UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN
"TECNOLOGIE DELL'INFORMAZIONE"

CICLO XXXII

Optimization-based speed planning for mobile robots and industrial manipulators

Coordinatore:
Chiar.mo Prof. Marco Locatelli

Tutore:
Chiar.mo Prof. Aurelio Piazzi

Dottorando: Andrea Minari

Anni 2016/2019

Ai miei genitori
per il loro costante sostegno

Contents

Introduction

This dissertation presents the results obtained during my PhD program in *Tecnologie dell'Informazione* at the University of Parma, in the three years period 2016-2019.

This thesis deals with the generation of the reference speed profile to be followed by a mobile robot or an industrial manipulator in order to complete an assigned task in minimum-time. The speed planning is a non trivial problem. Indeed, depending on the kind of robot considered and the purpose of the application, the sought speed profile has to take into account a series of limitations and constraints. For example, the velocity planner of an autonomous vehicle has to consider the limitations concerning the maximum and minimum acceleration and deceleration of the vehicle and the maximum speed allowed by the path, for instance in order to avoid the vehicle's sideslip, or imposed by the external environment. Furthermore, the speed planning problem for an industrial manipulator turns out to be more complex than the mobile robot one. In particular, the higher the number of degrees of freedom possessed by the robot, the higher the number of constraints and limitations on the speed profile have to be considered. Moreover, for the case of industrial manipulators, besides the speed and the acceleration constraints, aspects such as the dynamics of the robots and the actuators constraints have to be taken into account.

The aim of this work is to provide efficient algorithms that are able to return the optimal minimum-time speed reference profile. The efficiency of the speed planning algorithm is a mandatory feature when the robot has to operate in dynamic environments in which the motion replanning can be frequently requested. For instance, if an autonomous car has to start an overtaking manoeuvre, then the speed profile to be followed has to be immediately planned satisfying a real-time deadline.

As thoroughly discussed in this work, assuming that the path to be traversed is assigned, the minimum time speed planning problem can be formulated as a finite dimensional problem as the following one:

$$\begin{aligned} &\min_{\mathbf{x}\in\mathbb{R}^{\mathbf{n}}} f(\mathbf{x}) \\ &g_i(\mathbf{x}) \leq 0 \quad i = 1, \ldots, p, \end{aligned}$$

where the objective function is the time to be minimized, while the constraints represent the limitations that are imposed. The literature on the *Mathematical Programming* proposes a wide number of solvers, both commercial and open source, which are able to return a global or, at least, local solution to a finite dimensional optimization problem. However, such solvers do not exploit the special structure of speed planning problems and, thus, turn out to require computing times not compatible with real-time requirements.

This thesis proposes algorithms which exploit the structure of each finite dimensional problem formulated in order to get the optimal solution with the best possible time complexity and trying as much as possible to avoid using external solvers.

The structure and the main contributions of the thesis are the following.

- In the very first part of the thesis (Chapter 1) some of the fundamental and more recent results concerning the speed planning problem for mobile robots and industrial manipulators are presented.

- In Chapter 2 the minimum-time speed planning problem for a mobile robot that has to satisfy speed and longitudinal and normal acceleration constraints is solved. Assuming the path to be traversed is *a priori* known, the problem is formulated as a nonlinear program by using curvilinear discretization. After that, an algorithm that computes the optimal solution to the problem with linear time complexity is then presented. The complexity is proved being the best possible for such problem.

- Chapter 3 is an extension of the results presented in Chapter 2. In particular, the exact continuous-time solution of the minimum-time speed planning problem is computed without performing a finite-dimensional reduction. Chapter 3 presents a necessary and sufficient condition for the feasibility of the minimum-time speed planning problem for a mobile robot. Moreover it presents a simple functional operator, based on the the solution of two ordinary differential equations, which computes the optimal solution. Finally, the properties about the feasible set and about the functional operator are proved by using the lattice theory.

- Chapter 4 deals with the speed planning problem for industrial manipulators. This problem is somehow related to the one addressed in Chapter 2, but it is more challenging because of the higher number of constraints. Indeed, the speed law to be planned has to satisfy constraints on the speed, the acceleration and the torque for each degree of freedom of the robot. As for Chapter 2, the addressed optimization problem is the finite-dimensional reduction obtained by the curvilinear discretization of the path parametrized with respect to the arc-length. The proposed algorithm in proved to be optimal and it can achieve linear-time complexity $O(np)$, where n is the number of discretization points and p is the number of degrees of freedom of the robot. Again the complexity of the proposed algorithm is proved to be the best possible. Moreover, some implementation details to speed up the proposed algorithm are provided.

- In the last chapter (Chapter 5) the minimum-time speed planning prob-

lem for an autonomous vehicles is again solved, but it is updated with the addition of further constraints. In particular, in order to obtain a sufficiently smooth speed profile, constraints on the second derivative of the speed are considered. The aim of this chapter is to provide an efficient algorithm to solve a finite dimensional optimization problem by exploiting as much as possible the results presented in Chapters 2 and 4. The proposed algorithm is based on a standard line search method based on the sequential solution of convex problems. The main contribution lies in the procedure that is proposed to efficiently solve such problems.

Notation

Let us denote $\mathbb{R}_+$ the set of nonnegative real numbers and define the extended real line $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$. For a vector $\mathbf{x} \in \mathbb{R}^n$, $|\mathbf{x}| \in \mathbb{R}^n_+$ denotes the component-wise absolute value of $\mathbf{x}$. Define the norms $\|\mathbf{x}\|_2 := \sqrt{\sum_{i=1}^n |x_i|^2}$ and $\|\mathbf{x}\|_\infty := \max\{|x_1|, \dots, |x_n|\}$. Define $\mathbf{1} = [1 \dots 1]^T$. For $a, b \in \mathbb{R}$, define $a \wedge b = \min\{a, b\}$, $a \vee b = \max\{a, b\}$, and for $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$, define $\mathbf{u} \wedge \mathbf{v}$, $\mathbf{u} \vee \mathbf{v}$ as the analogous elementwise maximum and minimum operations. Define the partial order $\leq$ on $\mathbb{R}^n$ as $\mathbf{u} \leq \mathbf{w}$ if $u_i \leq w_i$, $i = 1, \dots, n$. Given $U \subset \mathbb{R}^n$, we say that $\mathbf{u} \in U$ is the component-wise maximum of U if $\mathbf{u} \geq \mathbf{w}$, $\forall \mathbf{w} \in U$. Finally, define $\bigvee U$ as $\max\{\mathbf{u} : \mathbf{u} \in U\}$ and $\bigwedge U$ as $\min\{\mathbf{u} : \mathbf{u} \in U\}$ with $U \subset \mathbb{R}^n$.

For $r \in \mathbb{N}$, we denote by $C^r([a, b], \mathbb{R}^n)$ the set of continuous functions from $[a, b] \subset \mathbb{R}$ to $\mathbb{R}^n$ that have continuous first r derivatives. For $f \in C^1([a, b], \mathbb{R})$, f' denotes the derivative and notation $\dot{f}$ is used if f is a function of time. We set $\|\boldsymbol{f}\|_\infty := \sup_{i=1,\dots,n} \sup\{|f_i(x)| : x \in [a, b]\}$. We say that $\boldsymbol{f} : [a, b] \to \mathbb{R}^n$ is bounded if there exists $M \in \mathbb{R}$ such that $\|\boldsymbol{f}(x)\|_\infty \leq M$.

The set of real continuous functions with piecewise-continuous first-order derivative is denoted by C^1_{p}. Given an interval $I = [a, b]$ and a measurable function $f : I \to \mathbb{R}$, define $L^\infty(I) = \{f : I \to \mathbb{R} \ : \ \|f\|_\infty < \infty\}$, and $W^{1,\infty}(I) =$

$\{f \in L^\infty(I) : Df \in L^\infty(I)\}$, where Df denotes the weak derivative of f. Let $f, g : I \to \mathbb{R}$, define $f \wedge g$, $f \vee g$, as the pointwise minimum and maximum operations, respectively. Moreover, let us define partial order $\leq$ as follows $f \leq g \iff (\forall x \in I)\ f(x) \leq g(x)$. Consider $h, g : \mathbb{N} \to \mathbb{R}$. We say that $h(n) = O(g(n))$, if there exists a positive constant M such that, for all sufficiently large values of n, $|h(n)| \leq M|g(n)|$.

Chapter 1

State of the art

An important problem in motion planning is the computation of the minimum-time motion of an autonomous vehicle or a robotic manipulator from a start configuration to a target one while avoiding collisions and satisfying kinematic, dynamic and mechanical constraints (for instance, on velocities, accelerations, torques and maximal steering angles). In literature this problem is approached in two ways: as *minimum-time trajectory planning problem* where both the path to be followed by the robot and the timing law on this path (i.e., the robot's velocity) are simultaneously designed, or as a decoupled problem where a (geometric) *path planning problem* is followed by a *minimum-time speed planning* on the planned path [1, 2, 3].

In general, the first paradigm requires the solution of a complex problem with an high computational cost [4]. The latter paradigm, which is known as *path-velocity decomposition* [2, 3], is a sub-optimal approach with respect to the former. In path planning, an high-level planner determines the geometric path that has to be followed by the robot taking into account obstacle avoidance and kinematic constraints. For instance, to comply with the vehicle's nonholonomic constraints, the planned path must have a certain geometric continuity and satisfy geometric interpolation conditions at its endpoints [5, 6, 7]. Moreover, in the case of industrial manipulators the planned path must also avoid or

handle kinematic singularities [8]. After that, the speed planner determines the speed of the robot along the path without violating actuator constraints.The decoupled paradigm is easier to implement in real-time when the robot's motion supervisor uses a sensor-based iterative motion planning [9, 5]. Moreover, if a very efficient speed planning methods is used, then there could be benefits for the overall motion planning. For instance, speed and path planning tasks can be iteratively solved, modifying the geometric path after each speed planning [10].

In this thesis, the *path-velocity decomposition* paradigm is followed and a path that joins the initial and the final configuration is assumed to be assigned compatibly with the type of the robot considered. About minimum time speed planning, literature collects different approaches that can be applied either to mobile robots, for instance an autonomous vehicles, or to industrial manipulators.

For mobile robots like autonomous vehicles, an almost time-optimal speed planning on a given path based on *ternary* polynomials is presented in [11]. Moreover, a simple solution based on root extraction of a quartic equation is reported in [12] emphasizing arbitrary speed/acceleration boundary conditions. Then, a more elaborate solution based on *time discretization* is presented in [13]. It uses a sequence of linear programming feasibility tests to obtain a minimum-time speed profile with constraints on velocities, accelerations and jerks. However, these works do not explicitly consider the constraint on the maximum normal (centripetal) acceleration of the vehicle. This constraint is central to prevent the vehicle from sideslipping, especially when high curvatures and velocities may be present. An early work considering this constraint is [14] which presents an algorithm using linear profiles for the longitudinal accelerations. Other works addressing this issue are [15], [16], [17], [18]. Solea and Nunes [15] achieve a solution with an algorithm based on the five-splines scheme of [19]. Villagra et al. [16] propose a closed-form speed profiler by using the elementary speed profiles provided in [20]. The speed planning of Chen et al. [17] is composed of simple linear or quadratic speed profiles. Li et al. [18] propose the generation of a speed profile on the curvilinear frame (i.e., a speed

as a function of the path's arc-length) taking into account maximal normal and longitudinal accelerations and maximal brake decelerations. Other works also consider constraints that come from the dynamic of the vehicles [21], and comfort and safety requirements [22, 23, 24]. Reference [22] solves a speed planning problem which, besides considering constraints on maximum longitudinal and normal acceleration and maximum speed, is subject to a further Lipschitz condition on the acceleration in order to get a smooth speed profile. In this case the authors propose an efficient algorithm suitable for real-time implementation which exploits the structure of the problem. Besides, Raineri et al. [23, 24] consider a speed planning problem subject to a jerk limitation and a position-dependent speed bound. Even tough the authors propose an heuristic approach, the methods they present are experimentally proved to be suitable for real-time implementation in an industrial scenario.

Some other works directly solve in continuous-time the speed planning for mobile robots, namely they provide an analytical or semi-analytical solution. In particular, reference [25] proposes a semi-analytical optimal solution of the speed planning problem. It presents an algorithm that, as a first step, identifies some specific points on the path curvature profile (e.g., points of minimum curvature radius and points between which the curvature is constant or monotone). After that, from each of these points, the algorithm computes two speed profiles by using a forward and a backward integration scheme, assuming that the vehicle is moving with the maximum allowed acceleration or deceleration. Finally, the optimal solution is obtained by computing the point-wise minimum among all the computed speed profiles. References [26, 27] propose a similar methodology under the assumption that the path is composed of a sequence of clothoids. As a first result, they provide the optimal speed profile for a single clothoid satisfying lateral and longitudinal speed limitations and boundary conditions. They exploit this result to compute the optimal speed profile for a sequence of clothoids. The algorithm in [26] and [27] starts with the acceleration profile which saturates the maximum speed constraints, then lowers this profile by taking into account the longitudinal acceleration constraints. Finally,

with a *forward* and a *backward sweep* it computes the optimal speed-law.

For what concerns the minimum time speed planning for the industrial manipulators, there are mainly three different families of speed profile generation methods: *Numerical Integration*, *Dynamic Programming*, and *Optimization-based*. Note that the same approaches can be also applied to the case of autonomous vehicles.

References [28, 29] are among the first works that address this problem by the *Numerical Integration* approach. In particular, they find the time-optimal speed law *as a function of arc-length* and not as a function of time (see also [30]). This choice simplifies the mathematical structure of the resulting problem and has been adopted by most of successive works. In [28, 29] the optimization problem is solved with iterative algorithms. In particular, reference [28] finds the points in which the acceleration changes sign using the numerical integration of the second order differential equations representing the motions obtained with the maximum and minimum possible accelerations. Reference [29] is based on geometrical considerations on the feasible set. This approach has some limitations due to the determination of the *switching points* that is the main source of failure of this approach (see [31, 32]). For recent results on *Numerical Integration* see [33, 34, 35, 36]. For instance, [35] considers the case of redundant manipulators, while [36] extends the applicability of the above cited approach to the case including constraints on the jerk and torque rate.

In the *Dynamic Programming* approach, the problem is solved with a finite element approximation of the Hamilton-Jacobi-Bellman equation (see [37, 38, 39]). The main difficultly with this approach is the high computational time due to the need of solving a problem with a large number of variables [40].

The *Optimization-based* approach is based on the approximation of the speed planning problem, which turns out to be an infinite dimensional optimization problem, with a finite dimensional one obtained through spatial discretization. Reference [41] is one of the early works using this approach. It shows that speed planning problem including speed acceleration and torque constraints becomes convex after a change of variables and that a discretized

version of the problem is a *Second-Order Cone Programming* (SOCP) problem [42]. This approach has the advantage that the optimization problem can be tackled with available solvers or by using an interior point algorithm (e.g. see [43, 42]). However, the convex programming approach could be inappropriate (see for instance [33]) for online motion planning since the computational time grows rapidly (even if still polynomially) with respect to the number of samples in the discretized problem. Subsequent works, starting from [41], propose algorithms that reduce the computational time (see [44, 45]). To reduce computational time, reference [44] proposes an approach based on *Sequential Linear Programming* (SLP). Namely, the algorithm proposed in [44] sequentially linearizes the objective function around the current point, while a trust region method ensures the convergence of the process. Further, [45] shows that, using a suitable discretization method, the time optimal speed profile can be obtained by *Linear Programming* (LP) with the benefit of a lower computation time with respect to convex solvers. Recently, reference [46] has proposed a novel approach to the speed planning. In particular it proposes a forward and backward approach that requires the computation of a sequence of linear programs. However, the solution returned is proved being optimal under the assumption that no zero-inertial points occur [31]. This work is widely discussed in Section 4.3.2 since presents some similarities with the approach proposed in the this thesis. Furthermore, references [47, 48] extend the approach proposed by [41] to different scenarios. In particular, [47] amends the minimum time problem including jerk, speed dependent and torque rate constraints. The resulting problem loses the convexity property, however the authors propose an iterative algorithm based on the sequential solution of convex problems to find a local solution. With a similar solution [48] applies the sequential convex approach of [47] to the waiter problem.

