{R}^{N^2}$ is a descent direction for $\hat{H}_k$ at $\hat{\boldsymbol{x}}_k \in \Omega$ and feasible subject to $\Omega_{\tilde{L}}$, then $\boldsymbol{d}_k$ is a descent direction for F at $\hat{\boldsymbol{x}}_k$ and feasible subject to Ω.*

Proof: The proof is given in Appendix B.10. □

An important consequence of Theorem 19, is the fact that we are able to avoid the line search procedure,

since a feasible descent direction $\boldsymbol{d}_k$ of $\hat{H}_k$ is also a feasible descent direction for F at $\hat{\boldsymbol{x}}_k$. This claim could not be made for bundle methods, as by using an arbitrary subgradient of the subdifferential, the descent direction found for the cutting plane model, may not even be a descent direction for the objective function. The line search procedure was used to overcome this issue. To see why a line search procedure is undesirable for a distributed approach, recall that the descent direction $\boldsymbol{d}_k$ is defined as $\boldsymbol{d}_k := [{d_k^1}^\top, \dots, {d_k^N}^\top]^\top$, with d_k^i being the descent direction for vehicle i. If each vehicle i ran a local line search procedure and obtained a step size τ_i, then the resulting descent direction $\tilde{\boldsymbol{d}}_k := [{\tau_1 d_k^1}^\top, \dots, {\tau_N d_k^N}^\top]^\top$ would no longer be the desired descent direction $\boldsymbol{d}_k$. Hence, a common and fixed step size τ is preferable, which, however, would render the classical bundle method non-convergent.

Note the difference between Theorem 17 and 19. Theorem 17 considers the local neighborhood of $\hat{\boldsymbol{x}}_k$, while Theorem 19 takes into account the local behavior of the nonsmooth functions at an increasing number of points $\hat{\boldsymbol{x}}_\ell$ for $\ell \in \mathcal{I}_k$. In other words, more and more information regarding the nonsmooth functions is accumulated as the set $\mathcal{I}_k$ expands. However, Theorem 19 requires the functions f_i and g_i to be convex, so that $\hat{f}_{i,k}(\hat{x}_k^i) = f_i(\hat{x}_k^i)$ and $\hat{g}_{i,k}(\hat{x}_k^i) = g_i(\hat{x}_k^i)$, respectively. This stems from the fact that for a convex function f we have $\hat{f}(x) \leq f(x)$ for all x. This property no longer holds for a nonconvex function f since we may, for example, have that $\hat{f}(x) > f(x)$. In [48] the notion of *subgradient locality measures* is introduced to extend the results to nonconvex functions. We shall see that we adopt a similar approach; however, for the sake of simplicity, in the following we assume that the functions f_i and g_i are convex and will revisit the issue with nonconvex functions in Section 6.2.5.

The direction finding problem can now be formulated using $\hat{H}_k(\hat{\boldsymbol{x}})$ as defined in Equation (6.24), since a feasible descent direction $\boldsymbol{d}_k$ of $\hat{H}_k$ at $\hat{\boldsymbol{x}}_k$, is a descent direction of F at $\hat{\boldsymbol{x}}_k$ and feasible subject to Ω. To this end, consider the following direction finding problem:

$$\begin{aligned} \text{(DFP1)} \quad \underset{\boldsymbol{d}_k}{\text{minimize}} \quad & \hat{H}_k(\hat{\boldsymbol{x}}_k + \boldsymbol{d}_k) + \frac{1}{2}\|\boldsymbol{d}_k\|^2 \\ \text{subject to} \quad & \tilde{L}(\hat{\boldsymbol{x}}_k + \boldsymbol{d}_k) = 0\,. \end{aligned}$$

Notice that $\tilde{L}\,\hat{\boldsymbol{x}}_k = 0$, since $\hat{\boldsymbol{x}}_k$ is a feasible point and, therefore, the equality constraint reduces to $\tilde{L}\,\boldsymbol{d}_k = 0$. Problem (DFP1) has a unique solution $\boldsymbol{d}_k$, since from Lemma 5 we know that the piecewise linear approximation $\hat{H}_k(\hat{\boldsymbol{x}})$ is convex, and thus $\hat{H}_k(\hat{\boldsymbol{x}}_k + \boldsymbol{d}_k) + \frac{1}{2}\|\boldsymbol{d}_k\|^2$ strictly convex, and the fact that the equality constraints are linear. Then, along the same lines as shown for Problem (6.18), we can conclude that a solution $\boldsymbol{d}_k$ to Problem (DFP1) is indeed a descent direction for $\hat{H}_k$ subject to $\Omega_{\tilde{L}}$.

The following theorem gives the necessary condition for Problem (DFP1) to attain a global minimizer at $\boldsymbol{d}_k^*$.

Theorem 20 *Let $\boldsymbol{d}_k^*$ be a global minimizer of (DFP1). Then there exists a vector $\nu \in \mathbb{R}^{N^2}$ such that*

$$0 \in \partial \hat{H}_k(\hat{\boldsymbol{x}}_k + \boldsymbol{d}_k^*) + \boldsymbol{d}_k^* + \tilde{L}^\top \nu \,.$$

The theorem can be proved along the same lines as Theorem 16, by using the fact that $\partial(\frac{1}{2}\|\boldsymbol{d}_k^*\|^2) = \boldsymbol{d}_k^*$.

The optimization problem (DFP1) is still nonsmooth since the function $\hat{H}_k(\hat{\boldsymbol{x}})$ still involves maximum functions:

$$\begin{aligned} \underset{\boldsymbol{d}_k}{\text{minimize}} \quad & \max_{i=1,\ldots,N} \{\, \hat{f}_{i,k}(\hat{x}_k^i + d_k^i) - f_i(\hat{x}_k^i)\,,\, \hat{g}_{i,k}(\hat{x}_k^i + d_k^i) \,\} + \frac{1}{2}\|\boldsymbol{d}_k\|^2 \\ \text{subject to} \quad & \tilde{L}\,\boldsymbol{d}_k = 0\,. \end{aligned}$$

We can transform this optimization problem into a smooth and linearly constrained optimization problem by introducing an *auxiliary* variable z_k, defined as:

$$z_k := \max_{i=1,\ldots,N} \{\, \hat{f}_{i,k}(\hat{x}_k^i + d_k^i) - f_i(\hat{x}_k^i)\,,\, \hat{g}_{i,k}(\hat{x}_k^i + d_k^i) \,\}\,.$$

The optimization problem (DFP1) is then equivalent to:

$$\begin{aligned} \text{(DFP2)} \quad \underset{(z_k,\boldsymbol{d}_k)}{\text{minimize}} \quad & z_k + \frac{1}{2}\|\boldsymbol{d}_k\|^2 \\ \text{subject to} \quad & -\alpha_{\ell,m}^i(\hat{x}_k^i) + \langle\, \xi_{\ell,m}^i, d_k^i \,\rangle - z_k \le 0\,, && \forall\ i = 1,\ldots,N\,,\ \ell \in \mathcal{I}_k\,,\ m \in \mathcal{I}_\ell^{f_i}\,, \\ & g_i(\hat{x}_k^i) - \beta_{\ell,m}^i(\hat{x}_k^i) + \langle\, \rho_{\ell,m}^i, d_k^i \,\rangle - z_k \le 0\,, && \forall\ i = 1,\ldots,N\,,\ \ell \in \mathcal{I}_k\,,\ m \in \mathcal{I}_\ell^{g_i}\,, \\ & \tilde{L}\,\boldsymbol{d}_k = 0\,, \end{aligned}$$

where the maps $\alpha_{\ell,m}^i(\hat{x}^i)$, and $\beta_{\ell,m}^i(\hat{x}^i)$ were defined by Equation (6.21). However, since this auxiliary variable z_k appears in all the linear inequality constraints, it also introduces coupling of these constraints. Hence, Problem (DFP2) cannot be solved in a distributed way. Instead, we define the vector $\boldsymbol{z}_k := [z_{1,k}, \ldots, z_{N,k}]^\top$ and formulate the following optimization problem:

$$\begin{aligned} \text{(DFP3)} \quad \underset{(\boldsymbol{z}_k,\boldsymbol{d}_k)}{\text{minimize}} \quad & \tfrac{1}{N}\sum_{i=1}^N z_{i,k} + \tfrac{1}{2}\|\boldsymbol{d}_k\|^2 \\ \text{subject to} \quad & -\alpha_{\ell,m}^i(\hat{x}_k^i) + \langle\, \xi_{\ell,m}^i, d_k^i \,\rangle - z_{i,k} \le 0\,, && \forall\ i = 1,\ldots,N\,,\ \ell \in \mathcal{I}_k\,,\ m \in \mathcal{I}_\ell^{f_i}\,, \\ & g_i(\hat{x}_k^i) - \beta_{\ell,m}^i(\hat{x}_k^i) + \langle\, \rho_{\ell,m}^i, d_k^i \,\rangle - z_{i,k} \le 0\,, && \forall\ i = 1,\ldots,N\,,\ \ell \in \mathcal{I}_k\,,\ m \in \mathcal{I}_\ell^{g_i}\,, \\ & \tilde{L}\,\boldsymbol{d}_k = 0\,, \\ & L\boldsymbol{z}_k = 0\,. \end{aligned}$$

Proposition 5 *$(z_k^*, \boldsymbol{d}_k^*)$ is the global minimizer of (DFP2) if and only if $(\boldsymbol{z}_k^*, \boldsymbol{d}_k^*)$ is the global minimizer of (DFP3), where $\boldsymbol{z}_k^* := z_k^* \mathbb{1}_N$.*

Proof: The proof is given in Appendix B.11. □

In order to ease the notation in what follows, we first define a collection of vectors. Let ϑ be, as we shall see subsequently, the vector of multipliers for Problem (DFP3) and defined as

$$\vartheta := [\vartheta_1^\top, \ldots, \vartheta_N^\top]^\top .$$

Each ϑ_i corresponds to the vector of multipliers for vehicle i and is given by

$$\vartheta_i := [\lambda^{i\top}, \mu^{i\top}, \nu_i^\top, \eta_i]^\top , \qquad \text{for } i = 1, \ldots, N ,$$

where

$$\begin{aligned}
\lambda^i &:= [\lambda_1^{i\top}, \ldots, \lambda_k^{i\top}]^\top , \\
\lambda_\ell^i &:= [\lambda_{\ell,1}^i, \ldots, \lambda_{\ell,n_{\zeta,\ell}^{f_i}}^i]^\top , \\
\mu^i &:= [\mu_1^{i\top}, \ldots, \mu_k^{i\top}]^\top , \\
\mu_\ell^i &:= [\mu_{\ell,1}^i, \ldots, \mu_{\ell,n_{\zeta,\ell}^{g_i}}^i]^\top , \\
\nu &:= [\nu_1^\top, \ldots, \nu_N^\top]^\top , \\
\nu_i &:= [\nu_{i,1}, \ldots, \nu_{i,N}]^\top , \\
\eta &:= [\eta_1, \ldots, \eta_N]^\top ,
\end{aligned}$$

with $\lambda_{\ell,m}^i, \mu_{\ell,m}^i, \nu_{i,q}, \eta_i \in \mathbb{R}$. Similarly, we define the vector-valued functions $\alpha_\ell^i(\hat{x}^i)$ and $\beta_\ell^i(\hat{x}^i)$ as

$$\alpha_\ell^i(\hat{x}^i) := \begin{bmatrix} \alpha_{\ell,1}^i(\hat{x}^i) \\ \vdots \\ \alpha_{\ell,n_{\zeta,\ell}^{f_i}}^i(\hat{x}^i) \end{bmatrix} , \qquad \beta_\ell^i(\hat{x}^i) := \begin{bmatrix} \beta_{\ell,1}^i(\hat{x}^i) \\ \vdots \\ \beta_{\ell,n_{\zeta,\ell}^{g_i}}^i(\hat{x}^i) \end{bmatrix} ,$$

where $\alpha_{\ell,m}^i(\hat{x}^i)$ and $\beta_{\ell,m}^i(\hat{x}^i)$ were given in Equation (6.21). Then, using the above notations, the dual

function $\Lambda(\vartheta; \hat{\boldsymbol{x}}_k)$ for (DFP3) is given by

$$\begin{aligned}\Lambda(\vartheta; \hat{\boldsymbol{x}}_k) = \min_{(\boldsymbol{z}_k, \boldsymbol{d}_k)} \tfrac{1}{N} \sum_{i=1}^{N} z_{i,k} + \tfrac{1}{2}\|\boldsymbol{d}_k\|^2 + \sum_{i=1}^{N} \sum_{\ell=1}^{k} \sum_{m \in \mathcal{I}_\ell^{f_i}} \lambda_{\ell,m}^i \left(-\alpha_{\ell,m}^i(\hat{x}_k^i) + \langle\, \xi_{\ell,m}^i, d_k^i \,\rangle - z_{i,k} \right) \\ + \sum_{i=1}^{N} \sum_{\ell=1}^{k} \sum_{m \in \mathcal{I}_\ell^{g_i}} \mu_{\ell,m}^i \left(g_i(\hat{x}_k^i) - \beta_{\ell,m}^i(\hat{x}_k^i) + \langle\, \rho_{\ell,m}^i, d_k^i \,\rangle - z_{i,k} \right) + \nu^\top \tilde{L}\, \boldsymbol{d}_k + \eta^\top L\, \boldsymbol{z}_k \,,\end{aligned} \tag{6.25}$$

and note that $\lambda_{\ell,m}^i \geq 0$ and $\mu_{\ell,m}^i \geq 0$ must hold, since they are the multipliers for the inequality constraints of (DFP3). Then the necessary conditions are

$$h_i(\vartheta) = 0\,, \qquad \text{for} \quad i = 1, \ldots, N \tag{6.26}$$

and

$$\boldsymbol{d}_k + \sum_{i=1}^{N} \sum_{\ell=1}^{k} \sum_{m \in \mathcal{I}_\ell^{f_i}} \lambda_{\ell,m}^i\, \xi_{\ell,m}^{i\top} + \sum_{i=1}^{N} \sum_{\ell=1}^{k} \sum_{m \in \mathcal{I}_\ell^{g_i}} \mu_{\ell,m}^i\, \rho_{\ell,m}^{i\top} + \tilde{L}^\top \nu = 0\,, \tag{6.27}$$

where

$$h_i(\vartheta) := \frac{1}{N} - \sum_{\ell=1}^{k} \sum_{m \in \mathcal{I}_\ell^{f_i}} \lambda_{\ell,m}^i - \sum_{\ell=1}^{k} \sum_{m \in \mathcal{I}_\ell^{g_i}} \mu_{\ell,m}^i + \sum_{j \in \mathcal{N}_i} (l_{ij}\eta_i - l_{ji}\eta_j)\,.$$

By substituting Equations (6.26) and (6.27) into Equation (6.25), we obtain for the dual function

$$\begin{aligned}\Lambda(\vartheta; \hat{\boldsymbol{x}}_k) = -\tfrac{1}{2}\| \sum_{i=1}^{N} \sum_{\ell=1}^{k} \sum_{m \in \mathcal{I}_\ell^{f_i}} \lambda_{\ell,m}^i\, \xi_{\ell,m}^{i\top} + \sum_{i=1}^{N} \sum_{\ell=1}^{k} \sum_{m \in \mathcal{I}_\ell^{g_i}} \mu_{\ell,m}^i\, \rho_{\ell,m}^{i\top} + \tilde{L}^\top \nu \|^2 \\ - \sum_{i=1}^{N} \sum_{\ell=1}^{k} \sum_{m \in \mathcal{I}_\ell^{f_i}} \lambda_{\ell,m}^i \alpha_{\ell,m}^i(\hat{x}_k^i) - \sum_{i=1}^{N} \sum_{\ell=1}^{k} \sum_{m \in \mathcal{I}_\ell^{g_i}} \mu_{\ell,m}^i \left(\beta_{\ell,m}^i(\hat{x}_k^i) - g_i(\hat{x}_k^i) \right)\,.\end{aligned}$$

The dual problem of (DFP3) can be formulated as follows:

$$\begin{aligned} \underset{\vartheta}{\text{maximize}} \quad & \Lambda(\vartheta; \hat{\boldsymbol{x}}_k) \\ \text{subject to} \quad & h_i(\vartheta) = 0\,, \\ & \lambda^i \geq 0\,,\ \mu^i \geq 0\,, \end{aligned}$$

for $i = 1, \ldots, N$. Hence, we can find the multipliers ϑ by solving the following optimization problem:

$$\begin{aligned}
\text{(DFP4)} \quad \underset{\vartheta}{\text{minimize}} \quad & \tfrac{1}{2} \| \sum_{i=1}^{N} \sum_{\ell=1}^{k} \sum_{m \in \mathcal{I}_\ell^{f_i}} \lambda_{\ell,m}^i \, \xi_{\ell,m}^{i\top} + \sum_{i=1}^{N} \sum_{\ell=1}^{k} \sum_{m \in \mathcal{I}_\ell^{g_i}} \mu_{\ell,m}^i \, \rho_{\ell,m}^{i\top} + \tilde{L}^\top \nu \|^2 \\
& + \sum_{i=1}^{N} \sum_{\ell=1}^{k} \sum_{m \in \mathcal{I}_\ell^{f_i}} \lambda_{\ell,m}^i \alpha_{\ell,m}^i(\hat{x}_k^i) + \sum_{i=1}^{N} \sum_{\ell=1}^{k} \sum_{m \in \mathcal{I}_\ell^{g_i}} \mu_{\ell,m}^i \left(\beta_{\ell,m}^i(\hat{x}_k^i) - g_i(\hat{x}_k^i) \right) \\
\text{subject to} \quad & h_i(\vartheta) = 0 \,, \\
& \lambda^i \geq 0 \,, \, \mu^i \geq 0 \,.
\end{aligned}$$

Since (DFP4) is the dual problem to the direction finding problem (DFP3), the (optimal) descent direction $\boldsymbol{d}_k$ at the kth iteration can be obtained from the optimal solution of Problem (DFP4). Let ϑ^* be the optimal solution of (DFP4) and consider Equation (6.27). Then the descent direction $\boldsymbol{d}_k$ can be found as:

$$\boldsymbol{d}_k = - \sum_{i=1}^{N} \sum_{\ell=1}^{k} \sum_{m \in \mathcal{I}_\ell^{f_i}} \lambda_{\ell,m}^{*i} \, \xi_{\ell,m}^{i\top} - \sum_{i=1}^{N} \sum_{\ell=1}^{k} \sum_{m \in \mathcal{I}_\ell^{g_i}} \mu_{\ell,m}^{*i} \, \rho_{\ell,m}^{i\top} - \tilde{L}^\top \nu^* \,. \tag{6.28}$$

Due to the constraint $\tilde{L} \, \boldsymbol{d}_k = 0$, the descent direction $\boldsymbol{d}_k$ must be of the form $\boldsymbol{d}_k = \mathbb{1}_N \otimes d_k$ with $d_k \in \mathbb{R}^N$. In other words, the descent direction d_k^i for each vehicle is the same for all vehicles.