Chapter 2

An optimal complexity algorithm for minimum-time speed planning

We consider the speed planning problem for a wheeled robot. In particular, we present an algorithm for finding the time-optimal speed law along an assigned path that satisfies speed and longitudinal and normal acceleration constraints. The addressed optimization problem is a finite dimensional reformulation of the continuous-time speed optimization problem, obtained by discretizing the speed profile with n points. The proposed algorithm has linear complexity with respect to n which is the best possible for this problem.

2.1 Problem formulation

We consider a wheeled robot which has to follow a given Cartesian path from a current configuration to a future desired one (see Figure 2.1). Let s_f be the total length of the path, we require to travel this distance in minimum-time while satisfying constraints on the speed and on the longitudinal and normal acceleration of the vehicle.

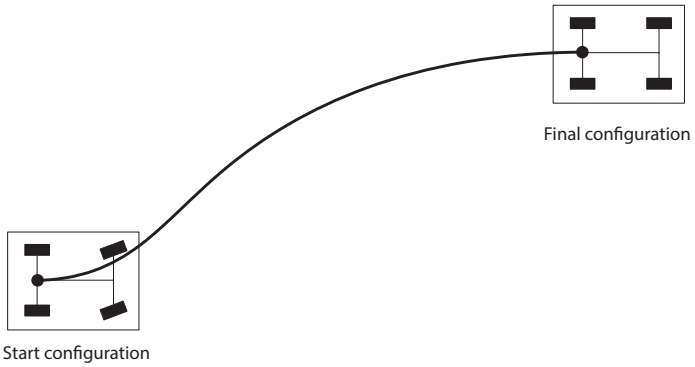


Figure 2.1: A path to follow for an autonomous car-like vehicle.

By denoting t_f the time to travel the path, the problem can be approached by searching a speed profile $v(t) \in C^1_{\mathrm{p}}[0, t_f]$ as the solution of the following problem

Problem 1 (Minimum-time problem in continuous time)**.**

$$\min_{v \in C^1_p[0,t_f]} \quad t_f$$

such that

$$\begin{aligned}
& s(t_f) = s_f \quad \textit{with} \quad s(t) := \int_0^t v(\xi) d\xi, \\
& v(0) = v_s, \quad v(t_f) = v_f, && (2.1)\\
& 0 \leq v(t) \leq v_{max}, \qquad t \in [0, t_f], && (2.2)\\
& a^L_{min} \leq \dot{v}(t) \leq a^L_{max}, \qquad t \in [0, t_f], && (2.3)\\
& v^2(t)|k(s(t))| \leq a^N_{max}, \qquad t \in [0, t_f]\,. && (2.4)
\end{aligned}$$

Equalities (2.1) are the interpolation conditions where v_{s} is the vehicle's start speed and v_f is the assigned final speed at the path end. Inequalities (2.2) impose that the speed cannot be negative (i.e., the vehicle cannot go backward along the planned path) and does not exceed the maximum value $v_{\max}$. Constraints (2.3) limit the maximum and minimum longitudinal accelerations by

$a^L_{\min} < 0$ and $a^L_{\max} > 0$ whereas inequality (2.4) imposes the upper bound $a^N_{\max} > 0$ on the absolute value of the normal acceleration. In Inequality (2.4) k is the curvature of the path.

A difficulty in addressing Problem 1 lies in the normal acceleration constraint (2.4). In this form, it is a nonlinear constraint that makes difficult to find a global solution. With the aim to overcome this difficulty we propose a problem reformulation based on the arc-length parametrization and a change of variables (see [41]).

Let $\boldsymbol{\gamma} \in C^2([0, s_f], \mathbb{R}^2)$ be a function such that $\|\boldsymbol{\gamma}'(\lambda)\| = 1, \forall \lambda \in [0, s_f]$. with s_f be the length of $\boldsymbol{\gamma}$. The image set $\boldsymbol{\gamma}([0, s_f])$ represents the path followed by a vehicle, $\boldsymbol{\gamma}(0)$ the initial configuration and $\boldsymbol{\gamma}(s_f)$ the final one. We want to compute the speed-law that minimizes the overall transfer time while satisfying some kinematic and dynamic requirements. To this end, let $\lambda : [0, t_f] \to [0, s_f]$ be a differentiable monotone increasing function that represents the vehicle position as a function of time and let $v : [0, s_f] \to [0, +\infty]$ be such that $(\forall t \in [0, t_f])\ \dot{\lambda}(t) = v(\lambda(t))$. In this way, $v(s)$ is the vehicle speed at position s. The position of the vehicle as a function of time is given by $\boldsymbol{x} : [0, t_f] \to \mathbb{R}^2$, $\boldsymbol{x}(t) = \boldsymbol{\gamma}(\lambda(t))$, the velocity and acceleration are given by

$$\begin{aligned}\dot{\boldsymbol{x}}(t) =& \boldsymbol{\gamma}'(\lambda(t))v(\lambda(t)),\\ \ddot{\boldsymbol{x}}(t) =& \boldsymbol{\gamma}''(\lambda(t))[v(\lambda(t))]^2 + \boldsymbol{\gamma}(\lambda(t))v'(\lambda(t))v(\lambda(t)).\end{aligned}$$

Note that we can decompose $\boldsymbol{\gamma}''(\lambda(t))$ in the following way

$$\boldsymbol{\gamma}''(\lambda(t)) = \langle \boldsymbol{\gamma}''(\lambda(t)), \boldsymbol{\gamma}'(\lambda(t))\rangle \boldsymbol{\gamma}'(\lambda(t)) + \langle \boldsymbol{\gamma}''(\lambda(t)), \boldsymbol{\gamma}'(\lambda(t))^{\perp}\rangle \boldsymbol{\gamma}'(\lambda(t))^{\perp}.$$

Since by initial hypothesis we have that $(\forall \lambda \in [0, s_f])\ \|\boldsymbol{\gamma}'(\lambda)\| = 1$, we can easily show that

$$\langle \boldsymbol{\gamma}''(\lambda(t)), \boldsymbol{\gamma}'(\lambda(t))\rangle = 0.$$

and rewrite the second derivative as

$$\ddot{\boldsymbol{x}}(t) = a_L(t)\boldsymbol{\gamma}'(\lambda(t)) + a_N(t)\boldsymbol{\gamma}'^{\perp}(\lambda(t)),$$

where $a_L(t) = v'(\lambda(t))v(\lambda(t))$ and $a_N(t) = k(\lambda(t))v(\lambda(t))^2$ are the longitudinal and normal components of acceleration, respectively. Here $k : [0, s_f] \to \mathbb{R}$ is the scalar curvature, defined as $k(s) = \left\langle \boldsymbol{\gamma}''(s), \boldsymbol{\gamma}'(s)^{\perp} \right\rangle$.

Consider the following change of variables where, $(\forall s \in [0, s_f])$ we set

$$a_L(s) = v'(s)v(s), \quad w(s) = v(s)^2, \tag{2.5}$$

and note that

$$w'(s) = 2a_L(s). \tag{2.6}$$

The traversed time to be minimized is then defined as

$$t_f = \int_0^{t_f} 1\, dt = \int_0^{s_f} v(s)^{-1}\, ds = \int_0^{s_f} w(s)^{-1/2}\, ds.$$

Then, Problem 1 can be rewritten as follows:

Problem 2 (Minimum-time problem with arc-length parametrization)**.**

$$\min_{w \in C_p^1([0,s_f])} \int_0^{s_f} w(s)^{-1/2}\, ds$$

such that

$$\begin{aligned}
&w(0) = v_s^2, \quad v(t_f) = v_f^2, && \\
&0 \le w(s) \le v_{max}^2, && s \in [0, s_f], \\
&2a_{min}^L \le w'(s) \le 2a_{max}^L, && s \in [0, s_f], \\
&w(s)|k(s)| \le a_{max}^N, && s \in [0, s_f]
\end{aligned}$$

Note that Problem 2 is a simplified version of the problem formulated in [41] for an industrial robot manipulator. Then, thanks to arc-length parametrization and the change of variables (2.5)-(2.6) we have obtained an optimization problem which is convex.

In this chapter we are not interested in finding the exact optimal solution to Problem 2 but we want to find an approximated solution based on a finite dimensional approximation. For this reason, we uniformly divide the path into

$n-1$ elementary path parts, each one of length $h := s_f/(n-1)$. We denote the endpoints of these elementary parts as $\boldsymbol{\gamma}_i$, $i = 1, 2, \ldots, n$ ($\boldsymbol{\gamma}_1$ and $\boldsymbol{\gamma}_n$ are the starting and final points respectively). Let w_i be the squared vehicle's speed at point γ_i. Then, the average speed on the i-th path part (between $\boldsymbol{\gamma}_i$ and $\boldsymbol{\gamma}_{i+1}$) is $(\sqrt{w_i} + \sqrt{w_{i+1}})/2$ and the corresponding travel time is

$$t_i := \frac{2h}{\sqrt{w_i} + \sqrt{w_{i+1}}}.$$

Therefore, the total time to travel the complete path is

$$t_f = \sum_{i=1}^{n-1} t_i = 2h \sum_{i=1}^{n-1} \frac{1}{\sqrt{w_i} + \sqrt{w_{i+1}}}.$$

At point $\boldsymbol{\gamma}_i$, let s_i be the arc length, then longitudinal acceleration on the i-th part of the path can be obtained using the forward Euler approximation as follows

$$w_i' = \frac{w_{i+1} - w_i}{h}$$

while the normal acceleration is $w_i k(s_i)$.

With this discretization, the addressed planning problem is to find a square speed sequence $\mathbf{w}^* := (w_1^*, \ldots, w_n^*) \in \mathbb{R}^n$ that is the optimal solution of the following problem

$$\min_{\mathbf{w} \in \mathbb{R}^n} 2h \sum_{i=1}^{n-1} \frac{1}{\sqrt{w_i} + \sqrt{w_{i+1}}}$$

such that

$$\begin{aligned}
& w_1 = v_s^2, \;\; w_n = v_f^2 \\
& 0 \le w_i \le v_{\max}^2, \; i = 1, \ldots, n \\
& 2h a_{\min}^L \le w_{i+1} - w_i \le 2h a_{\max}^L, && i = 1, \ldots, n-1, \\
& w_i |k(s_i)| \le a_{\max}^N, && i = 1, \ldots, n.
\end{aligned}$$

Define $k_i := k(s_i)$, $l_B := 2ha^L_{\min}$, $u_B := 2ha^L_{\max}$ (from the assumptions, note that $l_B < 0$ and $u_B > 0$ necessarily), and

$$u_i := \min\left\{v^2_{\max}, \frac{a^N_{\max}}{|k_i|}\right\},\ i = 1, \ldots, n.$$

Then, the minimum-time problem can be summarized as follows

Problem 3 (minimum-time speed planning problem)**.**

$$\min_{\mathbf{w}\in\mathbb{R}^n} \quad 2h\sum_{i=1}^{n-1}\frac{1}{\sqrt{w_{i+1}}+\sqrt{w_i}} \tag{2.7}$$

such that

$$w_1 = v_s^2,\ \ w_n = v_f^2 \tag{2.8}$$

$$0 \le \mathbf{w} \le \mathbf{u}, \tag{2.9}$$

$$w_{i+1} - w_i \le u_B, \qquad i = 1, \ldots, n-1, \tag{2.10}$$

$$w_i - w_{i+1} \le -l_B, \qquad i = 1, \ldots, n-1\,. \tag{2.11}$$

The feasible set of Problem 3 is a non-empty set since $\mathbf{w} = 0$ is a feasible solution, if we assume $v_s = v_f = 0$. Since Problem 3 is convex, we can easily find a solution with an interior point method [41].

2.2 Solution algorithm for a class of optimization problems

In this section we present a special class of optimization problems and an algorithm that finds their optimal solution. After that, we show how this operator can be easy converted in a simple three phases algorithm which has time-optimal complexity.

Consider the following optimization problem

Problem 4 (Generalized problem)**.**

$$\min_{\mathbf{v}\in\mathbb{R}^n} \Psi(\mathbf{v}) \tag{2.12}$$

such that

$$\mathbf{v}^- \leq \mathbf{v} \leq \mathbf{v}^+ \tag{2.13}$$

$$g_i(v_{i-1}, v_i) \leq 0,\ i = 2, \ldots, n \tag{2.14}$$

$$r_i(v_i, v_{i+1}) \leq 0,\ i = 1, \ldots, n-1, \tag{2.15}$$

with the following assumptions:

Assumption 1. *i.* $\Psi : \mathbb{R}^n \to \bar{\mathbb{R}}$ *is a monotonic decreasing function (i.e.,* $\mathbf{v} \geq \mathbf{w}$ *implies* $\Psi(\mathbf{v}) \leq \Psi(\mathbf{w})$*).*

ii. For $i = 2, \ldots, n$, $g_i(x, y) : \mathbb{R} \times \mathbb{R} \to \bar{\mathbb{R}}^{m_1}$ *is monotonic non decreasing with respect to* y *and monotonic non increasing with respect to* x*, while, for* $i = 1, \ldots, n-1$, $r_i(x, y) : \mathbb{R} \times \mathbb{R} \to \bar{\mathbb{R}}^{m_2}$ *is monotonic non decreasing with respect to* x *and monotonic non increasing with respect to* y*. Here,* m_1*,* m_2 *are positive real numbers.*

For $i = 2, \ldots, n$ define function $\varphi_i : \bar{\mathbb{R}} \to \bar{\mathbb{R}}$, such that $\varphi_i(-\infty) = -\infty$ and, if $x \in \mathbb{R}$

$$\varphi_i(x) = \begin{cases} -\infty, \text{ if } \{y \in \mathbb{R} \ : \ g_i(x, y) \leq 0\} \text{ is empty} \\ \sup\{y \in \mathbb{R} \ ; \ g_i(x, y) \leq 0\} \text{ otherwise} \end{cases}$$

Similarly, for $i = 1, \ldots, n-1$ define function $\rho : \bar{\mathbb{R}} \to \bar{\mathbb{R}}$, such that $\rho(-\infty) = -\infty$ and, if $y \in \mathbb{R}$

$$\rho_i(x) = \begin{cases} -\infty, \text{ if } \{y \in \mathbb{R} \ : \ r_i(x, y) \leq 0\} \text{ is empty} \\ \sup\{y \in \ \mathbb{R} \ ; \ r_i(x, y) \leq 0\} \text{ otherwise} \end{cases}$$

Note that φ and ρ_i are monotone non decreasing functions.

Assumption 2. *Superiority conditions:*

$$\begin{aligned} &i = 2, \ldots, n, && \forall\ x:\ v_i^- \le x \le v_i^+, \quad \varphi_i(x) \ge x, \\ &i = 1, \ldots, n-1, && \forall\ x:\ v_i^- \le x \le v_i^+, \quad \rho_i(x) \ge x. \end{aligned}$$

Set $P = \bar{\mathbb{R}}^n$, note that $< P; \wedge, \vee >$ is a complete lattice. Hence, for each subset $S \subseteq P$ exists the unique least upper bound $\mathbf{x} \in P$, such that

$$(\mathbf{y} \in P)\,[((\forall \mathbf{s} \in S)\ \mathbf{s} \le \mathbf{y}) \iff \mathbf{x} \le \mathbf{y}]$$

The least upper bound of S is denoted by $\bigvee S$ (see Definitions 2.1, 2.4 and Notation 2.3 on pages 33–34 of [49]).

Define functions $F, B, M : P \to P$, such that, for $\mathbf{u} \in P$, $F(\mathbf{u})$, $B(\mathbf{u})$ are given by the solutions of the difference equations

$$F(\mathbf{u}) = \begin{cases} f_1 = u_1 \\ f_i = \varphi_i(f_{i-1}) \wedge u_i, \quad i = 2, \ldots, n \end{cases},$$

$$B(\mathbf{u}) = \begin{cases} b_n = u_n \\ b_i = \rho_i(b_{i+1}) \wedge u_i, \quad i = n-1, \ldots, 1\,, \end{cases}$$

and

$$M(\mathbf{u}) = F(\mathbf{u}) \wedge B(\mathbf{u}).$$

We will refer to the following definitions.

Definition 1. *A function $\phi : P \to P$ is meet-preserving if, $\forall \mathbf{x}, \mathbf{y} \in P$,*

$$\phi(\mathbf{x} \wedge \mathbf{y}) = \phi(\mathbf{x}) \wedge \phi(\mathbf{y})\,.$$

Definition 2. *A function $\phi : P \to P$ is order preserving if, $\forall \mathbf{x}, \mathbf{y} \in P$,*

$$\mathbf{x} \le \mathbf{y} \Rightarrow \phi(\mathbf{x}) \le \phi(\mathbf{y})\,.$$

Proposition 1. *Functions F, B, M are meet-preserving and order preserving.*

Proof. Let $\mathbf{u}, \mathbf{v} \in P$. Let $\mathbf{a} = F(\mathbf{u})$, $\mathbf{b} = F(\mathbf{v})$, $\mathbf{c} = F(\mathbf{u} \wedge \mathbf{v})$. We prove that $\mathbf{c} = \mathbf{a} \wedge \mathbf{b}$, by induction. First, note that $c_1 = a_1 \wedge b_1$, then, assume that $c_k = a_k \wedge b_k$, $k = 1, \ldots, i-1$. Then, since φ_i is a monotonic nondecreasing function, $c_i = \varphi_i(a_{i-1} \wedge b_{i-1}) \wedge u_i = (\varphi_i(a_{i-1}) \wedge u_i) \wedge \varphi_i(b_{i-1}) \wedge u_i) = a_i \wedge b_i$. The proof that $B(\mathbf{u} \wedge \mathbf{v}) = B(\mathbf{u}) \wedge B(\mathbf{v})$ is analogous. Finally, $M(\mathbf{u} \wedge \mathbf{v}) = F(\mathbf{u} \wedge \mathbf{v}) \wedge B(\mathbf{u} \wedge \mathbf{v}) = F(\mathbf{u}) \wedge F(\mathbf{v}) \wedge B(\mathbf{u}) \wedge B(\mathbf{v}) = F(\mathbf{u}) \wedge B(\mathbf{u}) \wedge F(\mathbf{v}) \wedge B(\mathbf{v}) = M(\mathbf{u}) \wedge M(\mathbf{v})$. Since maps F, B, M are meet-preserving, they are also order preserving (see Proposition 2.19 on page 44 of [49]).

□

Proposition 2. *Function M satisfies the following properties, $\forall \mathbf{u} \in \mathbb{R}^n$,*

a) $M(\mathbf{u}) \leq \mathbf{u}$,

b) $M(M(\mathbf{u})) = M(\mathbf{u})$.

Proof. a) It is a consequence of the definition of M.

b) Let $\mathbf{f} = F(\mathbf{u})$, $\mathbf{b} = B(\mathbf{u})$, $\mathbf{m} = M(\mathbf{u})$, $\mathbf{a} = F(M(\mathbf{u}))$, $\mathbf{c} = B(M(\mathbf{u}))$. We prove that $\mathbf{a} = \mathbf{m}$, analogously it can be proved that $\mathbf{c} = \mathbf{m}$, which implies the thesis. Note that $a_1 = m_1$, by induction assume that $a_k = m_k$, $k = 1, \ldots, i-1$, then $a_i = \varphi_i(a_{i-1}) \wedge m_i = \varphi_i(m_{i-1}) \wedge m_i = \varphi_i(f_{i-1} \wedge b_{i-1}) \wedge (f_i \wedge b_i) = (\varphi_i(f_{i-1}) \wedge f_i) \wedge (\varphi_i(b_{i-1}) \wedge b_i) = f_i \wedge (\varphi_i(b_{i-1}) \wedge b_i)$, where we have used the fact that $\varphi_i(f_{i-1}) \geq f_i$, that follows from the definition $f_i = \varphi_i(f_{i-1}) \wedge u_i$. By definition of b_{i-1}, we have $b_{i-1} = \rho_{i-1}(b_i) \wedge u_{i-1}$, so that $a_i = f_i \wedge (\varphi_i(\rho_{i-1}(b_i) \wedge u_{i-1}) \wedge b_i) = f_i \wedge \varphi_i(\rho_{i-1}(b_i)) \wedge \varphi_i(u_{i-1}) \wedge b_i$. By Assumptions 2, we have that

$$\varphi_i(\rho_{i-1}(b_i)) \geq b_i.$$

Being φ_i monotonic non-decreasing, since $u_{i-1} \geq f_{i-1}$, and using again the fact that $\varphi_i(f_{i-1}) \geq f_i$, we have that

$$\varphi_i(u_{i-1}) \geq \varphi_i(f_{i-1}) \geq f_i.$$

Then, it follows that:

$$a_i = f_i \wedge \varphi_i(\rho_{i-1}(b_i)) \wedge \varphi_i(u_{i-1}) \wedge b_i = f_i \wedge b_i = m_i.$$

□

Proposition 3.

$$M(\mathbf{v}^+) = \bigvee\{\mathbf{x} \in P : \mathbf{x} \leq M(\mathbf{x}),\ \mathbf{x} \leq \mathbf{v}^+\}$$

Proof. Set $U = \{\mathbf{u} \in P : \mathbf{u} \leq \mathbf{v}^+\}$, note that $< U; \vee, \wedge >$ is a sublattice of $< P; \vee, \wedge >$, moreover by (a) of Proposition 2, if $\mathbf{x} \in U$, then $M(\mathbf{x}) \in U$. Since M is order preserving by Proposition 1, by the Knaster-Tarski Fixpoint Theorem (Theorem 2.35 on page 50 in [49])

$$\mathbf{x}^* = \bigvee\{\mathbf{x} \in P : \mathbf{x} \leq M(\mathbf{x}),\ \mathbf{x} \leq \mathbf{v}^+\}$$

is such that $\mathbf{x}^* = M(\mathbf{x}^*)$, moreover $\mathbf{x}^*$ is the greatest fixed point of M such that $\mathbf{x}^* \in U$. By b) of Proposition 2, $\mathbf{x} = M(\mathbf{v}^+)$ is also a fixed point of M, thus , by definition of $\mathbf{x}^*$, $\mathbf{x} \leq \mathbf{x}^*$. To prove that $\mathbf{x}^* = \mathbf{x}$ it remains to show that $\mathbf{x}^* \leq \mathbf{x}$. To this end assume, by contradiction, that $\mathbf{x}^* \not\leq \mathbf{x}$. Since $\mathbf{x}^* = M(\mathbf{x}^*)$, $\mathbf{x} = M(\mathbf{v}^+)$ and M is order preserving, it follows that $\mathbf{x}^* \not\leq \mathbf{v}^+$, which contradicts the definition of $\mathbf{x}^*$. □

Proposition 4. *The following two statements are equivalent*

i) Set $\{\mathbf{x} \in P : \mathbf{x} = M(\mathbf{x}),\ \mathbf{v}^- \leq \mathbf{x} \leq \mathbf{v}^+\}$ *is not empty,*

ii) $M(\mathbf{v}^+) \geq \mathbf{v}^-$.

Proof. ii) ⇒ i) It follows from the fact that $\mathbf{x}^* = M(\mathbf{v}^+)$ is such that $M(\mathbf{x}^*) = \mathbf{x}^*$ by b) of Proposition 2.

i) ⇒ ii) By contradiction, assume that $M(\mathbf{v}^+) \not\geq \mathbf{v}^-$. Choose any $\mathbf{z} \in P$ such that $\mathbf{z} = M(\mathbf{z})$ and $\mathbf{z} \leq \mathbf{v}^+$. By Proposition 3, $M(\mathbf{z}) \leq M(\mathbf{v}^+)$, hence $\mathbf{z} \leq M(\mathbf{v}^+) \not\geq \mathbf{v}^-$ so that $\mathbf{z} \not\geq \mathbf{v}^-$. Thus, being $\mathbf{z}$ generic, set $\{\mathbf{x} \in P : \mathbf{x} = M(\mathbf{x}),\ \mathbf{v}^- \leq \mathbf{x} \leq \mathbf{v}^+\}$ is empty. □

Proposition 5. *Set* $\mathbf{v} \in P$*, then* $\mathbf{v}$ *is feasible for Problem 4 if and only if* $\mathbf{v}^- \leq \mathbf{v} \leq \mathbf{v}^+$ *and* $M(\mathbf{v}) = \mathbf{v}$.

Proof. $\Rightarrow$) Assume that $\mathbf{v}$ is feasible for Problem 4, we first prove that $F(\mathbf{v}) = \mathbf{v}$. Let $\mathbf{f} = F(\mathbf{v})$, note that $f_1 = v_1$, by induction, assume that $f_k = v_k$, $k = 1, \ldots, i-1$. By feasibility, $g_i(v_{i-1}, v_i) \leq 0$, so that $\varphi_i(v_{i-1}) = \max\{y \in \mathbb{R} : g_i(v_{i-1}, y) \leq 0\} \geq v_i$ and $f_i = \varphi_i(f_{i-1}) \wedge v_i = \varphi_i(v_{i-1}) \wedge v_i = v_i$. Analogously, we prove that $B(\mathbf{v}) = \mathbf{v}$. Then, by definition of M, $M(\mathbf{v}) = \mathbf{v}$. Moreover, since $\mathbf{v}$ satisfies the bounds of Problem 4, $\mathbf{v}^- \leq \mathbf{v} \leq \mathbf{v}^+$.

$\Leftarrow$) Assume $M(\mathbf{v}) = \mathbf{v}$ and set $\mathbf{f} = F(\mathbf{v})$, $\mathbf{b} = B(\mathbf{v})$. Note that $f_1 = v_1$. Moreover, for $i = 2, \ldots, n$, $f_i \geq f_i \wedge b_i = v_i$, further, $f_i = \varphi_i(f_{i-1}) \wedge v_i \leq v_i$, hence $f_i = v_i$ and $F(\mathbf{v}) = \mathbf{v}$. Analogously, $B(\mathbf{v}) = \mathbf{v}$. Then $F(\mathbf{v}) = \mathbf{v}$, implies that, for $i = 2, \ldots, n$, $v_i = \gamma_i(v_{i-1}) \wedge v_i$, so that $v_i \leq \varphi_i(v_{i-1})$ and $g_i(v_{i-1}, v_i) \leq 0$, hence constraint (2.14) is satisfied. Analogously, $B(\mathbf{v}) = \mathbf{v}$ implies that (2.15) holds.

□

Proposition 6. *Problem 4 is feasible if and only if* $M(\mathbf{v}^+) \geq \mathbf{v}^-$.

Proof. It is a consequence of Propositions 4 and 5. □

Proposition 7. *If Problem 4 is feasible, then* $\mathbf{v}^* = M(\mathbf{v}^+)$ *is its optimal solution.*

Proof. By contradiction, assume that there exists $\tilde{\mathbf{v}}$ such that $\Psi(\tilde{\mathbf{v}}) \leq \Psi(\mathbf{v}^*)$and, by Proposition 5 $M(\tilde{\mathbf{v}}) = \tilde{\mathbf{v}}$. Then, since Ψ is monotonic, $\tilde{\mathbf{v}} \not\leq \mathbf{v}^*$. This is not possible since $\mathbf{v}^* = \bigvee\{\mathbf{x} \in P : \mathbf{x} \leq M(\mathbf{x}),\ \mathbf{x} \leq \mathbf{x}^+\} \geq \tilde{\mathbf{v}}$ by Proposition 3.

□

Remark 1. *It is straightforward from the definition of* M *that the output* $\mathbf{v}^*$ *of Algorithm 1 satisfies* $\mathbf{v}^* = M(\mathbf{v}^+)$.

Algorithm 1 consists in a sequence of three iterations. We call *forward iteration* the iteration representing the function F, *backward iterations* the iteration representing the function B, finally we name *meet iteration* the iteration of minimum component-wise executed by the function M.

The following ones are the main results of the chapter.

Algorithm 1: Solution of Problem 4

input : $\mathbf{v}^-,\ \mathbf{v}^+,\ \varphi_i, , i = 1, \ldots, n,\ \rho_i,\ i = n-1 \ldots, 1$

output: $\mathbf{v}^*$, $Feasible$

$f_1 \leftarrow v_1^+$

for $i \leftarrow 2$ **to** n **do**

  $f_i = \varphi_i(f_{i-1}) \wedge v_i^+$

$b_n \leftarrow v_n^+$

for $i \leftarrow n-1$ **to** 1 **do**

  $b_i = \rho_i(b_{i+1}) \wedge v_i^+$

for $i \leftarrow 1$ **to** n **do**

  $v_i^* = f_i \wedge b_i$

if $\mathbf{v}^* \geq \mathbf{v}^-$ **then**

  $Feasible \leftarrow True$

else

  $Feasible \leftarrow False$

Theorem 1. *The following statements hold:*

i) Problem 4 is feasible if and only if vector $\mathbf{v}^*$*, returned by Algorithm 1, satisfies condition*

$$\mathbf{v}^* \geq \mathbf{v}^- .$$

ii) If Problem 4 is feasible, then vector $\mathbf{v}^*$*, returned by Algorithm 1, is the optimal solution.*

iii) If Problem 4 is feasible, then vector $\mathbf{v}^*$*, returned by Algorithm 1, is the component-wise maximum of the feasible set of Problem 4.*

Proof. Theorem 1 is equivalent to Propositions 6, 7 and 3.

□

Theorem 2. *Algorithm 1 solves Problem 4 with linear time complexity with respect to the number of variables* n*. Moreover, such complexity is optimal.*

Proof. The time complexity of Algorithm 1 is linear with respect to n, since it is composed of three iterations on n. This is optimal in the sense that the time complexity of any algorithm able to solve Problem 4 is bounded from below by the $O(n)$ time required to load the problem data. □

2.3 Solution of the speed planning problem

In this section we show how to apply the results shown in Section 2.2 to Problem 3. To this end we introduce the following proposition.

Proposition 8. *Problem 3 belongs to the class of Problem 4.*

Proof. Problem 3 takes on the form of Problem 4 by setting $v_1^- = v_1^+ = v_s^2$, $v_n^- = v_n^+ = v_f^2$, for $i = 2, \ldots, n-1$, $v_i^- = 0$, $v_i^+ = \bar{u}_i$, $m_1 = 1$, $m_2 = 1$ and $\Psi(\mathbf{v}) = \sum_{i=1}^{n-1} h(v_i, v_{i+1})$ with

$$h(x,y) = \begin{cases} \frac{1}{\sqrt{x}+\sqrt{y}} & \text{if } x \wedge y \leq 0,\ x \vee y > 0 \\ \infty & \text{otherwise} \end{cases}$$

$$g_i(x,y) = y - x - u_B, \qquad \varphi_i(x) = (x + u_B) \wedge u_i, \quad i = 2, \dots, n,$$
$$r_i(x,y) = (-y + x + l_B), \quad \rho_i(y) = (y - l_B) \wedge u_i, \quad i = 1, \dots, n-1\,.$$

Note that the *superiority conditions* (see Assumption 2) are also satisfied. Indeed, being $u_B > 0$ (this is due to the definition of u_B in terms of $a^L_{\max}$, which is assumed to the positive in the initial problem formulation) we have that

$$\varphi_i(x) = x + u_B > x, \quad , i = 1, \dots, n-1$$

In the same way we can prove that $\rho_i(x) > x$, $i = n, \dots, 2$ holds too. □

After we have proved that Problem 3 belongs to the class of Problem 4 we can specialize Algortihm 1 to the case of Problem 3 (see Algorithm 2). In particular,F and B can be written as follows

$$F(\mathbf{u}) = \begin{cases} f_1 = v_s^2 \\ f_i = (f_{i-1} + u_B) \wedge u_i \quad i = 2, \dots, n, \end{cases}$$
$$B(\mathbf{u}) = \begin{cases} b_1 = v_f^2 \\ b_i = (b_{i+1} - l_B) \wedge u_i \quad i = n-1, \dots, 1. \end{cases}$$

Remark 2. *Due to the statements of Theorem 1, we claim that Algorithm 2 is able to check the feasibility of Problem 3 and that, in case the problem is feasible, vector* $\mathbf{w}^* = M(\mathbf{u})$ *represents its optimal solution. Moreover, by Theorem 2 it follows that Algorithm 2 has time-optimal complexity.*

2.4 Examples

Example 1 As a first example consider a continuous curvature path (see [6]) composed of a line segment, a clothoid, a circle arc, a clothoid and a final line segment (see Figure 2.2). The minimum-time speed planning on this path, whose total length is $s_f = 90$ m (see Problem 3), is addressed with the following