The direction finding problem (DFP4) is also formulated such that it allows the vehicles to compute the descent direction $\boldsymbol{d}_k$ in a distributed way. The next section presents the proposed distributed algorithm that can be implemented on each individual vehicle.

Remark 5 *In the next section, where the proposed distributed algorithm is presented, it will be shown that the multipliers λ^i and μ^i are computed onboard vehicle i. Hence, the simple bounds on these multipliers, i.e. $\lambda^i \geq 0$ and $\mu^i \geq 0$, can be implemented* locally *on vehicle i. Several approaches are available such as the gradient projection method [54], or by converting the simple bounds into equality constraints by introducing additional auxiliary variables [7]. Nevertheless, these isolated parts of the implementation will not affect the development of the distributed algorithm, other than cluttering the discussion with extra variables and/or multipliers. Hence, in the following they are omitted with the understanding that it must be ensured that the bounds on the mulitpliers λ_i and μ^i still hold.*

6.2.4 Distributed Algorithm

Before we proceed with the derivation of the distributed algorithm, let us first define the vector $\boldsymbol{p}$ as a function of ϑ:

$$\boldsymbol{p}(\vartheta) := \sum_{i=1}^{N}\sum_{\ell=1}^{k}\sum_{m\in\mathcal{I}_\ell^{f_i}} \lambda_{\ell,m}^i\, \xi_{\ell,m}^{i\top} + \sum_{i=1}^{N}\sum_{\ell=1}^{k}\sum_{m\in\mathcal{I}_\ell^{g_i}} \mu_{\ell,m}^i\, \rho_{\ell,m}^{i\top} + \tilde{L}^\top \nu\,.$$

Notice that by specific structure of the subgradients $\xi_{\ell,m}^i$ and $\rho_{\ell,m}^i$, we can write $\boldsymbol{p}(\vartheta)$ as

$$\begin{aligned}\boldsymbol{p}(\vartheta) &= [p_1^\top(\vartheta),\ldots,p_N^\top(\vartheta)]^\top\,,\\ p_i(\vartheta) &= \sum_{\ell=1}^{k}\Phi_\ell^i\lambda_\ell^i + \sum_{\ell=1}^{k}\Gamma_\ell^i\mu_\ell^i + \sum_{j\in\mathcal{N}_i}(l_{ij}\nu_i - l_{ji}\nu_j)\,,\end{aligned}$$

where the matrices Φ_ℓ^i and Γ_ℓ^i are defined as

$$\begin{aligned}\Phi_\ell^i &:= \left[\nabla\phi_i^\top(\hat{x}_\ell^i,\bar{\zeta}_1)\ldots\nabla\phi_i^\top(\hat{x}_\ell^i,\bar{\zeta}_{n_{\zeta,\ell}^{f_i}})\right]\,, \qquad \text{with}\ \ \bar{\zeta}_m\in\mathcal{Z}_\ell^{f_i}\,,\\ \Gamma_\ell^i &:= \left[\nabla\gamma_i^\top(\hat{x}_\ell^i,\bar{\zeta}_1)\ldots\nabla\gamma_i^\top(\hat{x}_\ell^i,\bar{\zeta}_{n_{\zeta,\ell}^{g_i}})\right]\,, \qquad \text{with}\ \ \bar{\zeta}_m\in\mathcal{Z}_\ell^{g_i}\,.\end{aligned}$$

Then the cost function $J(\vartheta;\hat{\boldsymbol{x}}_k)$ of (DFP4) can be written as:

$$J(\vartheta;\hat{\boldsymbol{x}}_k) = \tfrac{1}{2}\|\boldsymbol{p}(\vartheta)\|^2 + \sum_{i=1}^{N}\sum_{\ell=1}^{k}\sum_{m\in\mathcal{I}_\ell^{f_i}}\lambda_{\ell,m}^i\alpha_{\ell,m}^i(\hat{x}_k^i) + \sum_{i=1}^{N}\sum_{\ell=1}^{k}\sum_{m\in\mathcal{I}_\ell^{g_i}}\mu_{\ell,m}^i\left(\beta_{\ell,m}^i(\hat{x}_k^i) - g_i(\hat{x}_k^i)\right)\,.$$

Since the (linearly) constrained quadratic direction finding problem (DFP4) is differentiable, we can derive the Karush-Kuhn-Tucker (KKT) conditions, and use a first-order method to solve these necessary conditions. Let the Lagrangian $l(\vartheta,\omega)$ for (DFP4) be defined as

$$l(\vartheta,\omega) := J(\vartheta;\hat{\boldsymbol{x}}_k) + \omega^\top h(\vartheta)\,,$$

where ω are the Lagrange multipliers for the equality constraints $h(\vartheta) := [h_1(\vartheta),\ldots,h_N(\vartheta)]^\top$. The KKT conditions for (DFP4) can be derived from the Lagrangian by setting $\nabla_\vartheta l(\vartheta,\omega) = 0$ and $\nabla_\omega l(\vartheta,\omega) = 0$. Hence, the KKT conditions are given by

$$\begin{aligned}\nabla J^\top(\vartheta;\hat{\boldsymbol{x}}_k) + \nabla h^\top(\vartheta)\,\omega &= 0\,,\\ h(\vartheta) &= 0\,.\end{aligned} \tag{6.29}$$

We can use a simple first-order Lagrangian method that is based on linear approximations of the primal-dual equations given in (6.29). In this regard, first-order methods are very similar to the steepest descent method.

For example the *first-order Lagrangian method* is extensively described in the literature [8, 54], and for our primal-dual equations we obtain the following algorithm

$$\begin{aligned} {}^{r+1}\vartheta &= {}^{r}\vartheta - \tau_d \left(\nabla J^\top(\vartheta; \hat{\boldsymbol{x}}_k) + \nabla h^\top(\vartheta)\, \omega \right) , \\ {}^{r+1}\omega &= {}^{r}\omega + \tau_d\, h(\vartheta) , \end{aligned} \tag{6.30}$$

for some sufficiently small step size $\tau_d > 0$ that is known and common to all vehicles, and where r denotes the iteration step for the direction finding problem at the current iteration point $\hat{\boldsymbol{x}}_k$. In order for Equation (6.30) to be implemented in a distributed fashion, it is necessary that the gradient $\nabla J(\vartheta; \hat{\boldsymbol{x}}_k)$ and Jacobian $\nabla h(\vartheta)$ exhibit a block structure and, moreover, such that each block can be evaluated onboard its corresponding vehicle using only partial knowledge of the entire fleet of vehicles. As discussed in Section 6.1, this decoupled block structure is obtained by introducing the local estimates $\hat{x}^i$. To see this, recall that $\vartheta_i = [\lambda^{i\top}, \mu^{i\top}, \nu_i^\top, \eta_i]^\top$ corresponds to the vector of multipliers associated with the equality and inequality constraints of vehicle i. Next, if we derive $\nabla_{\vartheta_i} J(\vartheta; \hat{\boldsymbol{x}}_k)$ and $\nabla_{\vartheta_i} h(\vartheta)$, we can find that

$$\nabla_{\lambda^i} J^\top(\vartheta; \hat{\boldsymbol{x}}_k) = \begin{bmatrix} \Phi_1^{i\top}\, p_i + \alpha_1^i(\hat{x}_k^i) \\ \vdots \\ \Phi_k^{i\top}\, p_i + \alpha_k^i(\hat{x}_k^i) \end{bmatrix} , \qquad \nabla_{\lambda^i} h^\top(\vartheta) = \begin{bmatrix} \mathbf{0}_{n_\zeta^{f_i}}, \ldots, \underbrace{-\mathbb{1}_{n_\zeta^{f_i}}}_{i^{\text{th}}\ \text{column}}, \ldots, \mathbf{0}_{n_\zeta^{f_i}} \end{bmatrix}$$

$$\nabla_{\mu^i} J^\top(\vartheta; \hat{\boldsymbol{x}}_k) = \begin{bmatrix} \Gamma_1^{i\top}\, p_i + \beta_1^i(\hat{x}_k^i) - g_i(\hat{x}_k^i)\mathbb{1}_{n_{\zeta,1}^{g_i}} \\ \vdots \\ \Gamma_k^{i\top}\, p_i + \beta_k^i(\hat{x}_k^i) - g_i(\hat{x}_k^i)\mathbb{1}_{n_{\zeta,k}^{g_i}} \end{bmatrix} , \qquad \nabla_{\mu^i} h^\top(\vartheta) = \begin{bmatrix} \mathbf{0}_{n_\zeta^{g_i}}, \ldots, \underbrace{-\mathbb{1}_{n_\zeta^{g_i}}}_{i^{\text{th}}\ \text{column}}, \ldots, \mathbf{0}_{n_\zeta^{g_i}} \end{bmatrix}$$

$$\nabla_{\nu_i} J^\top(\vartheta; \hat{\boldsymbol{x}}_k) = \sum_{j \in \mathcal{N}_i} l_{ij}(p_i - p_j) , \qquad \nabla_{\nu_i} h^\top(\vartheta) = \mathbf{0}_{N \times N} ,$$

$$\nabla_{\eta_i} J^\top(\vartheta; \hat{\boldsymbol{x}}_k) = 0 , \qquad \nabla_{\eta_i} h^\top(\vartheta) = \boldsymbol{e}_i^\top L ,$$

(6.31)

where $\boldsymbol{e}_i$ denotes the ith unit vector, $n_\zeta^{f_i} = \sum_{\ell=1}^{k} n_{\zeta,\ell}^{f_i}$, and $n_\zeta^{g_i} = \sum_{\ell=1}^{k} n_{\zeta,\ell}^{g_i}$. It is clear that the gradient $\nabla J(\vartheta; \hat{\boldsymbol{x}}_k)$ and Jacobian $\nabla h(\vartheta)$ naturally exhibit the desired block structure, due to the specific form the subgradients $\xi_{\ell,m}^i$ and $\rho_{\ell,m}^i$ take (see Equation (6.19)). Moreover, this decoupled block structure is maintained if linear methods are used to solve the primal-dual equations. On the contrary, any nonlinear method such as Newton's method, may introduce non-zero cross-coupling terms. Therefore, in view of the development of distributed algorithms, the first-order method as given in Equation (6.30) is preferred over nonlinear methods, despite its lower rate of convergence.

Using Equation (6.31), we obtain the following iteration for the ith component of ${}^r\vartheta$:

$$
\begin{aligned}
{}^{r+1}\lambda_\ell^i &= {}^r\lambda_\ell^i - \tau_d \left({\Phi_\ell^i}^\top \, {}^r p_i + \alpha_\ell^i(\hat{x}_k^i) - {}^r\omega_i \, \mathbb{1}_{n_{\zeta,\ell}^{f_i}} \right) , \qquad \forall \ell \in \mathcal{I}_k \, , \\
{}^{r+1}\mu_\ell^i &= {}^r\mu_\ell^i - \tau_d \left({\Gamma_\ell^i}^\top \, {}^r p_i + \beta_\ell^i(\hat{x}_k^i) - g_i(\hat{x}_k^i) \, \mathbb{1}_{n_{\zeta,\ell}^{g_i}} - {}^r\omega_i \, \mathbb{1}_{n_{\zeta,\ell}^{g_i}} \right) , \qquad \forall \ell \in \mathcal{I}_k \, , \\
{}^{r+1}\nu_i &= {}^r\nu_i - \tau_d \sum_{j \in \mathcal{N}_i} l_{ij} ({}^r p_i - {}^r p_j) \, , \\
{}^{r+1}\eta_i &= {}^r\eta_i - \tau_d \sum_{j \in \mathcal{N}_i} l_{ij} ({}^r\omega_i - {}^r\omega_j) \, , \\
{}^{r+1}\omega_i &= {}^r\omega_i + \tau_d \left(\tfrac{1}{N} - \sum_{\ell=1}^{k} {\mathbb{1}_{n_{\zeta,\ell}^{f_i}}}^\top {}^r\lambda_\ell^i - \sum_{\ell=1}^{k} {\mathbb{1}_{n_{\zeta,\ell}^{g_i}}}^\top {}^r\mu_\ell^i + \sum_{j \in \mathcal{N}_i} (l_{ij} {}^r\eta_i - l_{ji} {}^r\eta_j) \right) ,
\end{aligned}
\tag{6.32}
$$

where

$$
{}^r p_i = \sum_{\ell=1}^{k} \Phi_\ell^i \, {}^r\lambda_\ell^i + \sum_{\ell=1}^{k} \Gamma_\ell^i \, {}^r\mu_\ell^i + \sum_{j \in \mathcal{N}_i} (l_{ij} {}^r\nu_i - l_{ji} {}^r\nu_j) \, . \tag{6.33}
$$

Algorithm (6.32) corresponds to the *local* iteration to be executed by vehicle i, and uses only local information and information from its neighbors $j \in \mathcal{N}_i$, namely p_j, ν_j, η_j and ω_j. Nevertheless, the algorithm requires a *two-step* information exchange between vehicle i and its neighbors, which will be discussed in detail in Section 6.2.6. Since each vehicle i is executing part of the algorithm (6.30) according to (6.32), we have indeed derived a distributed formulation of a centralized algorithm for solving the direction finding problem (DFP4).

Theorem 12 states that if the initial values are close enough to a solution (ϑ^*, ω^*) of the primal-dual equations (6.30) and a sufficiently small step size τ_d is used, then the sequence $\{{}^r\vartheta, {}^r\omega\}$ converges to this solution. To ensure convergence, it is required that the Hessian of the Lagrangian $\nabla^2 l(\vartheta^*, \omega^*)$ is positive definite. A weaker assumption, but identical for the same class of first-order centralized algorithms, can be made if instead an augmented version of the Lagrangian is used. This extension is presented in [58, 59, 62].

After the algorithm has converged, i.e. $\nabla_{\vartheta_i} J(\vartheta; \hat{\boldsymbol{x}}_k) = 0$ for $i = 1, \ldots, N$, the (optimal) feasible descent direction d_k^i for vehicle i at the kth iteration can be determined from the multipliers ϑ_i^* using Equation (6.28). In fact, the descent direction can be directly found from p_i, since from Equations (6.28) and (6.33) it follows that

$$
d_k^i = -p_i^* \, , \tag{6.34}
$$

where $p_i^* = p_i(\vartheta_i^*)$. Note that $p_i(\vartheta)$ is computed onboard vehicle i at each iteration step r and, hence, the descent direction d_k^i can be easily determined using Equation (6.34) after the direction finding algorithm has converged.