Algorithm 2: Minimum-time speed planning

input : $v_s^2,\ v_f^2,\ u_B,\ l_B,\ u_i,\ i = 1, \ldots, n$

output: $\mathbf{w}^*,\ Feasible$

$f_1 \leftarrow v_s^2$

for $i \leftarrow 2$ **to** n **do**

  $f_i = \min\left\{f_{i-1} + u_B, u_i\right\}$

$b_n \leftarrow v_{\mathrm{f}}$

for $i \leftarrow n-1$ **to** 1 **do**

  $b_i = \min\left\{b_{i+1} - l_B, u_i\right\}$

for $i \leftarrow 1$ **to** n **do**

  $w_i^* = \min\left\{f_i, b_i\right\}$

if $v_1^* = v_s^2$ *and* $v_n^* = v_f^2$ **then**

  $Feasible \leftarrow True$

else

  $Feasible \leftarrow False$

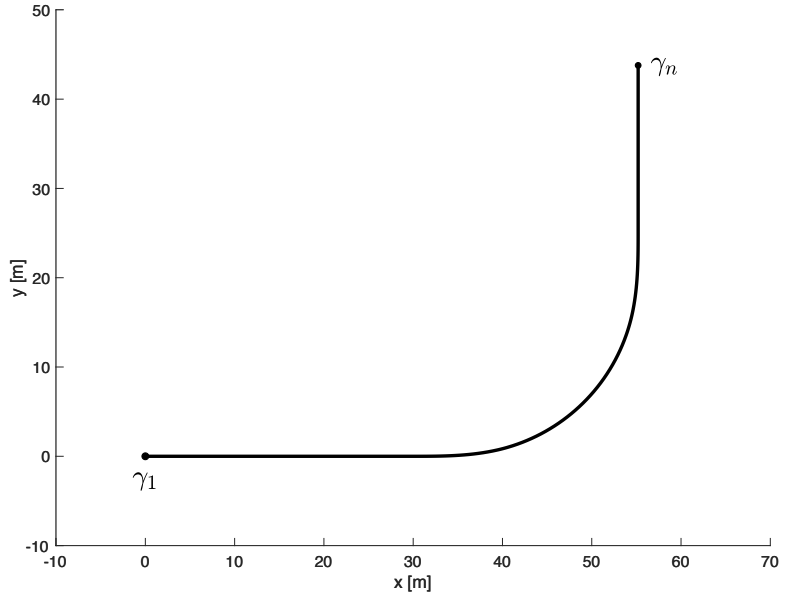


Figure 2.2: Example 1: a simple continuous curvature path.

data. The initial and final velocities are $v_s = 4$ ms^{-1} and $v_f = 2$ms^{-1}, the maximal speed is $v_{\max} = 15$ ms^{-1}, the longitudinal acceleration limits are $a^L_{\min} = -1.5$ ms^{-2} and $a^L_{\max} = 1.5$ ms^{-2}, and the maximal normal acceleration is $a^N_{\max} = 1$ ms^{-2}.

The number of samples is chosen as $n = 500$. Figures 2.3-2.4 and 2.5 show the corresponding functions F, B and M computed as the solution of the *forward*, *backward*, and *meet iteration* respectively. Figure 2.6 shows the final optimal speed profile computed through the *meet operator* $\mathbf{w}^* = M(\mathbf{u})$. This solution corresponds to the minimum-time $t^*_f = 16.53$ s. Note that the vehicle starts from speed v_s and accelerates to a local maximum speed. Then, it slows down before the beginning of the curve, in order to respect the maximum speed constraint due to the lateral acceleration on the curve. At the end of the curve, the vehicles accelerates and reaches a second local maximum speed after which it decelerates in order to reach the final speed v_f.

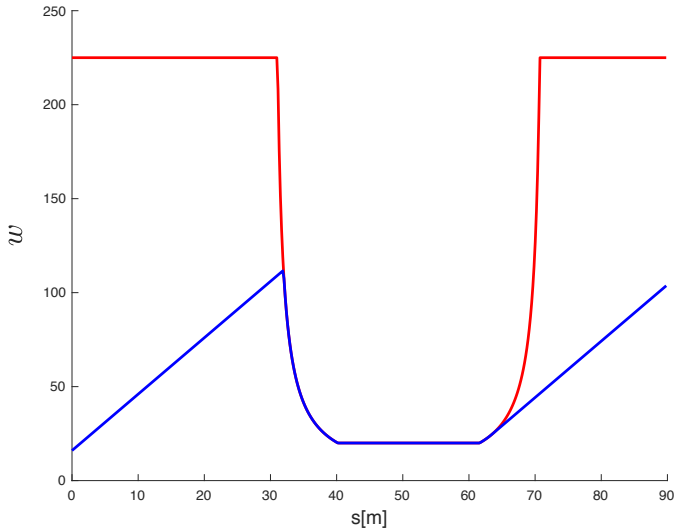


Figure 2.3: Example 1 (*Forward iteration*): The red line represents the upper-bound $\mathbf{u}$ allowed along the path while the blue line represents the square speed sequence $f_1, \ldots, f_n$ computed after *Forward iteration* F.

Example 2 Consider the same path and constraints as in Example 1, with different initial and final conditions: $v_s = 4$ ms^{-1}, $v_f = 15$ ms $^{-1}$. Figure 2.7 shows vector $\mathbf{w}^*$ obtained with Algorithm 2. In this case, Problem 3 is unfeasible by Theorem 1, being $w_n^* \neq v_f^2$. In fact, the allowed maximum longitudinal acceleration is not sufficient to reach the final condition on speed.

Example 3 As a third example, consider a speed planning on a G^2-path composed with $\boldsymbol{\eta}^2$-splines [5] (see Figure 2.8). A single $\boldsymbol{\eta}^2$-spline is a quintic polynomial curve that can interpolate given Cartesian points with associated unit tangent vectors and curvatures. The path in Figure 2.8 is composed by three $\boldsymbol{\eta}^2$-splines whose interpolating data is reported in Table 2.1: the θ's are the angles between the x-axis and the unit tangent vectors and the k's are the curvatures. The $\boldsymbol{\eta}$ parameters of these splines, which are free parameters to

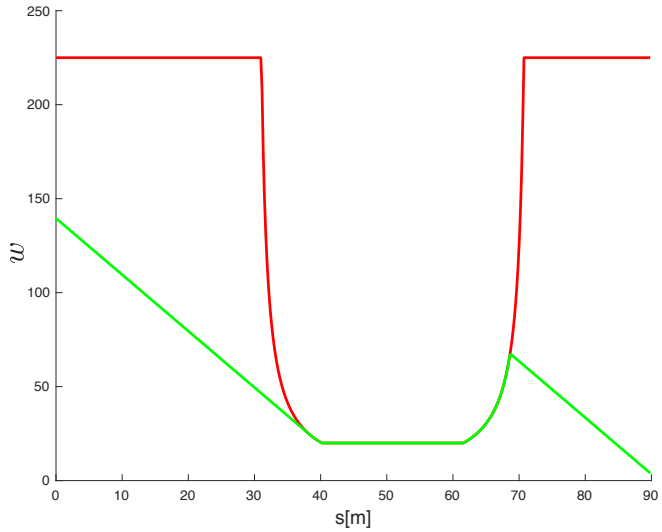


Figure 2.4: Example 1 (*Backward iteration*): The red line represents the upperbound **u** allowed along the path while the green line represents the square speed sequence $b_1, \ldots, b_n$ computed after *Backward iteration* B.

shape the spline path without affecting the interpolating conditions, are chosen as $\eta_1 = \eta_2 = 50$ and $\eta_3 = \eta_4 = 0$.

This planned path, which has total length $s_f = 153.05$ m, is composed of a lane-change path, an approximate clothoid, and an approximate circle arc (first, second, and third spline respectively). The speed planning is addressed with $v_s = v_f = 0$, a maximal speed $v_{\max} = 36.1$ ms^{-1}, longitudinal acceleration limits $a^L_{\min} = -10.5$ ms^{-2}, $a^L_{\max} = 4$ ms^{-2}, and maximal normal acceleration $a^N_{\max} = 7$ ms^{-2}. Algorithm 2 is applied with $n = 100$ achieving the minimum-time $t^*_f = 11.35$ s. The resulting optimal speed profile is plotted in Figure 2.9.

Computational experiments In this paragraph we want to evaluate the performance of Algorithm 2. In Section 2.3 we showed that Problem 3 belongs to the class of Problem 4. Since Problem 4 requires a decreasing objec-

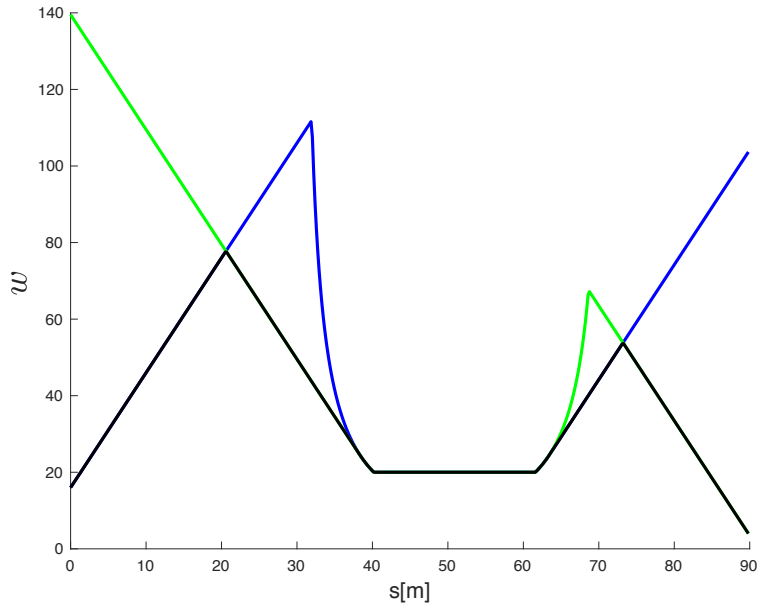


Figure 2.5: Example 1 (*Meet iteration*): In black the output of Algorithm 2 (the optimal solution $\mathbf{b}^*$) is shown starting from the sequences $f_1, \ldots, f_n$ and $b_1, \ldots, b_n$ previously computed.

Table 2.1: Interpolating data of the G^2-path.

	x m	y m	θ rad	k m^{-1}
p_A	0	0	0	0
p_B	50.00	15.00	0	0
p_C	98.76	23.19	0.50	1/50
p_D	124.67	63.53	1.50	1/50

tive function with respect the decision variables, we can substitute the convex function (2.7) with the following linear one: $\Psi(\mathbf{w}) = -\sum_i^n w_i$. This change of objective function allows us to use a linear solver (for instance Gurobi [50])

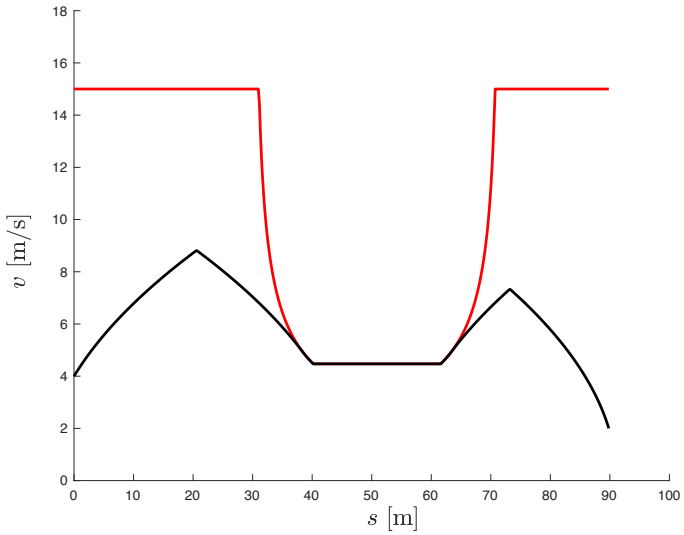


Figure 2.6: Example 1 (*Final result*): The red line represents the maximum speed allowed along the path while the black line is the approximated optimal solution the minimum time speed planning problem

to compute the optimal solution of Problem 3. Using the same experimental data of Example 2.4, we find the solution to Problem 3 with two different approaches:

- using the LP solver Gurobi [50] which solves the LP reformulation of Problem 3.
- using a C++ implementation of Algorithm 2.

The results are presented in Figure 2.10 for values of n from 100 to 100,000. Such results show that Algorithm 2 outperforms the commercial LP solver used. We measured the performance on a Macbook Pro equipped with an 2.3 GHz Intel Core i5.

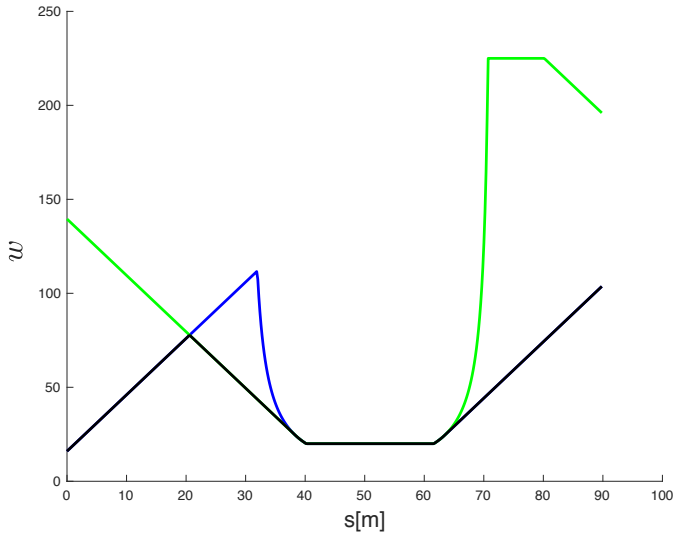


Figure 2.7: Example 2 (*Join iteration*): In black the output of Algorithm 2 is shown starting from the sequences $f_1, \ldots, f_n$ and $b_1, \ldots, b_n$ previously computed. The final speed condition is not satisfied $w_n^* \neq v_f^2$.

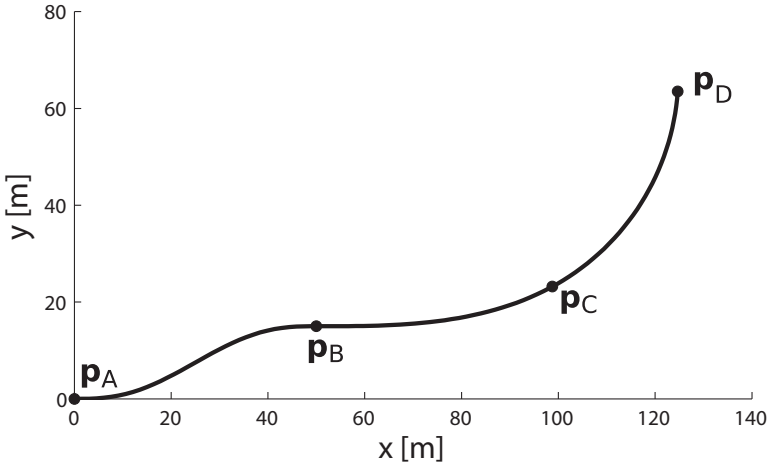


Figure 2.8: Example 3: A G^2-path composed of $\boldsymbol{\eta}^2$-splines [5].

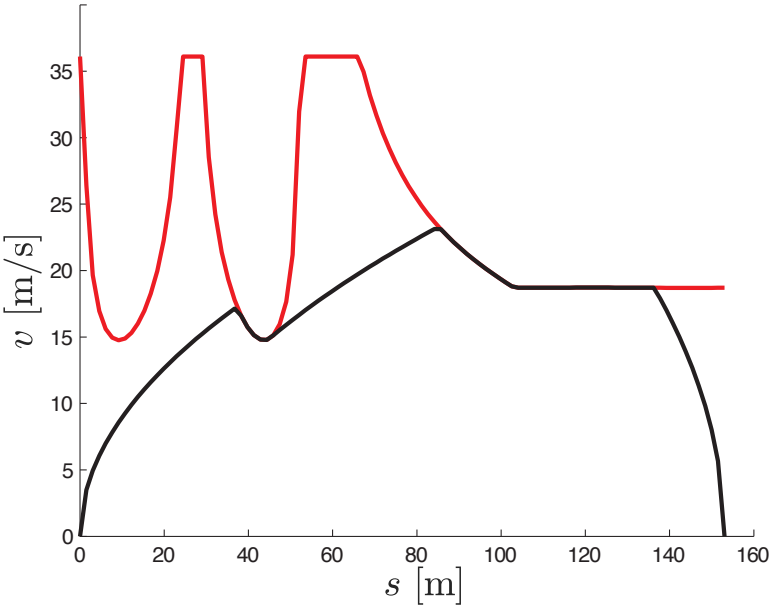


Figure 2.9: Example 3: The red line represents the maximum speed allowed along the path and the black line plots the optimal speed profile.