Finally, using the descent direction d_k^i, vehicle i updates its local estimate $\hat{x}_k^i$ according to

$$\hat{x}_{k+1}^i = \hat{x}_k^i + \tau\, d_k^i$$

for some sufficiently small step size $\tau > 0$ that is known and common to all vehicles. From Theorem 19 we know that such a step size exists so that d_k^i is a feasible descent direction for $f_i(\hat{x}^i)$ for all $i = 1, \ldots, N$. Our stopping criterion at iteration k is defined as $\|d_k^i\|^2 = 0$. In practical applications, a parameter $\epsilon_s > 0$ specifies a desired accuracy and the stopping criterion becomes $\|d_k^i\|^2 < \epsilon_s$. The next result justifies our choice of stopping criterion.

Theorem 21 *Let Assumption 5 hold on the inequality constraints $G(\hat{\boldsymbol{x}}^*)$. Suppose $\|d_k^i\|^2 = 0$ at the kth iteration. If $\hat{\boldsymbol{x}}_k = \mathbb{1}_N \otimes x_k$ is a local minimizer to problem (P3), then x_k is a local minimizer to Problem (P1).*

Proof: The proof is given in Appendix B.12. □

Recall that due to the constraint $\tilde{L}\, \boldsymbol{d}_k = 0$, the descent directions d_k^i are in fact the same for all vehicles. Additionally, given that the starting point $\hat{\boldsymbol{x}}_1 \in \Omega$, we have

$$\hat{\boldsymbol{x}}_k + \tau\, \boldsymbol{d}_k \in \Omega\,, \qquad \text{for all } k = 1, 2, \ldots\,,$$

and, in particular, $\hat{\boldsymbol{x}}_k + \tau\, \boldsymbol{d}_k \in \Omega_{\tilde{L}}$. Hence, at every iteration k the vehicles agree on updated iteration points $\hat{x}_{k+1}^i$ such that these are equal for all vehicles. It is important to emphasize that the feasible point descent algorithm requires a feasible starting point $\hat{\boldsymbol{x}}_1$. Generally, finding a feasible starting point $\hat{\boldsymbol{x}}_1$ that satisfies the equality constraints is nontrivial. However, since the structure of the nullspace of $\tilde{L}$ is known a priori, a feasible starting point subject to $\tilde{L}\, \hat{\boldsymbol{x}}_1 = 0$ must be of the form $\mathbb{1}_N \otimes \hat{x}_1$, where $\hat{x}_1 \in \mathbb{R}^N$ is the common starting point for all vehicles.

6.2.5 Nonconvex Cost Functions and Constraints

In the above derivation of the distributed algorithm to solve Problem (P1), Theorem 19 plays an important role. It shows that a feasible descent direction for the piecewise linear approximation $\hat{H}_k(\hat{\boldsymbol{x}})$ of the improvement function $H(\hat{\boldsymbol{x}}; \hat{\boldsymbol{y}})$ at $\hat{\boldsymbol{x}}_k$, is a feasible descent direction for F at $\hat{\boldsymbol{x}}_k$. This result allows us to omit the line search procedure, commonly used in bundle methods (see Section 5.3.2). The theorem assumes that the locally Lipschitz continuous functions f_i and g_i are convex, so that the following holds for all $\hat{\boldsymbol{x}}_k$

$$\hat{f}_{i,k}(\hat{x}_k^i) = f_i(\hat{x}_k^i) \quad \text{and} \quad \hat{g}_{i,k}(\hat{x}_k^i) = g_i(\hat{x}_k^i)\,.$$

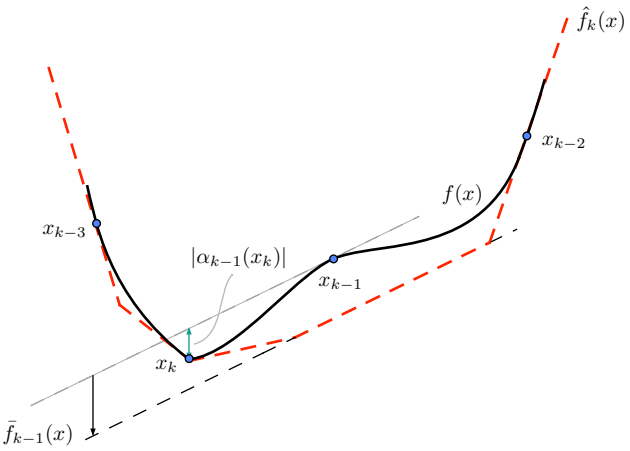


Figure 6.3: Cutting-plane model of a nonconvex function $f : \mathbb{R} \to \mathbb{R}$ using the whole subdifferential ∂f at each x_ℓ. The black solid line is the function $f(x)$, the black-dashed line is the linear approximation $\bar{f}_{k-1}(x)$ at x_{k-1}, and the red-dashed line is the piecewise linear approximation $\hat{f}_k(x)$ at the kth iteration.

For nonconvex functions this property may no longer hold. An example is shown in Figure 6.3. The locally Lipschitz continuous function $f : \mathbb{R} \to \mathbb{R}$ is nonconvex in the neighborhood of x_{k-1}. It is clear, that the linear approximation $\bar{f}_{k-1}(x)$ obtained at x_{k-1}, shown as a gray-dashed line, lies above $f(x)$ for $x = x_k$. Recall the functions $\alpha^i_{\ell,m}(x)$ and $\beta^i_{\ell,m}(x)$ as defined in Equation (6.21). For this example, we find for $\alpha_{k-1}(x)$ at x_k

$$\begin{aligned}\alpha_{k-1}(x_k) &= f(x_k) - f(x_{k-1}) - \langle\, \xi_{k-1}, x_k - x_{k-1} \,\rangle \\ &= f(x_k) - \bar{f}_{k-1}(x_k) \\ &< 0\,.\end{aligned}$$

In essence, the function $\alpha_{k-1}(x)$ denotes the *linearization error* of $\bar{f}_{k-1}(x)$ at x. For convex functions f, the linearization error $\alpha_\ell(x) \geq 0$ for all x, while this may not hold for nonconvex functions f. Hence, following the approach in [48], let us instead consider the following linear approximation $\bar{f}_{k-1}(x)$

$$\bar{f}_{k-1}(x) = f(x_k) + \langle\, \xi_{k-1}, x - x_k \,\rangle - |\alpha_{k-1}(x_k)|\,.$$

This modified linear approximation $\bar{f}_{k-1}(x)$ is shown with a black-dashed line in Figure 6.3. Effectively, the original linear approximation has been translated by an amount equal to $2\,|\alpha_{k-1}(x_k)|$, such that the approximation lies below the function f at x_k, i.e. $\bar{f}_{k-1}(x_k) < f(x_k)$. Note that the linear approximation $\bar{f}_{k-1}(x)$ does no longer go through $f(x_{k-1})$ and, hence, results in a poorer approximation of the function f at x_{k-1}.

It can be seen from Figure 6.3, that the effect a linear approximation $\bar{f}_\ell(x)$ has on the piecewise linear

approximation $\hat{f}_k(x)$ near x_k, depends on the value of $\alpha_\ell(x_k)$. The larger $\alpha_\ell(x_k)$ is, the lower the linear approximation $\bar{f}_\ell(x)$ lies below the function $f(x)$ and, hence, the less weight it carries near x_k in the construction of $\hat{f}_k(x)$. For convex functions, the linearization error $\alpha_\ell(x_k)$ naturally grows as the iteration point x_k moves away from the past iteration points x_ℓ, i.e. information collected farther away from x_k is less important near x_k. This is no longer true for nonconvex functions. And especially information collected at distant 'nonconvex parts' of the functions, we would like to ignore as much as possible, since those linearizations result in a poor approximation of the nonconvex function. To capture these effects, we use the distance measures as defined in [48].

Definition 12 *The distance measure at each iteration k for $\hat{x}^i_\ell$ with $\ell \in \mathcal{I}_k$ is given by*

$$s^i_{\ell,k} := \sum_{q=\ell}^{k-1} \|\hat{x}^i_{q+1} - \hat{x}^i_q\| \,.$$

Remark 6 *Recall that in the classical bundle method, so-called trial points y_j are used (see Section 5.3.2). Since our approach does not need trial points, Definition 12 is equivalent to the definition of the distance measures as given in [48], if the trial points y_j coincide with the iteration points x_j, i.e. only serious steps are taken.*

Finally, the subgradient locality measures were introduced in [48] to capture the effects of both negative linearization errors and obsolete past iteration points of nonconvex functions. Following the approach in [48], we define the subgradient locality measures $\tilde{\alpha}^i_{\ell,m}$ and $\tilde{\beta}^i_{\ell,m}$ for the nonconvex functions f_i and g_i, respectively as follows.

Definition 13 *The subgradient locality measures $\tilde{\alpha}^i_{\ell,m}$ and $\tilde{\beta}^i_{\ell,m}$ at each iteration k for $\hat{x}^i_\ell$ with $\ell \in \mathcal{I}_k$ are given by*

$$\tilde{\alpha}^i_{\ell,m}(\hat{x}^i_k) := \max\{|\alpha^i_{\ell,m}(\hat{x}^i_k)|, \delta^{f_i}(s^i_{\ell,k})^2\}\,,$$
$$\tilde{\beta}^i_{\ell,m}(\hat{x}^i_k) := \max\{|\beta^i_{\ell,m}(\hat{x}^i_k)|, \delta^{g_i}(s^i_{\ell,k})^2\}\,,$$

where $\delta^{f_i} \geq 0$ and $\delta^{g_i} \geq 0$ are the distance measure parameters ($\delta^{f_i} = 0$ if f_i is convex and $\delta^{g_i} = 0$ if g_i is convex).

To account for the nonconvexity effects, we substitute the linearization errors $\alpha^i_{\ell,m}(\hat{x}^i_k)$ and $\beta^i_{\ell,m}(\hat{x}^i_k)$ in our proposed distributed algorithm (6.32) with the subgradient locality measures $\tilde{\alpha}^i_{\ell,m}(\hat{x}^i_k)$ and $\tilde{\beta}^i_{\ell,m}(\hat{x}^i_k)$, respectively. Lastly, note that by using these subgradient locality measures, Lemma 5 and Theorem 19 hold

for nonconvex functions f_i and g_i, since we have for all $i = 1, \ldots, N$

$$\bar{f}_{i,\ell}(\hat{x}_k^i) \leq f_i(\hat{x}_k^i) \quad \text{and} \quad \bar{g}_{i,\ell}(\hat{x}_k^i) \leq g_i(\hat{x}_k^i)\,,$$

for all $\ell \in \mathcal{I}_k$ and, thus,

$$\hat{f}_{i,k}(\hat{x}_k^i) = f_i(\hat{x}_k^i) \quad \text{and} \quad \hat{g}_{i,k}(\hat{x}_k^i) = g_i(\hat{x}_k^i)\,,$$

due to the fact that $\bar{f}_{i,k}(\hat{x}_k^i) = f_i(\hat{x}_k^i)$ and $\bar{g}_{i,k}(\hat{x}_k^i) = g_i(\hat{x}_k^i)$.

6.2.6 Two-step Information Exchange

In order for vehicle i to update its own local variables using the distributed rules given in Equation (6.32), vehicle i requires information to be exchanged with its neighbors $j \in \mathcal{N}_i$. This requires a *two-step* information exchange protocol, in contrast to what is common practice for distributed (optimization) algorithms. This stems from the fact that the vector p_j with $j \in \mathcal{N}_i$ has to be exchanged so as to allow vehicle i to update its local multipliers ν_i. However, Equation (6.33) shows that each p_i of vehicle i depends on the multipliers ν_j of its neighbors. Therefore, the vector p_i can only be determined *after* the multipliers ν_i have been communicated over the network and received by the agents. For the distributed update rules given in Equation (6.32), we propose the following transmission protocol:

Time step r

The agents broadcast the vector ${}^r c_i$ over the communication network, defined as:

$${}^r c_i := \left[{}^r \nu_i^\top, {}^r \eta_i, {}^r \omega_i\right]^\top .$$

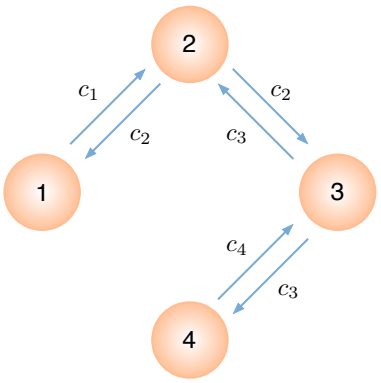
Upon receipt of ${}^r c_j$ of neighboring agents $j \in \mathcal{N}_i$, vehicle i is able to update the multipliers λ^i, μ^i, η_i, and ω_i according to the iteration scheme given in Equation (6.32). Moreover, at this time, vehicle i can also compute ${}^r c_i'$, given by:

$${}^r c_i' := {}^r p_i\,.$$

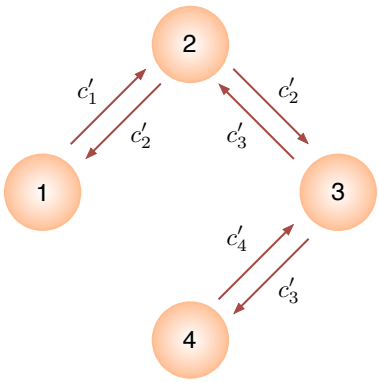
Time step r'

Iteration step r will be completed by transmitting the vector ${}^r c_i'$ over the network, and the remaining multiplier ν_i can be updated upon receipt of ${}^r c_j'$ from agents j in the neighborhood of vehicle i.

We illustrate the transmission protocol in Figure 6.4 for the example of four vehicles that was presented in Figure 6.1. The Laplacian L associated with the communication topology is given in Equation (6.3).



(a) Transmission at step r: exchange of $c_i = [\nu_i^\top, \eta_i, \omega_i]^\top$.



(b) Transmission at step r': exchange of $c'_i = p_i$.

Figure 6.4: Two-step communication model exchanging c_i and c'_i among agents.

Remark 7 *As the vehicles indirectly require information from vehicles in the network that do not share a direct communication link, a two-step information exchanged protocol is proposed. Although this is not common practice for existing distributed (optimization) algorithms, it is still a relaxed requirement compared to a communication topology that is complete, or a routing protocol that uses multi-hops so that an agent can send a package of information to any other agent in the network, even if they are not neighbors. The latter communication protocol is assumed, for example, in [97].*

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This dissertation consists of two parts that address the distributed cooperative trajectory-generation problem for multi-vehicle systems. The first part presented the formulation of a *centralized* cooperative trajectory-generation framework for generating a set of spatial trajectories for a team of cooperating vehicles that execute time-critical missions. In contrast to many existing path-planning methods, the proposed approach does not discretize the trajectories in space nor in time and, hence, guarantees safe collision-free trajectories where spatial separation of the trajectories is preferred over deconfliction in time. The cooperative trajectory-generation framework is formulated such that it also allows for a seamless integration with time-critical cooperative path-following algorithms of a larger cooperative control framework, that together ensure that the vehicles are able to execute collision-free maneuvers, while meeting the stringent spatial and temporal specifications of the mission.

In comparison with our earlier work, a new timing law is introduced and the dynamic constraints are expressed in terms of the trajectories by using the differential flatness property of the dynamic system. More specifically, by virtue of this differential flatness property, we were able to express the set of constraints in terms of Bézier and rational Bézier curves, through the use of Pythagorean Hodograph Bézier curves to describe the trajectories. As a result, the framework has the potential to compute these trajectories in (near) real-time, as computationally efficient algorithms are developed to evaluate the complex set of constraints. Consequently, the generated trajectories do not violate the dynamic constraints of the vehicles. In terms of deconfliction, it is shown that the framework is able to generate collision-free trajectories that are either spatially or temporally separated.

To show the efficacy of the proposed cooperative trajectory-generation framework, two different simulation examples of typical cooperative missions were presented. In each of the scenarios, a set of collision-free trajectories for a team of three UAVs was generated that does not violate the dynamics of the vehicles. It is demonstrated that by modifying the dynamic constraints accordingly, the trajectory-generation framework

can be used for different types of vehicles as both fixed-wing aircraft and multirotors were considered.