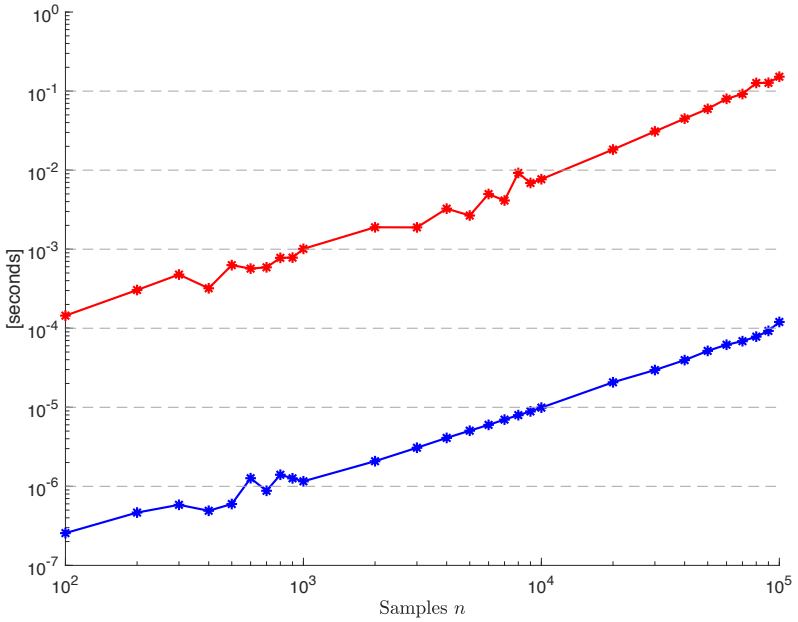


Figure 2.10: Computational Results: the red line represents the Gurobi's computational time, while the blue one represents the computational time of Algorithm 2

Chapter 3

A solution of the minimum-time continuous speed planning problem based on lattice theory

In Chapter 2 we found an approximated solution to a minimum time speed planning problem for an autonomous vehicle. In particular, we exploited the arc-length parametrization and a change of variables to formulate a convex problem. This problem has been discretized and efficiently solved using a suitable optimal time-complexity algorithm.

The aim of this chapter is to show that the same reasoning based on the lattice theory which we adopted in Chapter 2 can be used to find the continuous time solution to the speed planning problem. We present a necessary and sufficient condition for the feasibility of the problem and a simple operator, based on the solution of two ordinary differential equations, which computes the optimal solution. Theoretically, we show that the problem feasible set, if not empty, is a lattice, whose supremum element corresponds to the optimal solution.

3.1 Problem formulation

Following the same reasoning used in Section 2.1 we recall that the minimum time speed planning problem for an autonomous vehicle can be formulated as follows

Problem 5.

$$\min_{w\in W^{1,\infty}([0,s_f])} \int_0^{s_f} w(s)^{-\frac{1}{2}} ds \tag{3.1a}$$

$$\mu^-(s) < w(s) \leq \mu^+(s), \qquad s \in [0, s_f], \tag{3.1b}$$

$$\alpha^-(s) \leq w'(s) \leq \alpha^+(s), \qquad s \in [0, s_f], \tag{3.1c}$$

where

$$\mu^+(s) = v^+(s)^2 \wedge \frac{\beta(s)}{k(s)}, \qquad \mu^-(s) = v^-(s)^2 \tag{3.2}$$

represent the upper bound of w, (depending on the speed bound v^+ and the curvature k) and the lower bound of w, respectively. Functions α^- α^+ are the functions which come from the bounds imposed on the longitudinal acceleration of the vehicle. Note that, in a more general way with respect to Chapter 2 each bound is considered path dependent.

In this chapter, we actually address the following problem, which is slightly more general than Problem 5,

Problem 6.

$$\min_{w\in W^{1,\infty}([0,s_f])} \Psi(w) \tag{3.3a}$$

$$\mu^-(s) < w(s) \leq \mu^+(s), \qquad s \in [0, s_f], \tag{3.3b}$$

$$\alpha^-(s) \leq w'(s) \leq \alpha^+(s), \qquad s \in [0, s_f], \tag{3.3c}$$

where $\Psi : W^{1,\infty}([0, s_f]) \to \mathbb{R}$ is order reversing (i.e.,$(\forall x, y \in W^{1,\infty}([0, s_f]))$ $x \geq y \Rightarrow \Psi(x) \leq \Psi(y)$) and μ^-, μ^+, α^-, $\alpha^+ \in L^\infty([0, s_f])$ are assigned functions with $\mu^-, \alpha^+ \geq 0$, $\alpha^- \leq 0$. Note that the objective function (3.1a) is order

reversing, so that Problem 5 has the form of Problem 6. We consider a generic order-reversing function Ψ as objective function for the sake of generality. Consider the following:

Definition 3. *Let $\mathcal{Q}$ be the subset of $W^{1,\infty}([0, s_f,])$ such that $\mu \in \mathcal{Q}$ if* $\operatorname{sign}(\mu' - \alpha^+)$ *and* $\operatorname{sign}(\mu' - \alpha^-)$ *are Riemann integrable (i.e., in view of the boundedness of the* sign *function, a. e. continuous), where* $\operatorname{sign} : \mathbb{R} \to \{-1, 0, 1\}$ *is defined as*

$$\operatorname{sign}(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0. \end{cases}$$

3.2 A feasibility condition and the optimal solution

Define the *forward operator* $F : \mathcal{Q} \to W^{1,\infty}([0, s_f])$ such that $F(\mu) = \phi$, where ϕ is the solution of the following differential equation

$$\begin{cases} \phi'(x) = f(x, \phi) = \begin{cases} \alpha^+(x) \wedge \mu'(x), & \text{if } \phi(x) \geq \mu(x) \\ \alpha^+(x), & \text{if } \phi(x) < \mu(x) \end{cases} \\ \phi(0) = \mu(0)\,. \end{cases} \tag{3.4}$$

Note that the solution of (3.4) exists and is unique by Theorem 1 in Chapter 2, Section 10 of [51], since function f is bounded on $[0, s_f]$, the subset of $[0, s_f] \times \mathbb{R}$ in which f is discontinuous has zero measure and,

$$(\forall x \in [0, s_f])\,, (\forall u, y \in \mathbb{R})\,, (u - y)(f(x, u) - f(x, y)) \leq 0.$$

Conversely, define the *backward operator* $B : \mathcal{Q} \to W^{1,\infty}([0, s_f])$, such that $B(\mu) = \phi$, where ϕ is the solution of

$$\begin{cases} \phi'(x) = \begin{cases} \alpha^-(x) \vee \mu'(x), & \text{if } \phi(x) \geq \mu(x) \\ \alpha^-(x), & \text{if } \phi(x) < \mu(x) \end{cases} \\ \phi(s_f) = \mu(s_f)\,, \end{cases} \tag{3.5}$$

whose existence and uniqueness hold for the same reasons as (3.4).

Finally, define the *meet operator* $M : \mathcal{Q} \to W^{1,\infty}([0, s_f])$ as

$$M(\mu) = F(\mu) \wedge B(\mu). \tag{3.6}$$

We claim that the *meet operator* M allows checking the feasibility of Problem 6 and that, in case Problem 6 is feasible, function $w^* = M(\mu^+)$ represents its optimal solution.

Namely, the following is the main result of this chapter.

Theorem 3. *Let $\mu^+ \in \mathcal{Q}$, then the following statements hold:*

i. Problem 6 is feasible if and only if function $w^* = M(\mu^+)$ satisfies

$$w^* \geq \mu^- .$$

ii. If Problem 6 is feasible, then function $w^* = M(\mu^+)$ is its optimal solution.

Proofs of the results. Part *i.* follows from Proposition 16, part *ii.* follows from Proposition 17 (see Section 3.4).

Remark 3. *For special choices of α^+, α^-, μ, differential equations* (3.4) *and* (3.5) *may have a closed-form solution. Anyway, in the general case,* (3.4) *and* (3.5) *need be numerically solved. Using a constant step-size method (such as Euler's or a non-adaptive Runge-Kutta method), the computation time for the numerical solution of* (3.4) *or* (3.5) *is linear with respect to the number of steps. Roughly speaking, the method presented in Chapter 2 corresponds to the solution of* (3.4) *and* (3.5) *with Euler's method. Since Euler's method is a first-order method, the solution of* (3.4) *and* (3.5) *with an higher order method can be more efficient than the algorithm presented in Chapter 2.*

Further, one can solve (3.4) *and* (3.5) *with an adaptive step-size integration method. In general, with respect to fixed step-size methods, adaptive step-size methods attain the same precision at a reduced computational effort. Further,*

they do not require a preliminary choice of the integration step-size, since this parameter is adaptively tuned. In other words, with a careful choice of the method used for the numerical solution of (3.4) *and* (3.5)*, they can be solved in an efficient way. However, a more in-depth discussion of the numerical solution of* (3.4) *and* (3.5) *is outside the scope of this chapter.*

Remark 4. *As we showed, our algorithm is simpler than [25]. In fact, we need only one forward and one backward integration step, while the reference [25] proposes a semi-analytical solution that requires multiple integration operations. However, note that the constraints considered in [25] are slightly different from ours. Moreover, the method we present is more general than the one in [26, 27], since we consider generic paths and not a concatenation of clothoids. Further, differently from [26, 27], our algorithm does not need a preprocessing phase, needed to identify specific points or segments on the path curvature and the speed upper bound functions.*

3.3 Examples

Example 1 As a first example consider the path shown in Figure 3.1, whose curvature is defined as

$$k(s) = \begin{cases} 0, & \text{if } s \in [0, l_1) \\ k_\tau(s), & \text{if } s \in [l_1, l_2] \\ 1/R, & \text{if } s \in (l_2, l_3) \\ k_\tau(s), & \text{if } s \in [l_3, l_4] \\ 0, & \text{if } s \in (l_4, s_f] \end{cases} \tag{3.7}$$

where $k_\tau(s)$ is the 6-th degree Hermite polynomial used to guarantee the following interpolation conditions:

$$\begin{aligned} &k(l_1) = k(l_4) = 0, \\ &k(l_2) = k(l_3) = 1/R, \\ &k'(l_i) = k''(l_i) = 0, \qquad i = 1, \dots, 4. \end{aligned}$$

In this example, the total length is $s_f = 200$ m and the minimum-time speed

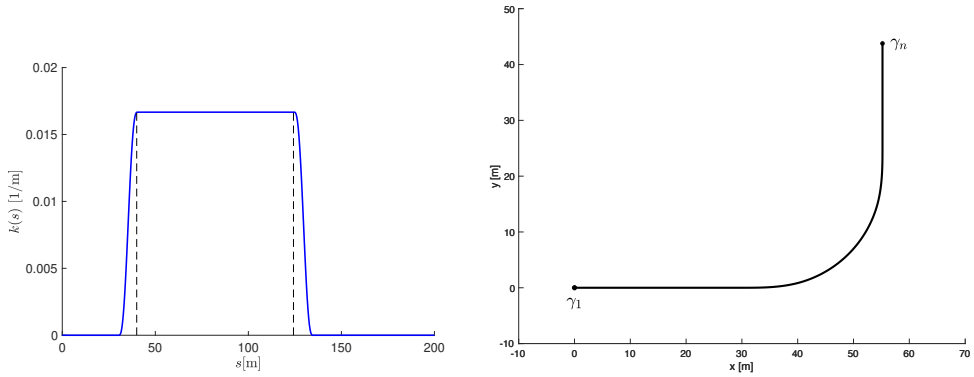


Figure 3.1: On the left, the blue line represent the curvature function k in (3.7) of the curve discussed in the first example. On the right, the black line represents the path from starting point γ_1 to the end point γ_n.

planning problem is addressed with $[l_1, l_2, l_3, l_4] = [30,\ 40,\ 124.2478,\ 134.2478]$, $R = 60$ m. The speed bounds v^+ and v^- are set as follows: $v^+(0) = v^-(0) = 0$, $v^+(s_f) = v^-(s_f) = 22$, while, for each $s \in (0, s_f)$, $v^-(s) = 0$ and $v^+(s) = 36.1$ ms^{-1}. The longitudinal acceleration limits are $\alpha^- = -10.5$ ms^{-2}and $\alpha^+ = 4$ ms^{-2}, and the maximal normal acceleration is $\beta = 7$ ms^{-2}.

The following results are obtained by numerically solving equations (3.4), (3.5) with a standard Runge-Kutta 45 integration scheme. Figure 3.2 shows the upper-bound function μ^+ obtained by (3.2) and the corresponding functions $F(u)$ and $B(u)$ computed as the solution of equations (3.4) and (3.5), respectively. Figure 3.3 shows the optimal solution w^* obtained by (3.6). In this

example, the vehicle starts with zero speed and accelerates to the upper bound. Then, it follows the speed bound in order to respect the maximum speed constraint due to the lateral acceleration on the curve. After that, at the end of the constant bound, the vehicle accelerates and reaches a second local maximum speed after which it decelerates quickly in order to reach the final speed $v^+(s_f)$.

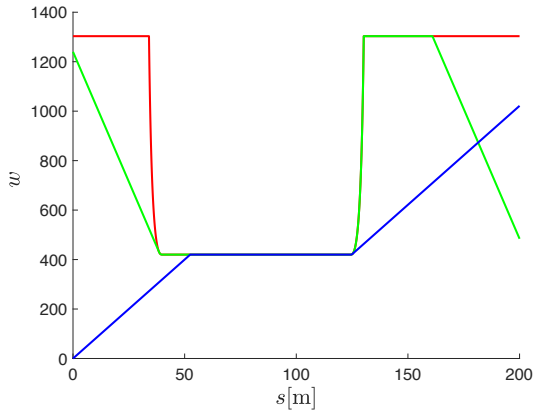


Figure 3.2: Example 1: The red line represents function μ^+ defined in (3.2), the blue line represents $F(\mu^+)$ while the green line represents $B(\mu^+)$.

Example 2 As a second example, consider the same path and constraints as in the first example, with different initial and final conditions: $v^-(0) = v^+(0) = 0$ ms^{-1}, $v^-(s_f) = v^+(s_f) = 35$ ms^{-1}. Figure 3.4 shows function w^* obtained by (3.6). In this case, Problem (3.3) is unfeasible by Theorem 3, being $w^*(s_f) < v^-(s_f)^2$. In fact, the allowed maximum longitudinal acceleration is not sufficient to reach the final condition on speed.

Example 3 As a third example, consider a curve obtained by a quintic polynomial curve which interpolates coordinates $x = [0,\ 2,\ 2.60,\ 1.75,\ 3]$, $y = [0,\ -0.5,\ 0,\ 2,\ 3]$ (see Figure 3.5). The speed planning is addressed with

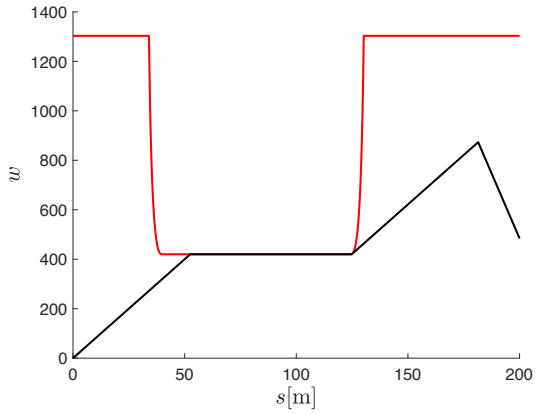


Figure 3.3: Example 1: The red line represents μ^+ defined in (3.2), the black line represents the optimal solution $w^* = M(\mu^+)$.