The first part of the dissertation focused on the formulation of the trajectory-generation framework and development of algorithms to handle the constraints. It was shown that the framework resulted in a nonconvex constrained optimization problem. In order to obtain preliminary results, standard (smooth) optimization software was used to solve the centralized optimization problem. The second part of the dissertation addressed the problem of distributing the trajectory generation over the individual vehicles in the network. It was shown, that the trajectory-generation framework is in fact a semi-infinite programming problem, which is inherently nonsmooth. Due to the formulation of the framework in terms of Bézier curves, the semi-infinite programming problem can be formulated as an ordinary (finite-dimensional) nonsmooth optimization problem. Moreover, compared to general nonsmooth functions, the use of Bézier curves allows to determine the *whole* subdifferential of the nonsmooth functions.

A distributed bundle method was proposed to solve the class of problems the trajectory-generation framework belongs to. The proposed distributed algorithm is based on existing algorithms for nonsmooth optimization, in particular the bundle methods, and distributed nonlinear programming methods. By exploiting the whole subdifferential, the proposed distributed bundle method avoids the complicated line search procedure, which is necessary for classical bundle methods. Additionally, the stopping criterion for terminating the algorithm is more intuitive and straightforward, compared to stopping criteria commonly employed in existing bundle methods.

7.2 Future Work

Cooperative Trajectory Generation: An open question that has not been answered yet for the proposed trajectory-generation framework is how to provide the optimization solver with a proper guess of the initial set of (feasible) curves. It was observed, that due to the highly nonlinear functions, the efficiency of the solvers was very sensitive to this initial guess, and scaling of the variables and objective function. Hence, obtaining a good initial guess is expected to decrease the computation time considerably.

Current on-going research efforts are on collision avoidance during mission-execution phase through trajectory re-planning, by exploiting the nice properties of Bézier curves that result in computationally efficient algorithms. However, it might be required during planning phase, to generate collision-free trajectories that avoid a priori known obstacles in the area of operations, for example, buildings in an urban area. Especially for very cluttered environments, quintic PH Bézier curves might not suffice due to the lack of degrees of freedom in shaping the curve. One way is to increase the degree of the PH Bézier curves, risking numeri-

cal instabilities associated with high degrees of polynomials. A more preferred way is to work with Bézier splines, i.e creating smooth Bézier curves from multiple individual (PH) Bézier curves. The research will be on extending the current framework to quintic PH Bézier splines by ensuring smoothness at the joints and formulating a timing law continuously applicable to all segments, while retaining the PH structure for the composite curve.

Distributed Nonsmooth Optimization: Immediate future research on the proposed distributed bundle method should analyze the convergence properties of the proposed algorithms. The emphasis of the work on the subject of distributed optimization that was presented in this dissertation, was more of exploratory kind, in search of a feasible approach to generate trajectories cooperatively in a distributed manner. Although descent methods and the distributed nonlinear programming method are shown to be individually convergent under certain assumptions, a more in-depth analysis is needed for the proposed combined approach.

The proposed approach did not take into consideration the issue of limited available computer memory onboard the vehicles. It is unlikely that an unlimited number of subdifferentials can be stored and, hence, this will cause serious memory storage problems. Also, the size of the direction finding problem will grow unboundedly and potentially result in an increase in computation time. Moreover, information of past iteration points that are far apart from the current iteration point, is most likely obsolete and could be discarded. The *subgradient aggregation strategy* presented in [48] offers a solution to aggregate the past subgradients and keep the number of constraints bounded in classical bundle methods. This strategy could be potentially used for our proposed distributed algorithm. However, recall that the vehicles may not have knowledge of the subset of iteration points that other vehicles in the network might have aggregated. Hence, research efforts should be on investigating the effects of storing information of different subsets of past iteration points by the vehicles and, if necessary, adapt the subgradient aggregation strategy to overcome this lack of information.

Proximal bundle methods employ a weight that captures the curvature of the nonsmooth function and is updated at each iteration. This modification improves the convergence rate compared to classical bundle methods. Along the same lines as before, this modification may be potentially useful if a work-around is found for the issue of lack of knowledge of the individual weights that are used by the vehicles during each iteration.

Last but certainly not least, the proposed distributed algorithm should be implemented so that it can be integrated as part of the distributed cooperative trajectory-generation framework as outlined in this dissertation.

Appendices

Appendix A

Quaternion Representation of Spatial PH Bézier Curves

This appendix is a summary of the results presented in [32], and the interested reader is referred to the aforementioned reference for a more in-depth discussion on the topic.

A quaternion $\mathcal{A} = a + a_x\hat{\mathbf{i}} + a_y\hat{\mathbf{j}} + a_z\hat{\mathbf{k}}$ comprises of a scalar part a and a vector part $\mathbf{a} = a_x\hat{\mathbf{i}} + a_y\hat{\mathbf{j}} + a_z\hat{\mathbf{k}}$, and its conjugate $\mathcal{A}^*$ is defined as $\mathcal{A}^* = a - a_x\hat{\mathbf{i}} - a_y\hat{\mathbf{j}} - a_z\hat{\mathbf{k}}$. We can also write the quaternion $\mathcal{A}$ as $\mathcal{A} = |\mathcal{A}|\,\mathcal{U}$, where $|\mathcal{A}|^2 = \mathcal{A}\mathcal{A}^* = a^2 + |\mathbf{a}|^2$ is the square of the magnitude of the quaternion $\mathcal{A}$, and $\mathcal{U}$ is the *unit* quaternion with $|\mathcal{U}| = 1$, defined as $\mathcal{U} = \cos\frac{1}{2}\delta + \sin\frac{1}{2}\delta\,\mathbf{n}$ for some angle δ and unit vector $\mathbf{n}$. Then, given a pure vector quaternion $\mathbf{v}$ and a unit quaternion $\mathcal{U}$, the quaternion product $\mathcal{U}\,\mathbf{v}\,\mathcal{U}^*$, represents a rotation of $\mathbf{v}$ through an angle δ about the unit vector $\mathbf{n}$, and results in a pure vector quaternion.

The first-order Hermite interpolation problem, concerned with the construction of a smooth curve matching given endpoints and derivatives, often yields equations of the following form when formulated in the quaternion representation:

$$\mathcal{A}\,\hat{\mathbf{i}}\,\mathcal{A}^* = \mathbf{c}\,, \tag{A.1}$$

with $\mathbf{c} = c_x\hat{\mathbf{i}} + c_y\hat{\mathbf{j}} + c_z\hat{\mathbf{k}}$. The *one-parameter family of general solutions* to (A.1) is given by:

$$\mathcal{A}(\phi) = \sqrt{\tfrac{1}{2}(1+\lambda)|\mathbf{c}|}\left(-\sin\phi + \cos\phi\,\hat{\mathbf{i}} + \tfrac{\mu\cos\phi + \nu\sin\phi}{1+\lambda}\hat{\mathbf{j}} + \tfrac{\nu\cos\phi - \mu\sin\phi}{1+\lambda}\hat{\mathbf{k}}\right), \tag{A.2}$$

where (λ, μ, ν) and $|\mathbf{c}| \in \mathbb{R}^+$ are respectively the direction-cosines and the magnitude of the vector $\mathbf{c}$. Next, consider a quaternion *polynomial*

$$\mathcal{A}(\zeta) = u(\zeta) + v(\zeta)\hat{\mathbf{i}} + p(\zeta)\hat{\mathbf{j}} + q(\zeta)\hat{\mathbf{k}}\,. \tag{A.3}$$

Then, the following product $\mathcal{A}(\zeta)\,\hat{\mathbf{i}}\,\mathcal{A}^*(\zeta)$ results in

$$\begin{aligned}\mathcal{A}(\zeta)\,\hat{\mathbf{i}}\,\mathcal{A}^*(\zeta) = {}& \left[u^2(\zeta) + v^2(\zeta) - p^2(\zeta) - q^2(\zeta)\right]\hat{\mathbf{i}} \\ & + 2\left[u(\zeta)q(\zeta) + v(\zeta)p(\zeta)\right]\hat{\mathbf{j}} + 2\left[v(\zeta)q(\zeta) - u(\zeta)p(\zeta)\right]\hat{\mathbf{k}}\,.\end{aligned} \tag{A.4}$$

In our approach, we seek to generate spatial PH Bézier curves of degree 5 to describe the paths $\mathbf{p}_{d,i}(\zeta_i)$ mathematically. It can be shown (see [32]) that Equation (A.4) represents the hodograph of the spatial PH path

$$\mathbf{p}'_{d,i}(\zeta_i) = \mathcal{A}_i(\zeta_i)\,\hat{\mathbf{i}}\,\mathcal{A}_i^*(\zeta_i) \tag{A.5}$$

and, hence, the hodograph $\mathbf{p}'_{d,i}(\zeta_i)$ must be of degree 4. To this end, we let $\mathcal{A}_i(\zeta_i)$ be a quadratic Bézier (quaternion) polynomial:

$$\mathcal{A}_i(\zeta_i) = \sum_{k=0}^{2} \bar{\mathcal{A}}_{i,k} b_k^2(\zeta_i)\,,$$

where the control points $\bar{\mathcal{A}}_{i,k}$ are quaternions and defined as

$$\bar{\mathcal{A}}_{i,k} = \bar{u}_{i,k} + \bar{v}_{i,k}\hat{\mathbf{i}} + \bar{p}_{i,k}\hat{\mathbf{j}} + \bar{q}_{i,k}\hat{\mathbf{k}}\,, \qquad \text{for} \quad k = 0, 1, 2\,.$$

With the above definitions, the quaternion polynomial of Equation (A.3) is recovered, where now $u_i(\zeta_i)$, $v_i(\zeta_i)$, $p_i(\zeta_i)$, and $q_i(\zeta_i)$ are quadratic Bézier polynomials:

$$u_i(\zeta_i) = \sum_{k=0}^{2} \bar{u}_{i,k} b_k^2(\zeta_i)\,,\; v_i(\zeta_i) = \sum_{k=0}^{2} \bar{v}_{i,k} b_k^2(\zeta_i)\,,\; p_i(\zeta_i) = \sum_{k=0}^{2} \bar{p}_{i,k} b_k^2(\zeta_i)\,,\; q_i(\zeta_i) = \sum_{k=0}^{2} \bar{q}_{i,k} b_k^2(\zeta_i)\,.$$

The control points of the spatial path $\mathbf{p}_{d,i}(\zeta_i)$, in the form $\bar{\mathbf{p}}_{i,k} = \bar{\mathrm{p}}_{x_{i,k}}\hat{\mathbf{i}} + \bar{\mathrm{p}}_{y_{i,k}}\hat{\mathbf{j}} + \bar{\mathrm{p}}_{z_{i,k}}\hat{\mathbf{k}}$, can be obtained by integrating Equation (A.5):

$$\begin{aligned}
\bar{\mathbf{p}}_{i,1} &= \bar{\mathbf{p}}_{i,0} + \frac{1}{5}\bar{\mathcal{A}}_{i,0}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}_{i,0}^*\,,\\
\bar{\mathbf{p}}_{i,2} &= \bar{\mathbf{p}}_{i,1} + \frac{1}{10}\left(\bar{\mathcal{A}}_{i,0}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}_{i,1}^* + \bar{\mathcal{A}}_{i,1}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}_{i,0}^*\right)\,,\\
\bar{\mathbf{p}}_{i,3} &= \bar{\mathbf{p}}_{i,2} + \frac{1}{30}\left(\bar{\mathcal{A}}_{i,0}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}_{i,2}^* + 4\bar{\mathcal{A}}_{i,1}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}_{i,1}^* + \bar{\mathcal{A}}_{i,2}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}_{i,0}^*\right)\,,\\
\bar{\mathbf{p}}_{i,4} &= \bar{\mathbf{p}}_{i,3} + \frac{1}{10}\left(\bar{\mathcal{A}}_{i,1}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}_{i,2}^* + \bar{\mathcal{A}}_{i,2}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}_{i,1}^*\right)\,,\\
\bar{\mathbf{p}}_{i,5} &= \bar{\mathbf{p}}_{i,4} + \frac{1}{5}\bar{\mathcal{A}}_{i,2}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}_{i,2}^*\,,
\end{aligned} \tag{A.6}$$

where $\bar{\mathbf{p}}_{i,0} = \mathbf{p}_i^{\mathrm{i}}$ and $\bar{\mathbf{p}}_{i,5} = \mathbf{p}_i^{\mathrm{f}}$. It is clear that the quintic PH Bézier curve $\mathbf{p}_{d,i}(\zeta_i)$ is completely determined by the three unknowns $\bar{\mathcal{A}}_{i,k}$ for $k = 0, 1, 2$. The first-order Hermite interpolation conditions at the endpoints provide three equations that can be used to solve for the three unknowns $\bar{\mathcal{A}}_{i,0}$, $\bar{\mathcal{A}}_{i,1}$, and $\bar{\mathcal{A}}_{i,2}$. First, let the derivatives at the initial and final point be given by the vectors $\mathbf{d}_i^{\mathrm{i}}$ and $\mathbf{d}_i^{\mathrm{f}}$, respectively. Then, the direction-

cosines $(\lambda_i, \mu_i, \nu_i)$ can be expressed in terms of the flight-path angle γ_i and course ψ_i at the endpoints:

$$\begin{aligned} \lambda_i^{\mathrm{i}} &= \cos\gamma_i^{\mathrm{i}} \cos\psi_i^{\mathrm{i}}\,, & \lambda_i^{\mathrm{f}} &= \cos\gamma_i^{\mathrm{f}} \cos\psi_i^{\mathrm{f}}\,, \\ \mu_i^{\mathrm{i}} &= \cos\gamma_i^{\mathrm{i}} \sin\psi_i^{\mathrm{i}}\,, & \mu_i^{\mathrm{f}} &= \cos\gamma_i^{\mathrm{f}} \sin\psi_i^{\mathrm{f}}\,, \\ \nu_i^{\mathrm{i}} &= \sin\gamma_i^{\mathrm{i}}\,, & \nu_i^{\mathrm{f}} &= \sin\gamma_i^{\mathrm{f}}\,. \end{aligned}$$

The magnitude of the vectors $\mathbf{d}_i^{\mathrm{i}}$ and $\mathbf{d}_i^{\mathrm{f}}$ are denoted by $|\mathbf{d}_i^{\mathrm{i}}| \in \mathbb{R}^+$, and $|\mathbf{d}_i^{\mathrm{f}}| \in \mathbb{R}^+$, respectively. Note, that these are *not* equal to the given initial and final speed v_i^{i}, and v_i^{f}, due to the decoupling through the timing law $\theta_i(t_d)$. With these definitions, interpolation of the end-derivatives yields the following two equations:

$$\bar{\mathcal{A}}_{i,0}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}^*_{i,0} = \mathbf{d}_i^{\mathrm{i}} \qquad \text{and} \qquad \bar{\mathcal{A}}_{i,2}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}^*_{i,2} = \mathbf{d}_i^{\mathrm{f}}\,.$$

These two equations are in the form of Equation (A.1) and, hence, the solutions for $\bar{\mathcal{A}}_{i,0}$ and $\bar{\mathcal{A}}_{i,2}$ are given by Equation (A.2):

$$\begin{aligned} \bar{\mathcal{A}}_{i,0}(|\mathbf{d}_i^{\mathrm{i}}|, \phi_{i,0}) =& \sqrt{\tfrac{1}{2}(1+\lambda_i^{\mathrm{i}})|\mathbf{d}_i^{\mathrm{i}}|} \\ & \times \left(-\sin\phi_{i,0} + \cos\phi_{i,0}\,\hat{\mathbf{i}} + \tfrac{\mu_i^{\mathrm{i}}\cos\phi_{i,0} + \nu_i^{\mathrm{i}}\sin\phi_{i,0}}{1+\lambda_i^{\mathrm{i}}}\hat{\mathbf{j}} + \tfrac{\nu_i^{\mathrm{i}}\cos\phi_{i,0} - \mu_i^{\mathrm{i}}\sin\phi_{i,0}}{1+\lambda_i^{\mathrm{i}}}\hat{\mathbf{k}} \right)\,, \end{aligned} \tag{A.7}$$

$$\begin{aligned} \bar{\mathcal{A}}_{i,2}(|\mathbf{d}_i^{\mathrm{f}}|, \phi_{i,2}) =& \sqrt{\tfrac{1}{2}(1+\lambda_i^{\mathrm{f}})|\mathbf{d}_i^{\mathrm{f}}|} \\ & \times \left(-\sin\phi_{i,2} + \cos\phi_{i,2}\,\hat{\mathbf{i}} + \tfrac{\mu_i^{\mathrm{f}}\cos\phi_{i,2} + \nu_i^{\mathrm{f}}\sin\phi_{i,2}}{1+\lambda_i^{\mathrm{f}}}\hat{\mathbf{j}} + \tfrac{\nu_i^{\mathrm{f}}\cos\phi_{i,2} - \mu_i^{\mathrm{f}}\sin\phi_{i,2}}{1+\lambda_i^{\mathrm{f}}}\hat{\mathbf{k}} \right)\,, \end{aligned} \tag{A.8}$$

where $\phi_{i,0}$ and $\phi_{i,2}$ are free angular parameters. Contrary to the general solution given by Equation (A.2), which only has angle ϕ as a single free parameter, $\bar{\mathcal{A}}_{i,0}$ and $\bar{\mathcal{A}}_{i,2}$ are also dependent on $|\mathbf{d}_i^{\mathrm{i}}|$, and $|\mathbf{d}_i^{\mathrm{f}}|$. These extra degrees of freedom are a result of the fact that we introduced the timing law in order to capture the temporal specifications, such as the specified initial and final speeds, separately.

The last unknown $\bar{\mathcal{A}}_{i,1}$ can be determined from the endpoint conditions:

$$\begin{aligned} \int_0^1 \mathcal{A}_i(\zeta_i)\,\hat{\mathbf{i}}\,\mathcal{A}_i^*(\zeta_i)\mathrm{d}\zeta_i =& \quad \mathbf{p}_i^{\mathrm{f}} - \mathbf{p}_i^{\mathrm{i}} \\ =& \quad \frac{1}{5}\bar{\mathcal{A}}_{i,0}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}^*_{i,0} + \frac{1}{10}\left(\bar{\mathcal{A}}_{i,0}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}^*_{i,1} + \bar{\mathcal{A}}_{i,1}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}^*_{i,0}\right) \\ & + \frac{1}{30}\left(\bar{\mathcal{A}}_{i,0}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}^*_{i,2} + 4\bar{\mathcal{A}}_{i,1}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}^*_{i,1} + \bar{\mathcal{A}}_{i,2}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}^*_{i,0}\right) \\ & + \frac{1}{10}\left(\bar{\mathcal{A}}_{i,1}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}^*_{i,2} + \bar{\mathcal{A}}_{i,2}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}^*_{i,1}\right) + \frac{1}{5}\bar{\mathcal{A}}_{i,2}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}^*_{i,2}\,. \end{aligned} \tag{A.9}$$

Equation (A.9) can be rewritten as

$$\begin{aligned}
&\left(3\bar{\mathcal{A}}_{i,0}+4\bar{\mathcal{A}}_{i,1}+3\bar{\mathcal{A}}_{i,2}\right)\hat{\mathbf{i}}\left(3\bar{\mathcal{A}}_{i,0}+4\bar{\mathcal{A}}_{i,1}+3\bar{\mathcal{A}}_{i,2}\right)^{*}\\
&\qquad=120\left(\mathbf{p}_i^{\mathrm{f}}-\mathbf{p}_i^{\mathrm{i}}\right)-15\left(\mathbf{d}_i^{\mathrm{i}}+\mathbf{d}_i^{\mathrm{f}}\right)+5\left(\bar{\mathcal{A}}_{i,0}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}_{i,2}^{*}+\bar{\mathcal{A}}_{i,2}\,\hat{\mathbf{i}}\,\bar{\mathcal{A}}_{i,0}^{*}\right)\\
&\qquad=\mathbf{c}\,,
\end{aligned}\tag{A.10}$$

where $\mathbf{c}=c_x\hat{\mathbf{i}}+c_y\hat{\mathbf{j}}+c_z\hat{\mathbf{k}}$, since the right hand side is a pure vector. Again, the solution to Equation (A.10) is given by the general solution (A.2), and using Equations (A.7) and (A.8), $\bar{\mathcal{A}}_{i,1}$ can be found as

$$\begin{aligned}
\bar{\mathcal{A}}_{i,1}(\phi_{i,1})=&-\tfrac{3}{4}\left(\bar{\mathcal{A}}_{i,0}+\bar{\mathcal{A}}_{i,2}\right)+\frac{\sqrt{\frac{1}{2}(1+\lambda_i^{\mathrm{c}})|\mathbf{c}|}}{4}\\
&\times\left(-\sin\phi_{i,1}+\cos\phi_{i,1}\,\hat{\mathbf{i}}+\tfrac{\mu_i^{\mathrm{c}}\cos\phi_{i,1}+\nu_i^{\mathrm{c}}\sin\phi_{i,1}}{1+\lambda_i^{\mathrm{c}}}\hat{\mathbf{j}}+\tfrac{\nu_i^{\mathrm{c}}\cos\phi_{i,1}-\mu_i^{\mathrm{c}}\sin\phi_{i,1}}{1+\lambda_i^{\mathrm{c}}}\hat{\mathbf{k}}\right),
\end{aligned}\tag{A.11}$$

where $(\lambda_i^{\mathrm{c}},\mu_i^{\mathrm{c}},\nu_i^{\mathrm{c}})$ and $|\mathbf{c}|\in\mathbb{R}^{+}$ are respectively the direction-cosines and the magnitude of the vector $\mathbf{c}$, and $\phi_{i,1}$ is another free angular parameter. However, in [32, 34, 35] it is shown that the shape of the curves are only dependent on *differences* of the angles $\phi_{i,0}$, $\phi_{i,1}$, and $\phi_{i,2}$. Hence, without loss of generality we assume that $\phi_{i,1}=-\frac{1}{2}\pi$. Then, in contrast to what is claimed in [32, 34, 35], we have to let $\phi_{i,0},\phi_{i,2}\in[-\pi,\pi]$, such that the parametrization describes the *complete* set of spatial PH Bézier curves. Thus, Equation (A.11) reduces to

$$\bar{\mathcal{A}}_{i,1}=-\tfrac{3}{4}\left(\bar{\mathcal{A}}_{i,0}+\bar{\mathcal{A}}_{i,2}\right)+\frac{\sqrt{\frac{1}{2}(1+\lambda_i^{\mathrm{c}})|\mathbf{c}|}}{4}\left(1-\tfrac{\nu_i^{\mathrm{c}}}{1+\lambda_i^{\mathrm{c}}}\hat{\mathbf{j}}+\tfrac{\mu_i^{\mathrm{c}}}{1+\lambda_i^{\mathrm{c}}}\hat{\mathbf{k}}\right).$$

It is clear that the unknowns $\bar{\mathcal{A}}_{i,0}$, $\bar{\mathcal{A}}_{i,1}$, and $\bar{\mathcal{A}}_{i,2}$ are completely determined by the four parameters $|\mathbf{d}_i^{\mathrm{i}}|$, $|\mathbf{d}_i^{\mathrm{f}}|$, $\phi_{i,0}$, and $\phi_{i,2}$. Therefore, a four-parameter family of solutions characterizes the spatial quintic PH Bézier curve that satisfies the first-order Hermite interpolation.

Appendix B

Proofs

B.1 Proof of Proposition 1

We start by showing that the Bernstein polynomial

$$b_k^n(\alpha\zeta^\alpha) = \binom{n}{k}\left(1-\alpha\zeta^\alpha\right)^{n-k}\left(\alpha\zeta^\alpha\right)^k \tag{B.1}$$

can be rewritten as

$$b_k^n(\alpha\zeta^\alpha) = \sum_{j=k}^{n} b_k^j(\alpha) b_j^n(\zeta^\alpha)\,.$$

We note that $(1-\alpha\zeta^\alpha) = (1-\zeta^\alpha) + (1-\alpha)\,\zeta^\alpha$ and perform a binomial expansion of (B.1):

$$\begin{aligned}
b_k^n(\alpha\zeta^\alpha) &= \tbinom{n}{k}\left(1-\alpha\zeta^\alpha\right)^{n-k}\left(\alpha\zeta^\alpha\right)^k \\
&= \tbinom{n}{k}\left[(1-\zeta^\alpha) + (1-\alpha)\,\zeta^\alpha\right]^{n-k}\left(\alpha\zeta^\alpha\right)^k \\
&= \tbinom{n}{k}\left(\alpha\zeta^\alpha\right)^k \Big[\tbinom{n-k}{0}(1-\zeta^\alpha)^{n-k}(1-\alpha)^0(\zeta^\alpha)^0 \\
&\qquad + \tbinom{n-k}{1}(1-\zeta^\alpha)^{n-(k+1)}(1-\alpha)^1(\zeta^\alpha)^1 + \ldots \\
&\qquad + \tbinom{n-k}{n-k}(1-\zeta^\alpha)^{n-(n)}(1-\alpha)^{n-k}(\zeta^\alpha)^{n-k}\Big] \\
&= \tbinom{n}{k}\Bigg[\tbinom{n-k}{0}\frac{\binom{n}{k}}{\binom{n}{k}}(1-\zeta^\alpha)^{n-k}(\zeta^\alpha)^k\frac{\binom{k}{k}}{\binom{k}{k}}(1-\alpha)^{k-k}\alpha^k \\
&\qquad + \tbinom{n-k}{1}\frac{\binom{n}{k+1}}{\binom{n}{k+1}}(1-\zeta^\alpha)^{n-(k+1)}(\zeta^\alpha)^{k+1}\frac{\binom{k+1}{k}}{\binom{k+1}{k}}(1-\alpha)^{(k+1)-k}\alpha^k + \ldots. \\
&\qquad + \tbinom{n-k}{n-k}\frac{\binom{n}{n}}{\binom{n}{n}}(1-\zeta^\alpha)^{n-(n)}(\zeta^\alpha)^n\frac{\binom{n}{k}}{\binom{n}{k}}(1-\alpha)^{n-k}\alpha^k\Bigg] \\
&= \tbinom{n}{k}\textstyle\sum_{j=k}^{n}\frac{\binom{n-k}{j-k}}{\binom{n}{j}\binom{j}{k}} b_k^j(\alpha) b_j^n(\zeta^\alpha) \\
&= \textstyle\sum_{j=k}^{n} b_k^j(\alpha) b_j^n(\zeta^\alpha)\,.
\end{aligned}$$

Next, given the subdivision point $\zeta = \alpha$, the control points $\bar{r}_k^{\alpha}$ of the Bézier polynomial $r^{\alpha}(\zeta^{\alpha})$ can be determined by the *de Casteljau* algorithm and it is easy to verify that these are given by:

$$\bar{r}_j^{\alpha} = \sum_{k=0}^{j} \bar{r}_k b_k^j(\alpha)\,, \qquad j = 0, \ldots, n\,.$$

Hence, the Bézier polynomial $r^{\alpha}(\zeta^{\alpha})$ is given by:

$$\begin{aligned}
r^{\alpha}(\zeta^{\alpha}) &= \sum_{j=0}^{n} \sum_{k=0}^{j} \bar{r}_k b_k^j(\alpha) b_j^n(\zeta^{\alpha}) \\
&= \sum_{k=0}^{0} \bar{r}_k b_k^0(\alpha) b_0^n(\zeta^{\alpha}) + \ldots + \sum_{k=0}^{n} \bar{r}_k b_k^n(\alpha) b_n^n(\zeta^{\alpha}) \\
&= \bar{r}_0 b_0^0(\alpha) b_0^n(\zeta^{\alpha}) + \ldots + \left[\bar{r}_0 b_0^0(\alpha) b_n^n(\zeta^{\alpha}) + \ldots + \bar{r}_n b_n^n(\alpha) b_n^n(\zeta^{\alpha})\right] \\
&= \bar{r}_0 \left[b_0^0(\alpha) b_0^n(\zeta^{\alpha}) + \ldots + b_0^n(\alpha) b_n^n(\zeta^{\alpha})\right] + \ldots + \bar{r}_n b_n^n(\alpha) b_n^n(\zeta^{\alpha}) \\
&= \sum_{k=0}^{n} \bar{r}_k \sum_{j=k}^{n} b_k^j(\alpha) b_j^n(\zeta^{\alpha}) \\
&= \sum_{k=0}^{n} \bar{r}_k b_k^n(\alpha \zeta^{\alpha}) \\
&= r(\alpha \zeta^{\alpha})\,,
\end{aligned}$$

which holds for all $\zeta^{\alpha} \in [0, 1]$, and completes the proof. □

B.2 Proof of Lemma 1

Recalling that we let $\hat{t}_{d,i}^{\alpha} = \hat{t}_{d,j}$ and using Proposition 1, we can write

$$\mathbf{p}_{d,i}^{\alpha}(\hat{t}_{d,i}^{\alpha}) = \mathbf{p}_{d,i}^{\alpha}(\hat{t}_{d,j}) = \mathbf{p}_{d,i}(\alpha\, \hat{t}_{d,j})\,.$$

Then, noting that $\alpha\, \hat{t}_{d,j} = \hat{t}_{d,i}$, we obtain

$$\mathbf{p}_{d,i}(\alpha\, \hat{t}_{d,j}) = \mathbf{p}_{d,i}(\hat{t}_{d,i})\,, \qquad \forall\, t_d \in [0, t_{d_{ij}}^{\mathrm{f}}]\,,$$

which completes the proof. □

B.3 Proof of Theorem 10

(i) First, we observe that

$$\limsup_{\tau\downarrow 0} \frac{f(x+\tau d)-f(x)}{\tau} \leq \limsup_{\substack{y\to x\\ \tau\downarrow 0}} \frac{f(y+\tau d)-f(y)}{\tau} = f^\circ(x;d)\,.$$

Hence, we obtain

$$L := \limsup_{\tau\downarrow 0} \frac{f(x+\tau d)-f(x)}{\tau} < 0\,.$$

Then there exists $\delta > 0$, such that $L+\delta < 0$. From the definition of the lim sup, there exists $\epsilon > 0$, such that

$$\frac{f(x+\tau d)-f(x)}{\tau} < L+\delta < 0\,, \qquad \text{for all } \tau \in (0,\epsilon\,]\,.$$

Therefore, $f(x+\tau d)-f(x) < 0$ for all $\tau \in (0,\epsilon\,]$, and we conclude that d is a descent direction for f at x.

(ii) From Theorem 4(ii) we know that

$$f^\circ(x;d) = \max\{\xi^\top d \,|\, \xi \in \partial f(x)\}\,, \qquad \text{for all } d \in \mathbb{R}^n\,.$$

Therefore, since $f^\circ(x;d) < 0$, it follows that $\xi^\top d < 0$ for all $\xi \in \partial f(x)$.

(iii) Suppose d is a descent direction for $\bar{f}$ at x. Then there exists $\epsilon' > 0$ such that

$$\bar{f}(x+\tau d) < \bar{f}(x)\,, \qquad \text{for all } \tau \in (0,\epsilon'\,]\,.$$

From the definition of $\bar{f}$ at x we obtain

$$\begin{aligned}
\bar{f}(x+\tau d) - \bar{f}(x) &= \max_{\xi\in\partial f(x)} \{\, f(x) + \langle\, \xi\,, \tau d\,\rangle\} - f(x) \\
&= \tau \max_{\xi\in\partial f(x)} \langle\, \xi\,, d\,\rangle \\
&= \tau\, f^\circ(x;d) \\
&< 0\,, \qquad\qquad \text{for all } \tau \in (0,\epsilon'\,]\,.
\end{aligned}$$

Then, as shown in part i, there exists $\epsilon \in (0,\epsilon'\,]$ such that $f(x+\tau d)-f(x) < 0$ for all $\tau \in (0,\epsilon\,]$, which means that d is a descent direction for f at x. □

B.4 Proof of Theorem 13

Suppose $\hat{\boldsymbol{x}}^* = \mathbb{1}_n \otimes \hat{x}^*$ is a local weak Pareto optimum of (P2). Since $\hat{\boldsymbol{x}}^* \in \Omega$, we have that

$$f_i(\hat{x}^*) - f_i(\hat{x}^*) = 0\,,$$
$$g_i(\hat{x}^*) \leq 0\,,$$

for all $i = 1, \ldots, N$ and, hence, $H(\hat{\boldsymbol{x}}^*; \hat{\boldsymbol{x}}^*) = 0$. Next, let $\rho > 0$ be the radius associated with the local weak Pareto optimum $\hat{\boldsymbol{x}}^*$ and let $\hat{\boldsymbol{x}} \in \mathcal{B}(\hat{\boldsymbol{x}}^*; \rho) \cap \Omega$. Note, by definition of Ω, that $\hat{\boldsymbol{x}} \in \Omega_{\tilde{L}}$. We consider two cases:

(i.) $g_i(\hat{x}^i) \geq 0$ for some $i \in \{1, \ldots, N\}$. Then we have that $H(\hat{\boldsymbol{x}}; \hat{\boldsymbol{x}}^*) \geq 0$.