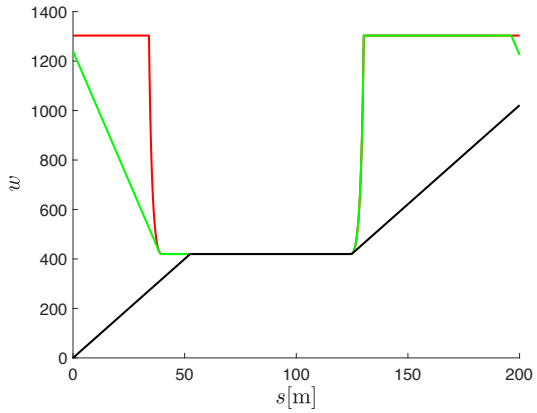


Figure 3.4: Example 2. The green line represents the function $B(u)(s)$ while the black one depicts function $w^* = M(\mu^+)$. The final condition is not satisfied: $w^*(s_f) \neq v_f^2$.

$v^+(0) = v^-(0) = 0$, $v^+(s_f) = v^-(s_f) = 0$ and with $v^-(s) = 0$ and $v^+(s) = 1.3$ ms^{-1} for each $s \in (0, s_f)$. The longitudinal acceleration limits are $\alpha^- = -0.1$ ms^{-2}, $\alpha^+ = 0.1$ ms^{-2}, and the maximal normal acceleration $\beta = 0.05$ ms^{-2}.

The resulting optimal speed profile is plotted in Figure 3.6.

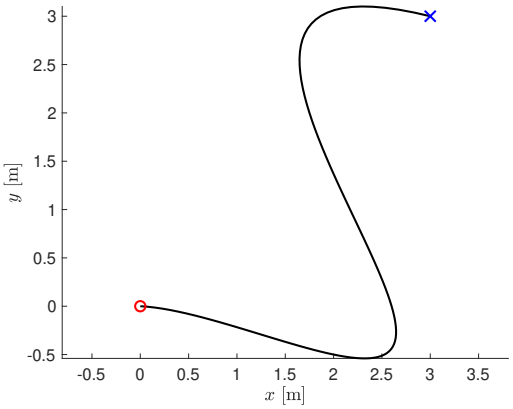


Figure 3.5: A path obtained by quintic-splines interpolation. The black line represents the path while the circle and the cross represent the start and the end point, respectively.

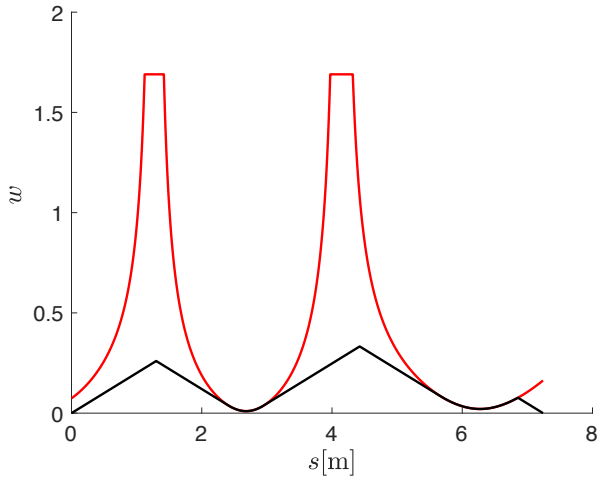


Figure 3.6: The red line represents function μ^+ defined in (3.2), while the black line is the optimal profile $w^* = M(\mu^+)$.

3.4 Proofs

Given the interval $I = [0, s_f]$, let

$$P = \left\{u \in L^\infty(I) :\ 0 \leq u(s) \leq \left\|\mu^+\right\|_\infty \text{ for almost every } s \in I\right\}$$

Note that $\langle P; \vee, \wedge\rangle$ is a complete lattice. Hence, for each subset $S \subseteq P$, there exists a unique least upper bound $u \in P$, such that,

$$(\forall v \in P)(\forall w \in S)\ w \leq v \iff u \leq v.$$

The least upper bound of S is denoted by $\bigvee S$. Dually, it is possible to define the greatest lower bound of $S \subseteq P$, denoted by $\bigwedge S$ (see Definitions 2.1, 2.4 and Notation 2.3 on pages 33-34 of [49]).

Given function $\chi : \mathbb{R} \to \{0, 1\}$ defined as follows

$$\chi(s) = \begin{cases} 1, & \text{if } \ s \geq 0 \\ 0, & \text{otherwise,} \end{cases}$$

let us define for all $x, y \in I$, function $A : I \times I \to \mathbb{R}$ as

$$A(x, y) = \int_x^y \left\{\alpha^+(\xi)\chi(y - x) + \alpha^-(\xi)\chi(x - y)\right\} d\xi. \tag{3.8}$$

Define operators $\bar{F}, \bar{B}, \bar{M} : P \to P$, such that, for $\mu \in P$, $\bar{F}(\mu)$ and $\bar{B}(\mu)$ are given as follows

$$\begin{cases} \bar{F}(\mu)(x) = \bigwedge\limits_{y \leq x} \{\mu(y) + A(y, x)\} \\ \bar{F}(\mu)(0) = \mu(0), \end{cases}$$

$$\begin{cases} \bar{B}(\mu)(x) = \bigwedge\limits_{y \geq x} \{\mu(y) + A(y, x)\} \\ \bar{B}(\mu)(s_f) = \mu(s_f). \end{cases}$$

and $\bar{M}(\mu) = \bar{F}(\mu) \wedge \bar{B}(\mu)$. Observe that $(\forall x \in I)$,

$$\bar{M}(\mu)(x) = \bigwedge_{y \leq x} \{\mu(y) + A(y, x)\} \wedge \bigwedge_{y \geq x} \{\mu(y) + A(y, x)\} = \bigwedge_{y \in I} \{\mu(y) + A(y, x)\}. \tag{3.9}$$

As we will show in Proposition 14, the operators we have just introduced are extensions of those defined in (3.4), (3.5) and (3.6), respectively.

We will refer to the following definitions.

Definition 4. *An operator $\phi : P \to P$ is meet preserving if,*

$$(\forall u, v \in P)\ \phi(u \wedge v) = \phi(u) \wedge \phi(v).$$

Definition 5. *An operator $\phi : P \to P$ is order preserving if, ,*

$$(\forall u, v \in P)\ u \leq v \Rightarrow \phi(u) \leq \phi(v).$$

Even though we will only use the fact that operators $\bar{F}, \bar{B}$ and $\bar{M}$ are order preserving, for completeness we state the following Proposition.

Proposition 9. *Operators $\bar{F}$, $\bar{B}$ and $\bar{M}$ are meet preserving and order preserving.*

Proof. Let $u, v \in P$, $w = u \wedge v$. We want to show that $\bar{F}(w) = \bar{F}(u) \wedge \bar{F}(v)$. By definition of $\bar{F}$ we have that $\bar{F}(w)(0) = \bar{F}(u)(0) \wedge \bar{F}(v)(0)$ and $(\forall x \in I)$,

$$\begin{aligned}
\bar{F}(w)(x) &= \bigwedge_{y \leq x} \{w(y) + A(y,x)\} = \bigwedge_{y \leq x} \{(u \wedge v)(y) + A(y,x)\} = \\
&= \bigwedge_{y \leq x} \{u(y) \wedge v(y) + A(y,x)\} = \\
&= \bigwedge_{y \leq x} \{u(y) + A(y,x)\} \wedge \bigwedge_{y \leq x} \{v(y) + A(y,x)\} = \\
&= \bar{F}(u)(x) \wedge \bar{F}(v)(x).
\end{aligned}$$

The proof that $\bar{B}(u \wedge v) = \bar{B}(u) \wedge \bar{B}(v)$ is analogous. Finally, $\bar{M}(u \wedge v) = \bar{F}(u \wedge v) \wedge \bar{B}(u \wedge v) = \bar{F}(u) \wedge \bar{F}(v) \wedge \bar{B}(u) \wedge \bar{B}(v) = \bar{F}(u) \wedge \bar{B}(u) \wedge \bar{F}(v) \wedge \bar{B}(v) = \bar{M}(u) \wedge \bar{M}(v)$. Since maps $\bar{F}, \bar{B}, \bar{M}$ are meet preserving, they are also order preserving (see Proposition 2.19 on page 44 of [49]). □

Proposition 10. *Function $A : I \times I \to \mathbb{R}$ defined as in* (3.8) *is a hemi-metric, that is, it satisfies the following properties:*

i. $(\forall x, y \in I)\ A(x, y) \geq 0$,

ii. $(\forall x \in I)\ A(x, x) = 0$,

iii. $(\forall x, y, z \in I)\ A(x, z) \leq A(x, y) + A(y, z)$ *(i.e., the triangular inequality holds).*
Moreover, equality holds if $x \geq y \geq z$ *or* $x \leq y \leq z$.

Proof. *i.* It holds, since α^+ is non-negative and α^- is non-positive over I.

ii. It holds trivially by definition of A.

iii. For $y \geq z \geq x$:

$$
\begin{aligned}
A(x,z) = &\int_x^z \left\{\alpha^+(\xi)\chi(z-x) + \alpha^-(\xi)\chi(x-z)\right\} d\xi = \int_x^z \alpha^+(\xi) d\xi \leq \\
&\leq \int_x^y \alpha^+(\xi) d\xi \leq \int_x^y \alpha^+(\xi) d\xi - \int_z^y \alpha^-(\eta) d\eta = \\
&= \int_x^y \alpha^+(\xi) d\xi + \int_y^z \alpha^-(\eta) d\eta = \\
&= \int_x^y \left\{\alpha^+(\xi)\chi(y-x) + \alpha^-(\xi)\chi(x-y)\right\} d\xi + \\
&+ \int_y^z \left\{\alpha^+(\eta)\chi(z-y) + \alpha^-(\eta)\chi(y-z)\right\} d\eta = \\
&= A(x,y) + A(y,z).
\end{aligned}
$$

The same reasoning applies also to the case in which $z \geq x \geq y$, $x \geq z \geq y$

or $y \geq x \geq z$. Next, let us show that equality holds for any $x \leq y \leq z$:

$$\begin{aligned} A(x,z) = &\int_x^z \left\{\alpha^+(\xi)\chi(z-x) + \alpha^-(\xi)\chi(x-z)\right\} d\xi = \\ &= \int_x^z \alpha^+(\xi) d\xi = \int_x^y \alpha^+(\xi) d\xi + \int_y^z \alpha^+(\eta) d\eta = \\ &= \int_x^y \left\{\alpha^+(\xi)\chi(y-x) + \alpha^-(\xi)\chi(x-y)\right\} d\xi + \\ &+ \int_y^z \left\{\alpha^+(\eta)\chi(z-y) + \alpha^-(\eta)\chi(y-z)\right\} d\eta = \\ &= A(x,y) + A(y,z). \end{aligned}$$

The proof that the equality holds also for any $x \geq y \geq z$ is analogous.

□

Proposition 11. *Function $\bar{M}$ satisfies the following properties, $(\forall \mu \in P)$,*

i. $\bar{M}(\mu) \leq \mu$,

ii. $\bar{M}^2(\mu) = \bar{M}(\mu)$, *where* $\bar{M}^2(\mu)$ *stands for* $\bar{M}(\bar{M}(\mu))$.

Proof. *i.* It is a consequence of the definition of $\bar{M}$.

ii. Let us now show that $\bar{F}(\bar{M}(\mu)) = \bar{M}(\mu)$: the fact that $\bar{F}(\bar{M}(\mu)) \leq \bar{M}(\mu)$ follows from the definition of $\bar{F}$ whilst, to prove the opposite inequality, note that, by Proposition 10 and (3.9),

$$\begin{aligned} \bar{F}(\bar{M}(\mu))(x) = &\bigwedge_{y \leq x} \left\{\bar{M}(\mu)(y) + A(y,x)\right\} = \\ &= \bigwedge_{y \leq x} \left\{ \bigwedge_{z \in I} \left\{\mu(z) + A(z,y)\right\} + A(y,x) \right\} \geq \\ &\geq \bigwedge_{z \in I} \left\{\mu(z) + A(z,x)\right\} = \bar{M}(\mu)(x). \end{aligned}$$

In the same way it can be proved that $\bar{B}(\bar{M}(\mu)) = \bar{M}(\mu)$, from which it follows that $\bar{M}(\bar{M}(\mu)) = \bar{M}(\mu)$.

□

Proposition 12.

$$\bar{M}(\mu^+) = \bigvee \left\{ u \in P : u \leq \bar{M}(u), u \leq \mu^+ \right\}$$

Proof. Set $U = \{u \in P : u \leq \mu^+\}$. Note that $\langle U; \vee, \wedge \rangle$ is a sublattice of $\langle P; \vee, \wedge \rangle$, moreover, by *i.* of Proposition 11, if $u \in U$, then $\bar{M}(u) \in U$. Since $\bar{M}$ is order preserving by Proposition 9, by the Knaster-Tarski Fixpoint Theorem (Theorem 2.35 on page 50 in [49])

$$u^* = \bigvee \left\{ u \in P : u \leq \bar{M}(u), u \leq \mu^+ \right\}$$

is such that u^* is the greatest fixed point of $\bar{M}$ such that $u^* \in U$. Let $u = \bar{M}(\mu^+)$, by part *ii.* of Proposition 11, we know that u is also a fixed point of $\bar{M}$, thus, by definition of u^*, $u^* \geq u$. To prove that $u^* = u$, that is, to prove that u is also the greatest fixed point, it remains to show that $u^* \leq u$. To this end, assume, by contradiction, that $u^* \not\leq u$. Since $u^* = \bar{M}(u^*)$, $u = \bar{M}(\mu^+)$ and the fact that $\bar{M}$ is order preserving, it follows that $u^* \not\leq \mu^+$, which contradicts the definition of u^* . □

Remark 5. *Given $u, v \in P$, if $u \not\leq v$, this does not imply that $u \geq v$ and $u \neq v$, as u and v may not be comparable with respect to partial order $\leq$.*

Proposition 13. *The following two statements are equivalent:*

i. Set $\left\{ u \in P : u = \bar{M}(u),\ \mu^- \leq u \leq \mu^+ \right\}$ is not empty.

ii. $\bar{M}(\mu^+) \geq \mu^-$.

Proof.

ii. $\Rightarrow$ *i.*) It follows from the fact that $u^* = \bar{M}(\mu^+)$ is such that $\bar{M}(u^*) = u^*$ by part *ii.* of Proposition 11.

i. $\Rightarrow$ *ii.*) By contradiction, assume that $\bar{M}(\mu^+) \not\geq \mu^-$. Choose any $w \in P$ such that $w = \bar{M}(w)$ and $w \leq \mu^+$. By Proposition 12, $\bar{M}(w) \leq \bar{M}(\mu^+)$, hence $w \leq \bar{M}(\mu^+)$, but $\bar{M}(\mu^+) \not\geq \mu^-$, so $w \not\geq \mu^-$. Thus, being w any fixed point of $\bar{M}$ such that $w \leq \mu^+$, set $\left\{u \in P : u = \bar{M}(u),\ \mu^- \leq u \leq \mu^+\right\}$ is empty .