(ii.) $g_i(\hat{x}^i) \leq 0$ for all $i = 1, \ldots, N$. Then, since $\hat{\boldsymbol{x}}^*$ is a local weak Pareto optimum, $f_i(\hat{x}^i) - f_i(\hat{x}^*) \geq 0$ for some $i \in \{1, \ldots, N\}$ and, hence, $H(\hat{\boldsymbol{x}}; \hat{\boldsymbol{x}}^*) \geq 0$.

Then, by Definition 7, $\hat{\boldsymbol{x}}^*$ is a local minimizer for $H(\cdot\,; \hat{\boldsymbol{x}}^*)$ on $\Omega_{\tilde{L}}$. □.

B.5 Proof of Proposition 4

First, recall that the constraint set $\Omega_{\tilde{L}}$ is defined as

$$\Omega_{\tilde{L}} := \{\hat{\boldsymbol{x}} \in \mathbb{R}^{N^2} \mid \tilde{L}\hat{\boldsymbol{x}} = 0\}\,.$$

The *tangent plane* $\mathrm{T}_{\Omega_{\tilde{L}}}$ at $\hat{\boldsymbol{x}}^*$ is defined as the collection of the derivatives at $\hat{\boldsymbol{x}}^*$ of all differentiable curves $\hat{\boldsymbol{x}}(t)$ on $\Omega_{\tilde{L}}$. Hence, it is clear that $T_{\Omega_{\tilde{L}}} \subset \mathrm{Null}(\tilde{L})$ since any curve $\hat{\boldsymbol{x}}(t)$ passing through $\hat{\boldsymbol{x}}^*$ at $t = t^*$ having derivative $\dot{\hat{\boldsymbol{x}}}(t^*)$ such that $\tilde{L}\dot{\hat{\boldsymbol{x}}}(t^*) \neq 0$, that is $\dot{\hat{\boldsymbol{x}}}(t^*) \notin \mathrm{Null}(\tilde{L})$, would not lie on $\Omega_{\tilde{L}}$.

To prove that $\mathrm{Null}(\tilde{L}) \subset T_{\Omega_{\tilde{L}}}$ we must show that if $\boldsymbol{y} \in \mathrm{Null}(\tilde{L})$ then there is a curve $\hat{\boldsymbol{x}}(t)$ on $\Omega_{\tilde{L}}$ passing through $\hat{\boldsymbol{x}}^*$ at $t = t^*$ with derivative $\dot{\hat{\boldsymbol{x}}}(t^*) = \boldsymbol{y}$. Let $\boldsymbol{y} = \mathbb{1}_N \otimes y$ for some $y \in \mathbb{R}^N$. Then we know that $\boldsymbol{y} \in \mathrm{Null}(\tilde{L})$. Next, we define a function $s : \mathbb{R} \to \mathbb{R}^N$ such that $\lim_{t \to 0} \frac{s(t)}{t} = \mathbf{0}_N$, and let $\boldsymbol{s}(t) = \mathbb{1}_N \otimes s(t)$. We construct the differentiable curve $\hat{\boldsymbol{x}}(t)$ for some interval $t \in [-a, a]$ with $a > 0$ as

$$\hat{\boldsymbol{x}}(t) = \begin{cases} \hat{\boldsymbol{x}}^* + t\,\boldsymbol{y} + \boldsymbol{s}(t) & t \neq 0\,, \\ \hat{\boldsymbol{x}}^* + t\,\boldsymbol{y} & t = 0\,. \end{cases}$$

Hence, it is clear that the curve lies on $\Omega_{\tilde{L}}$ for all $t \in [-a, a]$, and that $\hat{\boldsymbol{x}}(t^*) = \hat{\boldsymbol{x}}^*$ for $t^* = 0$. At t^* the curve is continuous and $\dot{\hat{\boldsymbol{x}}}(t^*) = \boldsymbol{y}$. □

B.6 Proof of Theorem 16

From Proposition 4 we know that the tangent plane $\mathrm{T}_{\Omega_{\tilde{L}}}$ at $\hat{\boldsymbol{x}}^*$ is given by

$$T_{\Omega_{\tilde{L}}}(\hat{\boldsymbol{x}}^*) = \mathrm{Null}(\tilde{L})\,.$$

Hence, from the definition of the normal plane $\mathrm{N}_{\Omega_{\tilde{L}}}$, and the fact that every vector in $\mathrm{Null}(\tilde{L})$ is orthogonal to every vector in $\mathrm{Range}(\tilde{L}^\top)$, we obtain

$$\begin{aligned} N_{\Omega_{\tilde{L}}}(\hat{\boldsymbol{x}}^*) &= \mathrm{Range}(\tilde{L}^\top)\,, \\ &= \left\{\hat{\boldsymbol{x}} \,|\, \hat{\boldsymbol{x}} = \tilde{L}^\top \nu\,, \nu \in \mathbb{R}^{N^2}\right\}\,. \end{aligned}$$

Therefore, if $\hat{\boldsymbol{x}}^*$ is a local minimizer of (P3), then from Theorem 15 we conclude that there exists a vector $\nu \in \mathbb{R}^{N^2}$ such that

$$0 \in \partial \hat{H}(\hat{\boldsymbol{x}}^*; \hat{\boldsymbol{x}}^*) + \tilde{L}^\top \nu\,.$$

This completes the proof. □

B.7 Proof of Theorem 17

Suppose $\boldsymbol{d} \in \mathbb{R}^{N^2}$ is a feasible descent direction for $\hat{H}$ at $\hat{\boldsymbol{x}}_k \in \Omega$ subject to $\Omega_{\tilde{L}}$. Then there exists $\epsilon > 0$ such that

$$\hat{H}(\hat{\boldsymbol{x}}_k + \tau \boldsymbol{d}) < \hat{H}(\hat{\boldsymbol{x}}_k) = 0\,, \tag{B.2}$$

and

$$\hat{\boldsymbol{x}}_k + \tau \boldsymbol{d} \in \Omega_{\tilde{L}}\,,$$

for all $\tau \in (0, \epsilon\,]$. Then, since $f_i(\hat{x}_k^i) = \bar{f}_i(\hat{x}_k^i)$, it follows from the definition of $\hat{H}$ that for $i = 1, \ldots, N$

$$\bar{f}_i(\hat{x}_k^i + \tau d^i) - \bar{f}_i(\hat{x}_k^i) < 0\,, \qquad \text{for all } \ \tau \in (0, \epsilon\,]\,,$$

and, as such, direction d^i is a descent direction for $\bar{f}_i$ at $\hat{x}_k^i$. Therefore, by Theorem 10, there exist $\epsilon^{f_i} \in (0, \epsilon\,]$ such that $f_i(\hat{x}_k^i + \tau d^i) - f_i(\hat{x}_k^i) < 0$ for all $\tau \in (0, \epsilon^{f_i}\,]$, which means that d^i is a descent direction for f_i at $\hat{x}_k^i$ and, thus, direction $\boldsymbol{d}$ is a descent direction for F at $\hat{\boldsymbol{x}}_k$.

Now we have to show that the direction $\boldsymbol{d}$ is feasible subject to Ω_g. If $g_i(\hat{x}_k^i) < 0$ for some $i = 1, \ldots, N$,

then by continuity of g_i there exist $\epsilon^{g_i} > 0$ such that $g_i(\hat{x}_k^i + \tau d^i) \leq 0$ for all $\tau \in (0, \epsilon^{g_i}]$. If $g_i(\hat{x}_k^i) = 0$ for some $i = 1, \ldots, N$, then, by Equation (B.2), we have

$$\bar{g}_i(\hat{x}_k^i + \tau d^i) < \bar{g}_i(\hat{x}_k^i) = g_i(\hat{x}_k^i) = 0\,, \qquad \text{for all } \tau \in (0, \epsilon]\,,$$

which means that d^i is a descent direction for $\bar{g}_i$ at $\hat{x}_k^i$. Using Theorem 10 we conclude that the direction d^i is also a descent direction for g_i at $\hat{x}_k^i$ and, therefore, there exist $\epsilon_0^{g_i} \in (0, \epsilon]$ such that

$$g_i(\hat{x}_k^i + \tau d^i) < g_i(\hat{x}_k^i) = 0\,, \qquad \text{for all } \tau \in (0, \epsilon_0^{g_i}]\,.$$

Let $\delta := \min_{i=1,\ldots,N}\{\epsilon^{f_i}, \epsilon^{g_i}, \epsilon_0^{g_i}\}$. Then we have that $\hat{\boldsymbol{x}}_k + \tau \boldsymbol{d} \in \Omega$ for all $\tau \in (0, \delta]$ and, hence, $\boldsymbol{d}$ is a feasible direction subject to Ω. □

B.8 Proof of Lemma 4

From Definition 2 for the subdifferential of a convex function f_i, we have that

$$\begin{aligned} f_i(\hat{x}^i) &\geq f_i(\hat{x}_\ell^i) + \langle\, \xi_{\ell,m}^i, \hat{x}^i - \hat{x}_\ell^i \,\rangle \\ &= f_i(\hat{x}^i) - \alpha_{\ell,m}^i(\hat{x}^i)\,, \qquad \text{for all } m \in \mathcal{I}_\ell^{f_i}\,, \end{aligned}$$

where the map $\alpha_{\ell,m}^i(\hat{x}^i)$ was defined by Equation (6.21). Hence, $\alpha_{\ell,m}^i(\hat{x}^i) \geq 0$ for all $\hat{x}^i \in \mathbb{R}^N$ and $m \in \mathcal{I}_\ell^{f_i}$. Consider $\bar{\ell} \in \mathcal{I}_k$. Then we find that for all $\hat{x}^i \in \mathbb{R}^N$

$$\begin{aligned} \hat{f}_{i,k}(\hat{x}^i) &= \max_{\ell \in \mathcal{I}_k} \bar{f}_{i,\ell}(\hat{x}^i) \\ &= \max_{\ell \in \mathcal{I}_k} \max_{m \in \mathcal{I}_\ell^{f_i}} \{f_i(\hat{x}_\ell^i) + \langle\, \xi_{\ell,m}^i, \hat{x}^i - \hat{x}_\ell^i \,\rangle - \alpha_{\ell,m}^i(\hat{x}_\ell^i)\} \\ &= f_i(\hat{x}_{\bar{\ell}}^i) + \max_{\ell \in \mathcal{I}_k} \max_{m \in \mathcal{I}_\ell^{f_i}} \{\langle\, \xi_{\ell,m}^i, \hat{x}^i - \hat{x}_{\bar{\ell}}^i \,\rangle - \alpha_{\ell,m}^i(\hat{x}_{\bar{\ell}}^i)\}\,. \end{aligned}$$

Since $\alpha_{\ell,m}^i(\hat{x}^i) \geq 0$ for all $\hat{x}^i \in \mathbb{R}^N$ and $m \in \mathcal{I}_\ell^{f_i}$, we have that $\alpha_{\ell,m}^i(\hat{x}_{\bar{\ell}}^i) \geq 0$ and, in particular from Equation (6.21), $\alpha_{\bar{\ell},m}^i(\hat{x}_{\bar{\ell}}^i) = 0$. Hence, it follows that $\hat{f}_{i,k}(\hat{x}_{\bar{\ell}}^i) = f_i(\hat{x}_{\bar{\ell}}^i)$ for all $\bar{\ell} \in \mathcal{I}_k$. We can repeat the same proof for the convex function g_i. □

B.9 Proof of Lemma 5

To prove that $\hat{H}_k(\hat{\boldsymbol{x}})$ is convex, we first show that the maximum of two convex functions is convex. Let $f_1 : \mathbb{R}^n \to \mathbb{R}$ and $f_2 : \mathbb{R}^n \to \mathbb{R}$ be convex, and $g : \mathbb{R}^n \to \mathbb{R}$ be defined as

$$g(x) := \max\{f_1(x), f_2(x)\}\,.$$

Next, let $y, z \in \mathbb{R}^n$ and $\lambda \in [0, 1]$. Since $f_1(x)$ is convex, and using the definition of $g(x)$, we obtain

$$\begin{aligned} f_1(\lambda y + (1-\lambda)z) &\leq \lambda f_1(y) + (1-\lambda) f_1(z) \\ &\leq \lambda g(y) + (1-\lambda) g(z)\,. \end{aligned}$$

Similarly, we find

$$f_2(\lambda y + (1-\lambda)z) \leq \lambda g(y) + (1-\lambda) g(z)\,.$$

Then we can derive that

$$\begin{aligned} g(\lambda y + (1-\lambda)z) &= \max\{f_1(\lambda y + (1-\lambda)z), f_2(\lambda y + (1-\lambda)z)\} \\ &\leq \lambda g(y) + (1-\lambda) g(z)\,, \end{aligned}$$

and, hence, $g(x)$ is a convex function. Lastly, note that $\hat{H}_k(\hat{\boldsymbol{x}})$ at $\hat{\boldsymbol{x}}_k$ is a maximum of a collection of linear (and convex) functions, and we conclude that $\hat{H}_k(\hat{\boldsymbol{x}})$ is convex.

For the second part, we first note from Corollary 1 and Definition 5 that the functions f_i and g_i are regular. We proceed by deriving the subdifferential for $H(\hat{\boldsymbol{x}}; \hat{\boldsymbol{x}}_k)$ at $\hat{\boldsymbol{x}}_k$. From the definition of $H(\hat{\boldsymbol{x}}; \hat{\boldsymbol{x}}_k)$, we have that $f_i(\hat{x}^i) - f_i(\hat{x}^i_k) = 0$ for all $i = 1, \ldots, N$ at $\hat{\boldsymbol{x}}_k$. Also, since $\hat{\boldsymbol{x}}_k$ is feasible subject to Ω_g, we also find that $g_j(\hat{x}^j_k) \leq 0$ for all $j = 1, \ldots, N$ and, therefore, $H(\hat{\boldsymbol{x}}_k; \hat{\boldsymbol{x}}_k) = 0$. Next, we define the set J as

$$J := \{\, j \in \{1, \ldots, N\} \,|\, g_j(\hat{x}^j_k) = H(\hat{\boldsymbol{x}}_k; \hat{\boldsymbol{x}}_k)\}\,.$$

Then from Theorem 6 in Chapter 5 we obtain

$$\partial F(\hat{\boldsymbol{x}}_k) = \operatorname*{conv}_{i \in I} \{\partial f_i(\hat{x}^i_k)\}\,, \qquad \partial G(\hat{\boldsymbol{x}}_k) = \operatorname*{conv}_{j \in J} \{\partial g_j(\hat{x}^j_k)\}\,,$$

$$\partial H(\hat{\boldsymbol{x}}_k; \hat{\boldsymbol{x}}_k) = \operatorname{conv}\{\partial F(\hat{\boldsymbol{x}}_k) \cup \partial G(\hat{\boldsymbol{x}}_k)\}\,,$$

where it is clear that $I = \{1, \ldots, N\}$. The equality follows from the regularity of the functions f_i and g_i. From the assumption that f_i and g_i are convex, and Lemma 4, we know that

$$\hat{f}_{i,k}(\hat{x}_k^i) = f_i(\hat{x}_k^i)\,,$$
$$\hat{g}_{j,k}(\hat{x}_k^j) = g_j(\hat{x}_k^j)\,.$$

Hence, from the definition of $\hat{H}_k(\hat{\boldsymbol{x}})$ and the fact that $\hat{\boldsymbol{x}}_k$ is feasible subject to Ω_g, we have that $\hat{H}_k(\hat{\boldsymbol{x}}_k) = 0$. Then, let us define the set $\hat{J}$ as

$$\hat{J} := \{\, j \in \{1, \ldots, N\} \,|\, \hat{g}_{j,k}(\hat{x}_k^j) = \hat{H}_k(\hat{\boldsymbol{x}}_k)\}\,,$$

and with the fact that $\hat{g}_{j,k}(\hat{x}_k^j) = g_j(\hat{x}_k^j)$ and $\hat{H}_k(\hat{\boldsymbol{x}}_k) = 0$, we conclude that $\hat{J} = J$. Then using Theorem 6 we obtain

$$\partial\hat{F}(\hat{\boldsymbol{x}}_k) = \operatorname*{conv}_{i\in\hat{I}} \{\partial\hat{f}_{i,k}(\hat{x}_k^i)\}\,, \qquad \partial\hat{G}(\hat{\boldsymbol{x}}_k) = \operatorname*{conv}_{j\in\hat{J}} \{\partial\hat{g}_{j,}(\hat{x}_k^j)\}\,,$$
$$\partial\hat{H}(\hat{\boldsymbol{x}}_k) = \operatorname{conv}\{\partial\hat{F}(\hat{\boldsymbol{x}}_k) \cup \partial\hat{G}(\hat{\boldsymbol{x}}_k)\}\,,$$

where $\hat{I} = \{1, \ldots, N\}$. It can be easily verified, that by construction, we have that

$$\partial\hat{f}_{i,k}(\hat{x}_k^i) = \partial f_i(\hat{x}_k^i)\,, \qquad \text{for all } i = 1, \ldots, N\,,$$
$$\partial\hat{g}_{j,k}(\hat{x}_k^j) = \partial g_j(\hat{x}_k^j)\,, \qquad \text{for all } j = 1, \ldots, N\,.$$