□

Proposition 14. *If $\mu \in \mathcal{Q}$, then $\bar{F}(\mu), \bar{B}(\mu) \in W^{1,\infty}(I)$ satisfy a. e.*

$$\begin{cases} \bar{F}(\mu)'(x) = \begin{cases} \alpha^+(x) \wedge \mu'(x), & \text{if } \bar{F}(\mu)(x) \geq \mu(x) \\ \alpha^+(x), & \text{if } \bar{F}(\mu)(x) < \mu(x) \end{cases} \\ \bar{F}(\mu)(0) = \mu(0), \end{cases} \tag{3.10}$$

and

$$\begin{cases} \bar{B}(\mu)'(x) = \begin{cases} \alpha^-(x) \vee \mu'(x), & \text{if } \bar{B}(\mu)(x) \geq \mu(x) \\ \alpha^-(x), & \text{if } \bar{B}(\mu)(x) < \mu(x) \end{cases} \\ \bar{B}(\mu)(s_f) = \mu(s_f). \end{cases} \tag{3.11}$$

Proof. Let $J = \{x \in I : \text{sign}(\mu' - \alpha^+)$ is continuous at $x\}$. Note that, since $\mu \in \mathcal{Q}$, J contains almost all elements of I. Let $x \in J$, then

$$\begin{aligned} &\lim_{h\to 0^+} \tfrac{\bar{F}(\mu)(x+h) - \bar{F}(\mu)(x)}{h} = \\ &= \lim_{h\to 0^+} h^{-1} \left[\bigwedge_{y \leq x+h} \{\mu(y) + A(y, x+h)\} - \bar{F}(\mu)(x) \right] = \\ &= \lim_{h\to 0^+} h^{-1} \left[\left(\bigwedge_{y \leq x} \{\mu(y) + A(y, x+h)\} - \bar{F}(\mu)(x) \right) \wedge \right. \\ &\left. \wedge \left(\bigwedge_{x < y \leq x+h} \{\mu(y) + A(y, x+h)\} - \bar{F}(\mu)(x) \right) \right] \end{aligned}$$

Since $A(y, x+h) = A(y, x)+A(x, x+h)$ by Proposition 10, the first parenthesis of (3.4) reduces to

$$\bigwedge_{y \leq x} \{\mu(y)+A(y, x+h)\}-\bar{F}(\mu)(x) = \bar{F}(\mu)(x)+A(x, x+h)-\bar{F}(\mu)(x) = A(x, x+h).$$

Being $\text{sign}(\mu' - \alpha^+)$ continuous at x, it is possible to choose $h > 0$ sufficiently small such that $\text{sign}(\mu' - \alpha^+)$ is constant on interval $[x, x + h]$. Set

$$J^+ = \left\{x \in J : \mu'(x) - \alpha^+(x) > 0\right\}$$

and $J^- = J \setminus J^+$. Then, the second parenthesis of (3.4) can be rewritten as

$$\bigwedge_{x < y \leq x+h} \{\mu(y) + A(y, x + h)\} - \bar{F}(\mu)(x) =$$
$$= \begin{cases} \mu(x + h) - \bar{F}(\mu)(x), & \text{if } \ x \in J^- \\ \mu(x) + A(x, x + h) - \bar{F}(\mu)(x), & \text{if } \ x \in J^+, \end{cases}$$

since in the former case, the minimum of $\mu(y) + A(y, x + h)$ over $[x, x + h]$ is attained at $x + h$, whilst in the latter is attained at x.

Hence, we have that

$$\lim_{h\to 0^+}\frac{\bar{F}(\mu)(x+h)-\bar{F}(\mu)(x)}{h}=$$

$$=\begin{cases}\lim\limits_{h\to 0^+} h^{-1}\left[A(x,x+h)\wedge\left(\mu(x+h)-\bar{F}(\mu)(x)\right)\right], & \text{if } x\in J^-\\ \lim\limits_{h\to 0^+} h^{-1}\left[A(x,x+h)\wedge\left(\mu(x)+A(x,x+h)-\bar{F}(\mu)(x)\right)\right], & \text{if } x\in J^+\end{cases}$$

$$=\begin{cases}\alpha^+(x)\wedge\lim\limits_{h\to 0^+} h^{-1}\left(\mu(x+h)-\bar{F}(\mu)(x)\right), & \text{if } x\in J^-\\ \alpha^+(x)\wedge\lim\limits_{h\to 0^+} h^{-1}\left(\mu(x)+A(x,x+h)-\bar{F}(\mu)(x)\right), & \text{if } x\in J^+\end{cases}$$

$$=\begin{cases}\alpha^+(x)\wedge\mu'(x), & \text{if } x\in J^- \text{ and } \bar{F}(\mu)(x)\geq\mu(x)\\ \alpha^+(x)\wedge+\infty=\alpha^+(x), & \text{if } x\in J^- \text{ and } \bar{F}(\mu)(x)<\mu(x)\\ \alpha^+(x)=\alpha^+(x)\wedge\mu'(x), & \text{if } x\in J^+ \text{ and } \bar{F}(\mu)(x)\geq\mu(x)\\ \alpha^+(x)\wedge+\infty=\alpha^+(x), & \text{if } x\in J^+ \text{ and } \bar{F}(\mu)(x)<\mu(x)\end{cases}$$

$$=\begin{cases}\alpha^+(x)\wedge\mu'(x), & \text{if } \bar{F}(\mu)(x)\geq\mu(x)\\ \alpha^+(x), & \text{if } \bar{F}(\mu)(x)<\mu(x).\end{cases}$$

Note that, by definition of $\bar{F}$, $\bar{F}(\mu)(x)\leq\mu(x)$ must hold. In conclusion, we proved that $\bar{F}(u)\in W^{1,\infty}(I)$ and satisfies (3.10).
Applying the same reasoning it can be proved that $\bar{B}(u)\in W^{1,\infty}(I)$ and satisfies (3.11). □

Proposition 15. *Assume that $\mu^+\in\mathcal{Q}$ and let $w\in P$, then w is feasible for Problem* (3.3) *(i.e., it satisfies constraints* (3.3b) *and* (3.3c)*), if and only if $\mu^-\leq w\leq\mu^+$ and $\bar{M}(w)=w$.*

Proof. ⇒) Assume that w is feasible for Problem 6, then w satisfies a. e. $w'\leq\alpha^+$. Thus, $\phi=w$ is the solution of (3.4) for $\mu=w$, which implies that $\bar{F}(w)=w$. Analogously $\bar{B}(w)=w$, so that $\bar{M}(w)=w$. Moreover, since w satisfies the bounds of Problem 6, it follows that $\mu^-\leq w\leq\mu^+$.

⇐) Condition (3.3b) holds by hypothesis. Since $\bar{M}(w)=w$, then it must be $\bar{F}(w)=w$. In fact, if by contradiction $\bar{F}(w)<w$, then $\bar{M}(w)\leq\bar{F}(w)<$

w, which contradicts the assumption. So $\bar{F}(w) = w$, implies that, a. e., $\bar{F}(w)' = w'$ which by definition of $\bar{F}(w)'$ in (3.4), implies that, a. e., $w' \leq \alpha^+$. Analogously, it must be $\bar{B}(w) = w$, which implies that, a. e. $w' \geq \alpha^-$ and condition (3.3c) holds.

□

Proposition 16. *Assume that $\mu^+ \in \mathcal{Q}$. Then, Problem 6 is feasible if and only if $\bar{M}(\mu^+) \geq \mu^-$.*

Proof. $\Rightarrow$) Let w be a feasible solution of Problem 6. Then, by Proposition 15, $\bar{M}(w) = w \leq \mu^+$. Hence, being $\bar{M}$ order preserving, $\bar{M}(\mu^+) \geq \bar{M}(w) \geq \mu^-$.

$\Leftarrow$) $w = \bar{M}(\mu^+)$ satisfies $\bar{M}(w) = w$ (by part *ii.* of Proposition 11) and $\mu^+ \leq w \leq \mu^-$ (by hypothesis). Hence, by Proposition 15, w is a feasible solution of Problem 6.

□

Proposition 17. *If $\mu^+ \in \mathcal{Q}$ and Problem 6 is feasible, then $w^* = \bar{M}(\mu^+)$ is its optimal solution.*

Proof. By contradiction, assume that there exists a feasible $\tilde{w}$ such that $\Psi(\tilde{w}) < \Psi(w^*)$. Since $\tilde{w}$ is feasible, Proposition 15 implies that $\bar{M}(\tilde{w}) = \tilde{w}$. Moreover, since Ψ is order reversing, $\tilde{w} \not\leq w^*$. This is not possible since $w^* = \bigvee\{w \in P : w \leq \bar{M}(w), w \leq \mu^+\} \geq \tilde{w}$, by Proposition 12 . □

Chapter 4

Optimal time-complexity speed planning for robot manipulators

In this chapter we still deal with a speed-planning problem but now related to a robotic manipulator. The problem is somehow related to the one discussed in Chapter 2 but it turns out to be more challenging due to more difficult constraints. In particular, we present an algorithm for finding the time-optimal speed law along an assigned path that satisfies speed and acceleration constraints and respects the maximum forces and torques allowed by the actuators. The addressed optimization problem is a finite dimensional reformulation of the continuous-time speed optimization problem, obtained by discretizing the speed profile with n points. The proposed algorithm has linear complexity with respect to n and to the number of degrees of freedom. Such complexity is the best possible for this problem. Numerical tests show that the proposed algorithm is significantly faster than algorithms already existing in literature.

4.1 Problem formulation

Let $\mathcal{Q}$ be a smooth manifold of dimension p that represents the configuration space of a robotic manipulator with p-degrees of freedom (p-DOF). Let $\bar{\gamma}$:

$[0,1] \to \mathcal{Q}$ be a smooth curve whose image set $\operatorname{Im}\bar{\gamma} = \Gamma$ represents the assigned path to be followed by the manipulator. We assume that there exist two open sets $U \supset \Gamma$, $V \subset \mathbb{R}^p$ and an invertible and smooth function $\phi : U \to V$. Function ϕ is a local chart that allows representing each configuration $\mathbf{q} \in U$ with coordinate vector $\phi(\mathbf{q}) \in \mathbb{R}^p$.

The coordinate vector $\mathbf{q}$ of a trajectory in U satisfies the dynamic equation

$$\boldsymbol{D}(\mathbf{q})\ddot{\mathbf{q}} + \boldsymbol{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \boldsymbol{\ell}(\mathbf{q}) = \boldsymbol{\tau}, \tag{4.1}$$

where $\mathbf{q} \in \mathbb{R}^p$ is the generalized position vector, $\boldsymbol{\tau} \in \mathbb{R}^p$ is the generalized force vector, $\boldsymbol{D}(\mathbf{q})$ is the mass matrix, $\boldsymbol{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the matrix accounting for centrifugal and Coriolis effects (assumed to be linear in $\dot{\mathbf{q}}$) and $\boldsymbol{\ell}(\mathbf{q})$ is the vector accounting for joints position dependent forces, including gravity. Note that we do not consider Coulomb friction forces.

Let $\boldsymbol{\gamma} \in C^2([0, s_f], \mathbb{R}^p)$ be a function such that $\phi(\boldsymbol{\gamma}[0, s_f]) = \Gamma$ and $(\forall \lambda \in [0, s_f])\ \|\boldsymbol{\gamma}'(\lambda)\| = 1$. The image set $\boldsymbol{\gamma}([0, s_f])$ represents the coordinates of the elements of reference path Γ. In particular, $\boldsymbol{\gamma}(0)$ and $\boldsymbol{\gamma}(s_f)$ are the coordinates of the initial and final configurations. Define t_f as the time when the robot reaches the end of the path. Let $\lambda : [0, t_f] \to [0, s_f]$ be a differentiable monotone increasing function that represents the position of the robot as a function of time and let $v : [0, s_f] \to [0, +\infty]$ be such that $(\forall t \in [0, t_f])\ \dot{\lambda}(t) = v(\lambda(t))$. Namely, $v(s)$ is the speed of the robot at position s. We impose $(\forall s \in [0, s_f])$ $v(s) \geq 0$. For any $t \in [0, t_f]$, using the chain rule, we obtain

$$\begin{aligned} \mathbf{q}(t) &= \boldsymbol{\gamma}(\lambda(t)), \\ \dot{\mathbf{q}}(t) &= \boldsymbol{\gamma}'(\lambda(t))v(\lambda(t)), \\ \ddot{\mathbf{q}}(t) &= \boldsymbol{\gamma}'(\lambda(t))v'(\lambda(t))v(\lambda(t)) + \boldsymbol{\gamma}''(\lambda(t))v(\lambda(t))^2, \end{aligned} \tag{4.2}$$

which represent the relations between the path $\boldsymbol{\gamma}$ and the generalized position q. Substituting (4.2) into the dynamic equations (4.1) and setting $s = \lambda(t)$, we rewrite the dynamic equation (4.1) as follows:

$$\mathbf{d}(s)v'(s)v(s) + \mathbf{c}(s)v(s)^2 + \mathbf{g}(s) = \boldsymbol{\tau}(s), \tag{4.3}$$

where the parameters in (4.3) are defined as

$$\begin{aligned}\mathbf{d}(s) &= \boldsymbol{D}(\boldsymbol{\gamma}(s))\boldsymbol{\gamma}'(s),\\ \mathbf{c}(s) &= \boldsymbol{D}(\boldsymbol{\gamma}(s))\boldsymbol{\gamma}''(s) + \boldsymbol{C}(\boldsymbol{\gamma}(s),\boldsymbol{\gamma}'(s))\boldsymbol{\gamma}'(s),\\ \mathbf{g}(s) &= \boldsymbol{\ell}(\boldsymbol{\gamma}(s)).\end{aligned} \tag{4.4}$$

Now, consider the following change of variables (previously introduced in [41]) where, $(\forall s \in [0, s_f])$ we set

$$a(s) = v'(s)v(s), \quad w(s) = v(s)^2, \tag{4.5}$$

and note that

$$w'(s) = 2a(s). \tag{4.6}$$

By substituting (4.5) and (4.6) in (4.2) and (4.3) it follows that:

$$\boldsymbol{\tau}(s) = \mathbf{d}(s)a(s) + \mathbf{c}(s)w(s) + \mathbf{g}(s), \tag{4.7}$$

$$\ddot{\mathbf{q}}(s) = \boldsymbol{\gamma}'(s)a(s) + \boldsymbol{\gamma}''(s)w(s)^2, \tag{4.8}$$

$$\dot{\mathbf{q}}(s)^2 = [\boldsymbol{\gamma}'(s)]^2 w(s). \tag{4.9}$$

The objective function is given by the overall travel time t_f defined as

$$t_f = \int_0^{t_f} 1\, dt = \int_0^{s_f} v(s)^{-1}\, ds = \int_0^{s_f} w(s)^{-1/2}\, ds.$$

Finally, let $\boldsymbol{\mu}, \boldsymbol{\psi}, \boldsymbol{\alpha} : [0, s_f] \to \mathbb{R}^p_+$ be assigned bounded functions. we consider the following minimum time problem:

Problem 7.

$$\min_{\mathbf{q},\in C^2, a, \boldsymbol{\tau}\in C^0, w\in C^1} \int_0^{s_f} w(s)^{-1/2}\, ds, \tag{4.10}$$

such that $(\forall s \in [0, s_f])$

$$\mathbf{d}(s)a(s) + \mathbf{c}(s)w(s) + \mathbf{g}(s) = \boldsymbol{\tau}(s), \tag{4.11}$$

$$\boldsymbol{\gamma}'(s)a(s) + \boldsymbol{\gamma}''(s)w(s) = \ddot{\mathbf{q}}(s), \tag{4.12}$$

$$w'(s) = 2a(s), \tag{4.13}$$

$$|\boldsymbol{\tau}(s)| \leq \boldsymbol{\mu}(s), \tag{4.14}$$

$$0 \leq \boldsymbol{\gamma}'(s)^2 w(s) \leq \boldsymbol{\psi}(s)^2, \tag{4.15}$$

$$|\ddot{\mathbf{q}}(s)| \leq \boldsymbol{\alpha}(s), \tag{4.16}$$

$$w(0) = 0,\ w(s_f) = 0, \tag{4.17}$$

where (4.14) represents the bounds on generalized forces, (4.15) and (4.16) represent the bounds on joints speed and acceleration. Constraints (4.17) specify the interpolation conditions at the beginning and at the end of the path. The squares of the two vectors $\boldsymbol{\gamma}'(s)$ and $\boldsymbol{\psi}(s)$ in (4.15) are to be intended component-wise.

The following assumption is a basic requirement for fulfilling constraint (4.15).

Assumption 3. *We assume that* $\boldsymbol{\psi}$ *is a positive continuous function, i.e.,* $(\forall s \in [0, s_f])$ $\psi_i(s) > 0$ *with* $i = 1, \ldots, p$.

Next assumption requires that the maximum allowed generalized forces are able to counteract external forces (such as gravity) when the manipulator is fixed at each point of Γ.

Assumption 4. *We assume that* $\exists \varepsilon \in \mathbb{R}$, $\varepsilon > 0$ *such that* $(\forall s \in [0, s_f])$ $\boldsymbol{\mu}(s) - |\mathbf{g}(s)| > \varepsilon \mathbf{1}$.

Indeed for $w = 0$ condition (4.14) reduces to $(\forall s \in [0, s_f])$ $|\mathbf{g}(s)| \leq \boldsymbol{\mu}(s)$.