Hence, we conclude that $\partial\hat{H}(\hat{\boldsymbol{x}}_k) = \partial H(\hat{\boldsymbol{x}}_k; \hat{\boldsymbol{x}}_k)$. □

B.10 Proof of Theorem 19

By assumption, the objective functions f_i are convex and, therefore, from Lemma 4 we know that

$$\hat{f}_{i,k}(\hat{x}_k^i) = f_i(\hat{x}_k^i)\,.$$

This implies that $\hat{H}_k(\hat{\boldsymbol{x}}_k) = 0$. Now, suppose $\boldsymbol{d}_k \in \mathbb{R}^{N^2}$ is a descent direction for $\hat{H}_k$ at $\hat{\boldsymbol{x}}_k \in \Omega$ and feasible subject to $\Omega_{\tilde{L}}$. Then there exists $\epsilon > 0$ such that

$$\hat{H}_k(\hat{\boldsymbol{x}}_k + \tau \boldsymbol{d}_k) < \hat{H}_k(\hat{\boldsymbol{x}}_k) = 0\,, \tag{B.3}$$

and

$$\hat{\boldsymbol{x}}_k + \tau \boldsymbol{d}_k \in \Omega_{\tilde{L}}\,,$$

for all $\tau \in (0, \epsilon\,]$. It follows from the definitions of $\hat{H}_k$ and $\hat{f}_{i,k}$ that, for $i = 1, \ldots, N$ and $\ell \in \mathcal{I}_k$,

$$\hat{f}_{i,k}(\hat{x}_k^i + \tau d_k^i) - f_i(\hat{x}_k^i) < 0\,, \qquad \text{for all } \ \tau \in (0, \epsilon\,]$$

and, thus,

$$\bar{f}_{i,\ell}(\hat{x}_k^i + \tau d_k^i) - f_i(\hat{x}_k^i) < 0\,, \qquad \text{for all } \ \tau \in (0, \epsilon\,]\,.$$

In particular, for $\ell = k$ we find that

$$\begin{aligned}
\bar{f}_{i,k}(\hat{x}_k^i + \tau d_k^i) - f_i(\hat{x}_k^i) &= \max_{m \in \mathcal{I}_k^{f_i}} \{\, f_i(\hat{x}_k^i) + \langle\, \xi_{k,m}^i\,,\, \hat{x}_k^i + \tau d_k^i - \hat{x}_k^i \,\rangle - f_i(\hat{x}_k^i) \,\} \\
&= \tau \max_{m \in \mathcal{I}_k^{f_i}} \langle\, \xi_{k,m}^i\,,\, d_k^i \,\rangle \\
&= \tau f_i^\circ(\hat{x}_k^i; d_k^i)\,, \qquad \text{for all } \ \tau \in (0, \epsilon\,]\,.
\end{aligned}$$

With $\bar{f}_{i,k}(\hat{x}_k^i + \tau d_k^i) - f_i(\hat{x}_k^i) < 0$, we obtain $f_i^\circ(\hat{x}_k^i; d_k^i) < 0$ for all $\tau \in (0, \epsilon\,]$. Then by Theorem 10 there exist $\epsilon^{f_i} \in (0, \epsilon\,]$ such that $f_i(\hat{x}_k^i + \tau d_k^i) - f_i(\hat{x}_k^i) < 0$ for all $\tau \in (0, \epsilon^{f_i}\,]$ and, therefore, $\boldsymbol{d}_k$ is a descent direction for F at $\hat{\boldsymbol{x}}_k$.

Next, we have to show that the direction $\boldsymbol{d}_k$ is feasible subject to Ω_g. If $g_i(\hat{x}_k^i) < 0$ for some $i = 1, \ldots, N$, then by continuity of g_i there exist $\epsilon^{g_i} > 0$ such that $g_i(\hat{x}_k^i + \tau d_k^i) \leq 0$ for all $\tau \in (0, \epsilon^{g_i}\,]$. If $g_i(\hat{x}_k^i) = 0$ for some $i = 1, \ldots, N$, then, from the assumption that the inequality constraints g_i are convex and Lemma 4, we obtain

$$\hat{g}_{i,k}(\hat{x}_k^i) = 0\,.$$

Also, from the definition of $\bar{g}_{i,\ell}(\hat{x}^i)$, it follows that for $\ell = k$ we have

$$\bar{g}_{i,k}(\hat{x}_k^i) = 0\,.$$

By Equation (B.3), we have

$$\hat{g}_{i,k}(\hat{x}_k^i + \tau d_k^i) < 0\,, \qquad \text{for all } \ \tau \in (0, \epsilon\,]\,,$$

and, therefore, again from the definition of $\hat{g}_{i,k}(\hat{x}^i)$, we find that

$$\bar{g}_{i,\ell}(\hat{x}_k^i + \tau d_k^i) < 0\,, \qquad \text{for all } \ \tau \in (0, \epsilon\,]\,.$$

Hence, for $\ell = k$ we find that $\bar{g}_{i,k}(\hat{x}_k^i + \tau d_k^i) < \bar{g}_{i,k}(\hat{x}_k^i)$, which means that d_k^i is a descent direction for $\bar{g}_{i,k}$ at $\hat{x}_k^i$. Then by Theorem 10 the direction d_k^i is also a descent direction for g_i at $\hat{x}_k^i$ and, therefore, there exist $\epsilon_0^{g_i} \in (0, \epsilon\,]$ such that

$$g_i(\hat{x}_k^i + \tau d_k^i) < g_i(\hat{x}_k^i) = 0\,, \qquad \text{for all } \ \tau \in (0, \epsilon_0^{g_i}\,]\,.$$

Let $\delta := \min\limits_{i=1,\ldots,N} \{\epsilon^{f_i}, \epsilon^{g_i}, \epsilon_0^{g_i}\}$, then we have that $\hat{\boldsymbol{x}}_k + \tau \boldsymbol{d}_k \in \Omega$ for all $\tau \in (0, \delta\,]$ and, hence, $\boldsymbol{d}_k$ is a feasible direction subject to Ω. □

B.11 Proof of Proposition 5

Let $(\boldsymbol{z}_k^*, \boldsymbol{d}_k^*)$ be the global minimizer of (DFP3). As $\boldsymbol{z}_k^*$ satisfies the constraint $L\,\boldsymbol{z}_k^* = 0$, we know that $\boldsymbol{z}_k^*$ lies in the nullspace of L. Since the Laplacian L corresponds to a connected graph, the nullspace of L is given by

$$\text{Null}(L) = \{\gamma \mathbb{1}_N \,|\, \gamma \in \mathbb{R}\} \tag{B.4}$$

and, hence, $\boldsymbol{z}_k^*$ must be of the form $\boldsymbol{z}_k^* = z_k^* \mathbb{1}_N$ for some $z_k^* \in \mathbb{R}$. Therefore, $z_{i,k} = z_k$ for $i = 1, \ldots, N$ must hold for $\boldsymbol{z}_k$ to be a candidate minimizer of (DFP3). Consequently, the objective function for (DFP3) can be found as

$$\tfrac{1}{N} \sum_{i=1}^{N} z_{i,k} + \tfrac{1}{2}\|\boldsymbol{d}_k\|^2 = z_k + \tfrac{1}{2}\|\boldsymbol{d}_k\|^2\,,$$

where the latter corresponds to the objective function for problem (DFP2). Similarly, it can be verified that due to the specific structure of $\boldsymbol{z}_k$, the inequality constraints of (DFP3) are equivalent to the inequality constraints of (DFP2) and, hence, we have recovered the optimization problem (DFP2).

To prove the converse, let $(z_k^*, \boldsymbol{d}_k^*)$ be the global minimizer of (DFP2) and let $\boldsymbol{z}_k = z_k^* \mathbb{1}_N$. Then since $\boldsymbol{z}_k \in \text{Null}(L)$ by Equation (B.4), the equality constraint $L\,\boldsymbol{z}_k = 0$ of Problem (DFP3) is satisfied. The proof can be completed by the same arguments as in the first part, that the objective functions and inequality constraints for Problems (DFP3) and (DFP2) are equivalent by virtue of the structure of $\boldsymbol{z}_k$. This completes the proof. □

B.12 Proof of Theorem 21

Suppose $\|d_k^i\|^2 = 0$ at the kth iteration, or equivalently, $d_k^i = 0$. Since d_k^i is the global minimizer obtained by vehicle i to the distributed direction finding problem for iteration k, that means that $d_k^{*^i} = d_k^i$. Note that since $\boldsymbol{d}_k \in \Omega_{\tilde{L}}$, it must hold that $\boldsymbol{d}_k = \mathbb{1}_n \otimes d_k$ and, hence, $d_k = d_k^i$ for all $i = 1, \ldots, N$. Therefore, we conclude that $\boldsymbol{d}_k^* = 0$ is a global minimizer for (DFP1) at iteration k.

Then, from Theorem 20, we know that the following necessary condition holds

$$0 \in \partial \hat{H}_k(\hat{\boldsymbol{x}}_k + \boldsymbol{d}_k^*) + \boldsymbol{d}_k^* + \tilde{L}^\top \nu \,,$$

and with $\boldsymbol{d}_k^* = 0$ we have at iteration point $\hat{\boldsymbol{x}}_k + \boldsymbol{d}_k^* = \hat{\boldsymbol{x}}_k$ that

$$0 \in \partial \hat{H}_k(\hat{\boldsymbol{x}}_k) + \tilde{L}^\top \nu \,.$$

Lemma 5 tells us further that $\partial \hat{H}_k(\hat{\boldsymbol{x}}_k) = \partial H(\hat{\boldsymbol{x}}_k; \hat{\boldsymbol{x}}_k)$ and the following hold

$$0 \in \partial H(\hat{\boldsymbol{x}}_k; \hat{\boldsymbol{x}}_k) + \tilde{L}^\top \nu \,.$$

Then, by Theorem 16, the iteration point $\hat{\boldsymbol{x}}_k$ satisfies the necessary condition to be a local minimizer for $H(\hat{\boldsymbol{x}}; \hat{\boldsymbol{x}}^*)$. Lastly, since Assumption 5 holds, then by Theorem 14 and the fact that $\hat{\boldsymbol{x}}_k = \mathbb{1}_N \otimes x_k$, where $x_k = \hat{x}_k^i$ for all $i = 1, \ldots, N$, we conclude that if $\hat{\boldsymbol{x}}_k$ is a local minimizer for Problem (P3), then x_k is a local minimizer of Problem (P1). This completes the proof. □

References

[1] *Concept of operations for the Next Generation Air Transportation System.* Joint Planning and Development Office, Version 3.2, September 2010. Available at www.dtic.mil/dtic/tr/fulltext/u2/a535795.pdf. 1

[2] N. Akhtar, J. F. Whidborne, and A. K. Cooke, *Real-time trajectory generation technique for dynamic soaring UAVs*, in UKACC International Conference on Control, Manchester, UK, September 2008. 3

[3] F. Augugliaro, A. Schoellig, and R. D'Andrea, *Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach*, in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal, October 2012, pp. 1917–1922. 3, 7

[4] A. Bagirov, N. Karmitsa, and M. M. Mäkelä, *Introduction to Nonsmooth Optimization: Theory, Practice and Software*, Springer International Publishing, Switzerland, 2014. 47

[5] N. Bedrossian, S. Bhatt, M. Lammers, L. Nguyen, and Y. Zhang, *First ever flight demonstration of Zero Propellant Maneuver^{TM} attitude control concept*, in AIAA Guidance, Navigation and Control Conference, Hilton Head, SC, August 2007. AIAA-2007-6734. 3

[6] N. S. Bedrossian, S. Bhatt, W. Kang, and I. M. Ross, *Zero-propellant maneuver guidance*, IEEE Control Systems Magazine, 29 (2009), pp. 53–73. 3

[7] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, New York, NY, 1982. 102

[8] ——, *Nonlinear Programming*, Athena Scientific, Belmont, MA, 1999. 84, 104

[9] K. P. Bollino and L. R. Lewis, *Collision-free multi-UAV optimal path planning and cooperative control for tactical applications*, in AIAA Guidance, Navigation and Control Conference, Honolulu, HI, August 2008. AIAA 2008-7134. 3, 6

[10] K. P. Bollino, L. R. Lewis, P. Sekhavat, and I. M. Ross, *Pseudospectral optimal control: A clear road for autonomous intelligent path planning*, in AIAA Infotech@Aerospace, Rohnert Park, CA, May 2007. AIAA 2007-2831. 3

[11] J.-F. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal, *Numerical Optimization*, Springer-Verlag Berlin Heidelberg, New York, NY, 2006. 71

[12] F. Borrelli, D. Subramanian, A. U. Raghunathan, and L. T. Biegler, *MILP and NLP techniques for centralized trajectory planning of multiple unmanned air vehicles*, in American Control Conference, Minneapolis, MN, June 2006, pp. 5763–5768. 3

[13] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, U.K., 2004. 47

[14] J.-W. Chang, Y.-K. Choi, M.-S. Kim, and W. Wang, *Computation of the minimum distance between two Bézier curves/surfaces*, Computers & Graphics, 35 (2011), pp. 677–684. 10, 34

[15] X.-D. Chen, L. Chen, Y. Wang, G. Xu, and J.-H. Yong, *Computing the minimum distance between two Bézier curves*, Journal of Computational and Applied Mathematics, 229 (2009), pp. 294–301. 34

[16] Y. Chen, M. Cutler, and J. P. How, *Decoupled multiagent path planning via incremental sequential convex programming*, in 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, May 2015, pp. 5954–5961. 3, 7

[17] R. Choe, *Distributed cooperative trajectory generation for multiple autonomous vehicles using Pythagorean Hodograph Bézier curves*. Preliminary Examination, University of Illinois at Urbana-Champaign, Urbana, IL, United States, July 2014. 10, 41

[18] R. Choe, V. Cichella, E. Xargay, N. Hovakimyan, A. C. Trujillo, and I. Kaminer, *A trajectory-generation framework for time-critical cooperative missions*, in AIAA Infotech@Aerospace Conference, Boston, MA, August 2013. AIAA 2013-4582. 3, 9, 11, 16, 17, 18, 19, 20

[19] H. I. Choi, D. S. Lee, and H. P. Moon, *Clifford algebra, spin representation, and rational parameterization of curves and surfaces*, Advances in Computational Mathematics, 17 (2002), pp. 5–48. 30

[20] V. Cichella, R. Choe, B. S. Mehdi, E. Xargay, N. Hovakimyan, V. Dobrokhodov, and I. Kaminer, *Trajectory generation and collision avoidance for safe operation of cooperating UAVs*, in AIAA Guidance, Navigation and Control Conference, National Harbor, MD, January 2014. AIAA 2014-0972. 1, 3, 9, 11, 13, 16, 17, 18, 19, 20, 23

[21] V. Cichella, R. Choe, S. B. Mehdi, E. Xargay, N. Hovakimyan, V. Dobrokhodov, I. Kaminer, A. M. Pascoal, and A. P. Aguiar, *Safe coordinated maneuvering of teams of multirotor unmanned aerial vehicles: A cooperative control framework for multi-vehicle, time-critical missions*, IEEE Control Systems Magazine, 36 (2016), pp. 59–82. 1, 50

[22] V. Cichella, I. Kaminer, E. Xargay, V. Dobrokhodov, N. Hovakimyan, A. P. Aguiar, and A. M. Pascoal, *A Lyapunov-based approach for time-coordinated 3D path-following of multiple quadrotors*, in IEEE Conference on Decision and Control, Maui, HI, December 2012, pp. 1776–1781. 1, 23

[23] F. H. Clarke, *Generalized gradients and applications*, Transactions of the American Mathematical Society, 205 (1975), pp. 247–262. 66

[24] ——, *Optimization and Nonsmooth Analysis*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1990. 47, 61, 63, 64

[25] J. M. Danskin, *The theory of max-min, with applications*, SIAM Journal on Applied Mathematics, 14 (1966), pp. 641–664. 61, 66

[26] P. J. Davis, *Interpolation and Approximation*, Dover books on Mathematics, Dover Publications, New York, 1975. 31

[27] T. D. DeRose, *Composing Bézier simplexes*, ACM Transactions on Graphics, 7 (1988), pp. 198–221. 32, 43

[28] V. R. Desaraju and J. P. How, *Decentralized path planning for multi-agent teams with complex constraints*, Autonomous Robots, 32 (2012), pp. 385–403. 4

[29] C. Ericson, *The Gilbert-Johnson-Keerthi algorithm*. SIGGRAPH Presentation, 2004. Sony Computer Entertainment America. 35

[30] F. Fahroo and I. M. Ross, *On discrete-time optimality conditions for pseudospectral methods*, in AIAA/AAS Astrodynamics Specialist Conference, Keystone, CO, August 2006. AIAA 2006-6304. 3

[31] R. T. Farouki, *The elastic bending energy of Pythagorean-hodograph curves*, Computer Aided Geometric Design, 13 (1996), pp. 227–241. 27, 37

[32] ——, *Pythagorean-Hodograph Curves*, Springer-Verlag, Berlin Heidelberg, 2008. 19, 28, 29, 30, 32, 37, 38, 39, 40, 43, 115, 116, 118

[33] ——, *The Bernstein polynomial basis: A centennial retrospective*, Computer Aided Geometric Design, 29 (2012), pp. 379–419. 32, 39, 40, 43

[34] R. T. Farouki, M. al Kandari, and T. Sakkalis, *Hermite interpolation by rotation-invariant spatial Pythagorean-Hodograph curves*, Advances in Computational Mathematics, 17 (2002), pp. 369–383. 30, 118

[35] R. T. Farouki, C. Giannelli, C. Manni, and A. Sestini, *Identification of spatial PH quintic Hermite interpolants with near-optimal shape measures*, Computer Aided Geometric Design, 25 (2008), pp. 274–297. 30, 118

[36] R. T. Farouki and C. Y. Han, *Algorithms for spatial Pythagorean-Hodograph curves*, in Geometric Properties for Incomplete Data, vol. 31 of Computational Imaging and Vision, Springer, 2006, pp. 43–58. 30

[37] R. T. Farouki and T. Sakkalis, *Pythagorean hodographs*, IBM Journal of Research and Development, 34 (1990), pp. 736–752. 27, 37

[38] E. Frazzoli, M. A. Dahleh, and E. Feron, *Real-time motion planning for agile autonomous vehicles*, Journal of Guidance, Control and Dynamics, 25 (2002), pp. 116–129. 4

[39] R. Ghabcheloo, A. P. Aguiar, A. M. Pascoal, C. Silvestre, I. Kaminer, and J. P. Hespanha, *Coordinated path-following in the presence of communication losses and delays*, SIAM Journal on Control and Optimization, 48 (2009), pp. 234–265. 23

[40] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, *A fast procedure for computing the distance between complex objects in three-dimensional space*, IEEE Journal of Robotics and Automation, 4 (1988), pp. 193–203. 35

[41] Q. Gong, L. R. Lewis, and I. M. Ross, *Pseudospectral motion planning for autonomous vehicles*, Journal of Guidance, Control and Dynamics, 32 (2009), pp. 1039–1045. 3

[42] A. J. Häusler, R. Ghabcheloo, A. M. Pascoal, A. P. Aguiar, I. Kaminer, and V. N. Dobrokhodov, *Temporally and spatially deconflicted path planning for multiple autonomous marine vehicles*, in 8th Conference on Manoeuvring and Control of Marine Craft, Guarujá (SP), Brazil, September 2009. 10

[43] R. Hettich and K. O. Kortanek, *Semi-infinite programming: Theory, methods, and applications*, SIAM Review, 35 (1993), pp. 380–429. 59

[44] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms I*, Springer-Verlag Berlin Heidelberg, New York, NY, 1993. 71

[45] ——, *Convex Analysis and Minimization Algorithms II*, Springer-Verlag Berlin Heidelberg, New York, NY, 1993. 71

[46] B. Johansson, M. Rabi, and M. Johansson, *A simple peer-to-peer algorithm for distributed optimization in sensor networks*, New Orleans, LA, December 2007, pp. 4705–4710. 57

[47] M. Karpenko and I. M. Ross, *A review of pseudospectral optimal control: From theory to flight*, Annual Reviews in Control, 36 (2012), pp. 182–197. 3

[48] K. C. Kiwiel, *Methods of Descent for Nondifferentiable Optimization*, vol. 1133 of Lecture Notes in Mathematics, Springer Berlin Heidelberg, Berlin, 1985. 8, 71, 76, 94, 98, 107, 108, 113

[49] Y. Kuwata and J. P. How, *Cooperative distributed robust trajectory optimization using receding horizon MILP*, IEEE Transactions on Control System Technology, 19 (2011), pp. 423–431. 3

[50] S. M. LaValle, *Planning Algorithms*, Cambridge University Press, Cambridge, UK, 2006, ch. 5: Sampling-Based Motion Planning. 4

[51] C. Lemaréchal, *Nondifferentiable Optimisation. Subgradient and ϵ-Subgradient Methods*, vol. 117 of Lecture Notes in Economics and Mathematical Systems, Springer Berlin Heidelberg, 1976, pp. 191–199. 71, 72, 73

[52] P. Lindemann, *The Gilbert-Johnson-Keerthi distance algorithm*, in Media Informatics Proseminar on Algorithms in Media Informatics, 2009. 35

[53] M. I. Lizárraga and G. H. Elkaim, *Spatially deconflicted path generation for multiple UAVs in a bounded airspace*, in IEEE/ION Position, Location and Navigation Symposium, Monterey, CA, May 2008, pp. 1213–1218. 3, 9, 10, 27, 37

[54] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming*, Springer, New York, NY, 2008. 84, 89, 102, 104

[55] M. M. Mäkelä, V.-P. Eronen, and N. Karmitsa, *On Nonsmooth Multiobjective Optimality Conditions with Generalized Convexities*, Springer New York Heidelberg Dordrecht London, New York, NY, 2014, pp. 333–357. 88

[56] M. M. Mäkelä and P. Neittaanmäki, *Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control*, World Scientific Publishing Co., Singapore, 1992. 8, 61, 62, 64, 65, 68, 73, 76, 91, 93

[57] I. Matei and J. S. Baras, *A performance comparison between two consensus-based distributed optimization algorithms*, Tech. Rep. 2012-05, The Institute for Systems Research, College Park, MD, May 2012. http://hdl.handle.net/1903/12480. 4, 5, 79, 80, 82

[58] ———, *Distributed algorithms for optimization problems with equality constraints*, Florence, Italy, December 2013, pp. 2352–2357. 4, 5, 79, 80, 82, 105

[59] ———, *Distributed algorithms for optimization problems with equality constraints*, Tech. Rep. 2013-05, The Institute for Systems Research, College Park, MD, February 2013. http://drum.lib.umd.edu/handle/1903/13693. 4, 5, 79, 80, 82, 105

[60] ———, *Distributed nonlinear programming methods for optimization problems with inequality constraints*, Osaka, Japan, December 2015, pp. 2649–2654. 4, 5, 79, 80, 82, 89

[61] ———, *Nonlinear programming methods for distributed optimization*, Tech. Rep. 2015-01, The Institute for Systems Research, College Park, MD, January 2015. http://hdl.handle.net/1903/16055. 4, 5, 79, 80, 82

[62] I. Matei, J. S. Baras, M. Nabi, and T. Kurtoglu, *An extension of the method of multipliers for distributed nonlinear programming*, Los Angeles, CA, December 2014, pp. 6951–6956. 4, 5, 79, 80, 82, 105

[63] T. W. McLain and R. W. Beard, *Coordination variables, coordination functions, and cooperative timing missions*, Journal of Guidance, Control and Dynamics, 28 (2005), pp. 150–161. 23

[64] S. B. Mehdi, R. Choe, V. Cichella, and N. Hovakimyan, *Collision avoidance through path replanning using Bézier curves*, in AIAA Guidance, Navigation and Control Conference, Kissimmee, FL, January 2015. AIAA 2015-0598. 1, 10

[65] S. B. Mehdi, R. Choe, and N. Hovakimyan, *Piecewise Bézier curves for avoiding collisions during multi-vehicle coordinated missions*, Journal of Guidance, Control and Dynamics. Submitted. 1, 10

[66] ——, *Avoiding multiple collisions through trajectory replanning using piecewise Bézier curves*, in IEEE Conference on Decision and Control, Osaka, Japan, December 2015. 1, 10

[67] D. Mellinger and V. Kumar, *Minimum snap trajectory generation and control for quadrotors*, in 2011 IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, May 2011, pp. 5954–5961. 3

[68] M. B. Milam, K. Mushambi, and R. M. Murray, *A new computational approach to real-time trajectory generation for constrained mechanical systems*, in IEEE Conference on Decision and Control, Sydney, Australia, December 2000, pp. 845–851. 3, 21

[69] A. Nedić and A. Ozdaglar, *On the rate of convergence of distributed subgradient methods for multi-agent optimization*, New Orleans, LA, December 2007, pp. 4711–4716. 4, 5

[70] ——, *Distributed subgradient methods for multi-agent optimization*, IEEE Transactions on Automatic Control, 54 (2009), pp. 48–61. 4, 5, 57

[71] A. Nedić, A. Ozdaglar, and P. A. Parrilo, *Constrained consensus and optimization in multi-agent networks*, IEEE Transactions on Automatic Control, 55 (2010), pp. 922–938. 4, 5, 57

[72] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering, Springer, New York, NY, 2006. 89

[73] J. Ousingsawat and M. E. Campbell, *Optimal cooperative reconnaissance using multiple vehicles*, Journal of Guidance, Control and Dynamics, 30 (2007), pp. 122–132. 3

[74] P. Parpas and B. Rüstem, *An algorithm for the global optimization of a class of continuous minimax problems*, Journal of Optimization Theory and Applications, 141 (2009), pp. 461–473. 59

[75] E. Polak, *On the mathematical foundations of nondifferentiable optimization in engineering design*, SIAM Review, 29 (1987), pp. 21–89. 59

[76] ——, *Optimization: Algorithms and Consistent Approximations*, Springer-Verlag, New York, NY, 1997. 47, 59, 90

[77] R. L. Raffard, C. J. Tomlin, and S. P. Boyd, *Distributed optimization for cooperative agents: Application to formation flight*, Atlantis, Paradise Island, Bahamas, December 2004, pp. 2453–2459. 4

[78] T. R. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970. 61, 62, 68

[79] I. M. Ross, *A beginner's guide to DIDO: A Matlab application package for solving optimal control problems*, Tech. Rep. TR-711, Elissar Global, Monterey, 2007. 6

[80] B. Rüstem and M. Howe, *Algorithms for Worst-Case Design and Applications to Risk Management*, Princeton University Press, Princeton, NJ, 2002. 59

[81] B. Rüstem, S. Žaković, and P. Parpas, *Convergence of an interior point algorithm for continuous minimax*, Journal of Optimization Theory and Applications, 136 (2008), pp. 87–103. 59

[82] ——, *An interior point algorithm for continuous minimax: Implementation and computation*, Optimization Methods & Software, 23 (2008), pp. 911–928. 59

[83] E. Scholte and M. E. Campbell, *Robust nonlinear model predictive control with partial state information*, IEEE Transactions on Control System Technology, 16 (2008), pp. 636–651. 3

[84] T. Schouwenaars, J. P. How, and E. Feron, *Decentralized cooperative trajectory planning of multiple aircraft with hard safety guarantees*, in AIAA Guidance, Navigation and Control Conference, Providence, RI, August 2004. AIAA 2004-5141. 3

[85] M. A. Shah and N. Aouf, *3D cooperative Pythagorean Hodograph path planning and obstacle avoidance for multiple UAVs*, in IEEE International Conference on Cybernetic Intelligence Systems, Reading, UK, September 2010. 3, 9, 10, 37

[86] M. Shanmugavel, A. Tsourdos, R. Żbikowski, B. A. White, C.-A. Rabbath, and N. Léchevin, *A solution to simultaneous arrival of multiple UAVs using Pythagorean Hodograph curves*, in American Control Conference, Mineapolis, MN, June 2006, pp. 2813–2818. 3, 9, 10, 19, 20, 37

[87] A. Shapiro, *Semi-infinite programming, duality, discretization and optimality conditions*, Optimization, 58 (2009), pp. 133–161. 59

[88] N. Z. Shor, *Minimization Methods for Non-Differentiable Functions*, vol. 3 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 1985. 72

[89] K. Srivastava, A. Nedić, and D. Stipanović, *Distributed Bregman-distance algorithms for min-max optimization*, in Agent-Based Optimization, I. Czarnowski, P. Jędrzejowicz, and J. Kacprzyk, eds., vol. 456 of Studies in Computational Intelligence, Springer-Verlag Berlin Heidelberg, 2013, pp. 143–174. 4, 5, 57

[90] K. Srivastava, A. Nedić, and D. M. Stipanović, *Distributed constrained optimization over noisy networks*, Atlanta, GA, December 2010, pp. 1945–1950. 4, 5

[91] ——, *Distributed min-max optimization in networks*, Corfu, Greece, July 2011. 4, 5

[92] O. Stein, *How to solve a semi-infinite optimization problem*, European Journal of Operational Research, 223 (2012), pp. 312–320. 59

[93] G. Still, *Generalized semi-infinite programming: Theory and methods*, European Journal of Operational Research, 119 (1999), pp. 301–313. 59

[94] S. Subchan, B. A. White, A. Tsourdos, M. Shanmugavel, and R. Żbikowski, *Pythagorean Hodograph (PH) path planning for tracking airborne contaminant using sensor swarm*, in IEEE International Instrumentation and Measurement Technology Conference, Victoria, Vancouver Island, Canada, May 2008. 3, 9, 37

[95] V. T. Taranenko, *Experience of Employment of Ritz's, Poincare's, and Lyapunov's Methods for Solving Flight Dynamics Problems*, Air Force Engineering Academy Press, Moscow, 1968. (in Russian). 10

[96] V. T. Taranenko and V. G. Momdzhi, *Direct Variational Method in Boundary Problems of Flight Dynamics*, Mashinostroenie Press, Moscow, 1986. (in Russian). 10

[97] H. Terelius, U. Topcu, and R. M. Murray, *Decentralized multi-agent optimization via dual decomposition*, vol. 44, Milan, Italy, August-September 2011, pp. 11245–11254. 4, 110

[98] A. Tsourdos, B. White, and M. Shanmugavel, *Cooperative Path Planning of Unmanned Aerial Vehicles*, American Institute of Aeronautics and Astronautics, Inc, Reston, VA, 2011. 3, 9, 10, 13, 19, 20

[99] G. van den Bergen, *A fast and robust GJK implementation for collision detection of convex objects*, Journal of Graphics Tools, 4 (1999), pp. 7–25. 35

[100] D. Varagnolo, F. Zanella, A. Cenedese, G. Pillonetto, and L. Schenato, *Newton-Raphson consensus for distributed convex optimization*, IEEE Transactions on Automatic Control, 61 (2016), pp. 994–1009. 4, 5

[101] F. G. Vázquez, J.-J. Rückmann, O. Stein, and G. Still, *Generalized semi-infinite programming: A tutorial*, Journal of Computational and Applied Mathematics, 217 (2008), pp. 394–419. 59

[102] P. Wolfe, *On the convergence of gradient methods under constraint*, IBM Journal of Research and Development, 16 (1972), pp. 407–411. 71

[103] ——, *A Method of Conjugate Subgradients for Minimizing Nondifferentiable Functions*, vol. 3 of Mathematical Programming Studies, Springer Berlin Heidelberg, 1975, ch. 8, pp. 145–173. 71

[104] E. Xargay, *Time-Critical Cooperative Path-Following Control of Multiple Unmanned Aerial Vehicles*, PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, United States, May 2013. 1, 23, 31

[105] E. Xargay, V. Dobrokhodov, I. Kaminer, A. M. Pascoal, N. Hovakimyan, and C. Cao, *Time-critical cooperative control for multiple autonomous systems*, IEEE Control Systems Magazine, 32 (2012), pp. 49–73. 23

[106] E. Xargay, I. Kaminer, A. Pascoal, N. Hovakimyan, V. Dobrokhodov, V. Cichella, A. P. Aguiar, and R. Ghabcheloo, *Time-critical cooperative path following of multiple unmanned aerial vehicles over time-varying networks*, Journal of Guidance, Control and Dynamics, 36 (2013), pp. 499–516. 1, 23

[107] O. A. Yakimenko, *Direct method for rapid prototyping of near-optimal aircraft trajectories*, Journal of Guidance, Control and Dynamics, 23 (2000), pp. 865–875. 3, 10