Remark 6. *Assumptions 3 and 4 will be assumed to hold true throughout the paper.*

The following proposition shows that Problem 7 admits a solution.

Proposition 18. *Problem 7 admits an optimal solution w^*, and moreover,*

$$\int_0^{s_f} w^*(s)^{-1/2}\, ds \leq U < \infty,$$

where U is a constant depending on problem data.

Proof. See Appendix 4.5 □

Our purpose is not to directly solve Problem 7, but find an approximated solution based on a finite dimensional approximation. Namely,we consider the following problem, obtained by uniformly sampling the interval $[0, s_f]$ in n points $s_1,\ s_2, \ldots, s_n$ from $s_1 = 0$ to $s_n = s_f$:

Problem 8.

$$\min_{\ddot{\mathbf{q}},\boldsymbol{\tau},\boldsymbol{a},\mathbf{w}} \; 2h \sum_{i=1}^{n-1} \frac{1}{\sqrt{w_i} + \sqrt{w_{i+1}}}, \tag{4.18}$$

such that $i = 1, \ldots, n-1$,

$$\mathbf{d}_i a_i + \mathbf{c}_i w_i + \mathbf{g}_i = \boldsymbol{\tau}_i, \tag{4.19}$$

$$\boldsymbol{\gamma}_i' a_i + \boldsymbol{\gamma}_i'' w_i = \ddot{\mathbf{q}}_i, \tag{4.20}$$

$$w_{i+1} - w_i = 2a_i h, \tag{4.21}$$

$$|\boldsymbol{\tau}_i| \leq \boldsymbol{\mu}_i, \tag{4.22}$$

$$|\ddot{\mathbf{q}}_i| \leq \boldsymbol{\alpha}_i \tag{4.23}$$

$$0 \leq \left[\boldsymbol{\gamma}_i'\right]^2 w_i \leq \boldsymbol{\psi}_i^2, \tag{4.24}$$

$$w_1 = 0,\ w_n = 0. \tag{4.25}$$

$$\mathbf{w} \in \mathbb{R}^n\ \boldsymbol{a} \in \mathbb{R}^{n-1}, \boldsymbol{\tau}_i \in \mathbb{R}^p, \tag{4.26}$$

where $h = \frac{s_n}{n-1}$, $\boldsymbol{\alpha}_i = \alpha(s_i)$, $\boldsymbol{\psi}_i = \psi(s_i)$, $\boldsymbol{\mu}_i = \mu(s_i)$, $\boldsymbol{\gamma}_i' = \boldsymbol{\gamma}'(s_i)$, $\boldsymbol{\gamma}_i'' = \boldsymbol{\gamma}''(s_i)$, $\mathbf{d}_i = D(\boldsymbol{\gamma}_i)\boldsymbol{\gamma}_i'$, $\mathbf{c}_i = D(\boldsymbol{\gamma}_i)\boldsymbol{\gamma}_i'' + C(\boldsymbol{\gamma}_i, \boldsymbol{\gamma}_i')\boldsymbol{\gamma}_i'$, and $\mathbf{g}_i = g(\boldsymbol{\gamma}_i)$, with $i = 1, \ldots, n$. Thank to constraints (4.19)-(4.21), it is possible to eliminate variables $\boldsymbol{\tau}_i$ and a_i and use only w_i, with $i = 1, \ldots, n$, as decision variables. The feasible set of Problem 8 is a non-empty set since $\mathbf{w} = 0$ is a feasible

solution (in fact, it also has a nonempty interior). Since Problem 8 is convex, we can easily find a solution with an interior point method (this fact is already known, see for instance [41]).

4.2 Solution algorithms and complexity issues for a class of optimization problems

In this section we present an optimal time complexity algorithm that solves a specific class of optimization problems.

4.2.1 Exact algorithm for the solution of special structured problems

The problems under consideration have the form

$$\begin{aligned} \min \quad & g(v_1, \dots, v_n) \\ & v_i \leq f_j^i(v_{i+1}) \quad i = 1, \dots, n-1, \quad j = 1, \dots, r_i \\ & v_{i+1} \leq b_k^i(v_i) \quad i = 1, \dots, n-1 \quad k = 1, \dots, t_i, \\ & 0 \leq v_i \leq u_i \qquad i = 1, \dots, n, \end{aligned} \tag{4.27}$$

where we make the following assumptions.

Assumption 5. *We assume:*

- *g monotonic non increasing;*
- *f_j^i, concave, increasing and $f_j^i(0) > 0$, $i = 1, \dots, n-1$, $j = 1, \dots, r_i$;*
- *b_k^i, concave, increasing and $b_k^i(0) > 0$, $i = 1, \dots, n-1$, $k = 1, \dots, t_i$.*

The constraints in (4.27) can be rewritten in compact form as follows:

$$\begin{aligned} & v_i \leq \min\{B_i(v_{i+1}), u_i\} \qquad i = 1, \dots, n-1, \\ & v_{i+1} \leq \min\{F_i(v_i), u_{i+1}\} \quad i = 1, \dots, n-1, \end{aligned}$$

where:

$$B_i(v_{i+1}) = \min_{j=1,\dots,r_i} f_j^i(v_{i+1}),$$
$$F_i(v_i) = \min_{k=1,\dots,t_i} b_k^i(v_i).$$

Note that F_i and B_i are both concave and increasing over $\mathbb{R}_+$, since they are the minimum of a finite number of functions with the same properties. We prove that the same holds for $F_i \circ B_i$.

Proposition 19. *$F_i \circ B_i$ is increasing and concave over $\mathbb{R}_+$.*

Proof. The fact that $F_i \circ B_i$ is increasing follows immediately from the increasingness of F_i and B_i. For what concerns concavity, $(\forall x, y \geq 0),\ \lambda \in [0,1]$:

$$F_i \circ B_i(\lambda x + (1-\lambda)y) = F_i(B_i(\lambda x + (1-\lambda)y)) \geq$$
$$\geq F_i(\lambda B_i(x) + (1-\lambda)B_i(y)) \geq \lambda F_i \circ B_i(x) + (1-\lambda)F_i \circ B_i(y),$$

where the first inequality is a consequence of the concavity of B_i and the fact that F_i is increasing, while the second inequality comes from concavity of F_i. □

It immediately follows that:

$$F_i \circ B_i(x) - x \ \text{ concave}, \quad F_i \circ B_i(0) > 0. \tag{4.28}$$

Then, there exists at most one point $\bar{v}_{i+1} > 0$ such that $F_i \circ B_i(\bar{v}_{i+1}) - \bar{v}_{i+1} = 0$. Similarly, there exists at most one point $\bar{v}_i > 0$ such that $B_i \circ F_i(\bar{v}_i) - \bar{v}_i = 0$. Note that $\bar{v}_i, \bar{v}_{i+1}$ are the positive fixed points of $B_i \circ F_i$ and $F_i \circ B_i$, respectively. Alternatively, $(\bar{v}_i, \bar{v}_{i+1})$ is also the optimal solution of the following two-dimensional convex problem:

$$\begin{aligned} \max \quad & v_i + v_{i+1} \\ & v_i \leq f_j^i(v_{i+1}) \quad j = 1, \dots, r_i \\ & v_{i+1} \leq b_k^i(v_i) \quad k = 1, \dots, t_i. \end{aligned}$$

The following result holds.

Proposition 20. *Under Assumption 5, the optimal solution of (4.27) is the component-wise maximum of its feasible region, i.e., if we denote by X the feasible region, it is the point $\mathbf{v}^* \in X$ such that $\mathbf{v} \leq \mathbf{v}^*$ for all $\mathbf{v} \in X$.*

Proof. See Chapter 2 and reference [45]. □

We consider Algorithm 3 for the solution of problem (4.27). The algorithm

Algorithm 3: Forward-Backward algorithm for the solution of the problem.

Data: $\mathbf{u} \in \mathbb{R}^n_+$;

Set $\bar{\mathbf{u}} = \mathbf{u}$;

/* Forward phase */

foreach $i \in \{1, \ldots, n-1\}$ **do**

  Compute the nonnegative fixed points $\bar{v}_i$ and $\bar{v}_{i+1}$ for $B_i \circ F_i$ and $F_i \circ B_i$, respectively (if they do not exist, set $\bar{v}_i = +\infty$, $\bar{v}_{i+1} = +\infty$);

  Set $\bar{u}_i = \min\{\bar{u}_i, B_i(\bar{u}_{i+1}), \bar{v}_i\}$;

  Set $\bar{u}_{i+1} = \min\{\bar{u}_{i+1}, F_i(\bar{u}_i), \bar{v}_{i+1}\}$;

/* Backward phase */

foreach $i \in \{n-1, \ldots, 1\}$ **do**

  Set $\bar{u}_i = \min\{B_i(\bar{u}_{i+1}), \bar{u}_i\}$;

is correct, as stated in the following proposition.

Proposition 21. *Algorithm 3 returns the optimal solution $\mathbf{v}^*$ of problem (4.27).*

Proof. We first remark that at each iteration $\bar{\mathbf{u}} \geq \mathbf{v}^*$ holds. If a fixed point $\bar{v}_{i+1}$ for $F_i \circ B_i$ exists, then after the backward propagation, $\bar{u}_{i+1} \leq \bar{v}_{i+1}$, and $\bar{u}_i = \min\{\bar{u}_i^{old}, B_i(\bar{u}_{i+1})\}$, where $\bar{u}_i^{old}$ denotes the upper bound for v_i after the forward phase. We show that

$$F_i(\bar{u}_i) = \min\{F_i(\bar{u}_i^{old}), F_i \circ B_i(\bar{u}_{i+1})\} \geq \bar{u}_{i+1}. \tag{4.29}$$

If this is true for all i, then the point $\bar{\mathbf{u}}$ at the end of the backward phase is a feasible solution of (4.27). Indeed, by definition of $\bar{u}_i$ in the backward phase, $\forall i$

$$\bar{u}_i \leq B_i(\bar{u}_{i+1}),$$

while by (4.29), $\forall i$

$$\bar{u}_{i+1} \leq F_i(\bar{u}_i),$$

so that $\bar{\mathbf{u}}$ is feasible for (4.27). Since $\bar{\mathbf{u}} \geq \mathbf{v}^*$ holds and g is monotone non increasing, we have that $\bar{\mathbf{u}}$ is the optimal solution of (4.27). We only need to prove that (4.29) is true. Note that $\bar{u}_i^{old}$ is the result of the first forward propagation, so that $F_i(\bar{u}_i^{old}) \geq \bar{u}_{i+1}^{old} \geq \bar{u}_{i+1}$. Thus, we need to prove that $F_i \circ B_i(\bar{u}_{i+1}) \geq \bar{u}_{i+1}$ with $\bar{u}_{i+1} \leq \bar{v}_{i+1}$. In view of (4.28), if the fixed point $\bar{v}_{i+1}$ for $F_i \circ B_i$ exists, it is the unique nonnegative root of $F_i \circ B_i(x) - x$ and

$$F_i \circ B_i(x) - x > 0, \ \forall x \leq \bar{v}_{i+1},$$

from which the result is proved. Otherwise, if no fixed point exists,

$$F_i \circ B_i(x) - x > 0, \ \forall x \in \mathbb{R}_+,$$

form which the result is still proved. □

Under a given condition, Algorithm 3 can be further simplified.

Remark 7. *If F_i and B_i, $i = 1, \ldots, n$, fulfill the so called* superiority condition*, i.e. $F_i(x), B_i(x) > x, \ \forall x \in \mathbb{R}_+$, then $\bar{v}_i, \bar{v}_{i+1} = +\infty$ and the forward phase can be reduced to*

$$\bar{u}_{i+1} = \min\{\bar{u}_{i+1}, F_i(\bar{u}_i)\}.$$

4.2.2 Solving the subproblems in the forward phase and complexity results

We first remark that

$$\bar{u}_i = \min\{\bar{u}_i, B_i(\bar{u}_{i+1}), \bar{v}_i\},$$
$$\bar{u}_{i+1} = \min\{\bar{u}_{i+1}, F_i(\bar{u}_i), \bar{v}_{i+1}\},$$

defined in the forward phase of Algorithm 3 are the solution of the 2-D convex optimization problem

$$\begin{aligned} \max \quad & v_i + v_{i+1} \\ & v_i \le f_j^i(v_{i+1}) \quad j = 1, \dots, r_i \\ & v_{i+1} \le b_k^i(v_i) \quad k = 1, \dots, t_i \\ & 0 \le v_i \le f_{r_i+1}^i(v_{i+1}) \equiv \bar{u}_i \\ & 0 \le v_{i+1} \le b_{t_i+1}^i(v_i) \equiv \bar{u}_{i+1}. \end{aligned} \tag{4.30}$$

Alternatively, $\bar{u}_i, \bar{u}_{i+1}$ can also be detected as fixed points of $B_i^{\bar{\mathbf{u}}} \circ F_i^{\bar{\mathbf{u}}}$ and $F_i^{\bar{\mathbf{u}}} \circ B_i^{\bar{\mathbf{u}}}$, respectively, where

$$\begin{aligned} F_i^{\bar{\mathbf{u}}}(x) &= \min\{\bar{u}_i, F_i(x)\}, \\ B_i^{\bar{\mathbf{u}}}(x) &= \min\{\bar{u}_{i+1}, B_i(x)\}. \end{aligned}$$

Although any convex optimization or any fixed point solver could be exploited for detecting these values, we propose the simple Algorithm 4, which turns out to be quite effective in practice. We denote by

$$[F_i]^{-1}(x) = \max_{j=1,\dots,r_i} \{[f_j^i]^{-1}(x)\}.$$

Note that f_j^i increasing and concave implies that $[f_j^i]^{-1}$ is increasing and convex and, consequently, $[F_i]^{-1}$ is increasing and convex. We illustrate how the algorithm works through an example

Example 1. *Let*

$$\begin{aligned} b_1^i(x) &= \tfrac{3}{2}x + 2, \quad b_2^i(x) = x + 3, \\ b_3^i(x) &= \tfrac{1}{2}x + 5, \quad b_4^i(x) = 8. \end{aligned}$$

and

$$\begin{aligned} [f_1^i]^{-1}(x) &= x - 1, \qquad [f_2^i]^{-1}(x) = \tfrac{9}{2}x - 8, \\ [f_3^i]^{-1}(x) &= 5x - 10. \end{aligned}$$

Moreover, let $u_i = u_{i+1} = 8$. Then, we initially set $\bar{x}_1 = 8$. In the first iteration we have

$$\bar{y} = B_i^{\bar{\mathbf{u}}}(\bar{x}_1) = \min_{k=1,\dots,4} \{b_k^i(\bar{x}_1)\} = \min\{14, 11, 9, 8\} = 8,$$

Algorithm 4: Algorithm for the computation of the new values $\bar{u}_i$ and $\bar{u}_{i+1}$.

1 Let $\bar{x}_1 = \bar{u}_i$;
2 Set $\bar{y} = -\infty$, $\bar{z} = +\infty$, $h = 1$;
3 **while** $\bar{y} < \bar{z}$ **do**
4 Set $\bar{y} = B_i^{\bar{\mathbf{u}}}(\bar{x}_h)$ and $\bar{k} \in \{1, \dots, t_i + 1\}$: $b_{\bar{k}}^i(\bar{x}_h) = \bar{y}$;
5 Set $\bar{z} = [F_i]^{-1}(\bar{x}_h)$ and $\bar{j} \in \{1, \dots, r_i\}$: $[f_{\bar{j}}^i]^{-1}(\bar{x}_h) = \bar{z}$;
6 Let $\bar{x}_{h+1}$ be the solution of the one-dimensional equation $f_{\bar{j}}^i(b_{\bar{k}}^i(x)) = x$;
7 Set $h = h + 1$;
8 **return** $(\bar{u}_i, \bar{u}_{i+1}) = (\bar{x}_h, \bar{y})$;

with $\bar{k} = 4$, *and*

$$\bar{z} = [F_i]^{-1}(\bar{x}_1) = \max_{j=1,\dots,3}\{[f_j^i]^{-1}(\bar{x}_1)\} = \max\{7, 28, 30\} = 30,$$

with $\bar{j} = 3$. *Then,* $\bar{x}_2$ *is the solution of the equation* $8 = 5x - 10$, *i.e.,* $\bar{x}_2 = \frac{18}{5}$ *(See also Figure 4.1). In the second iteration we have*

$$\bar{y} = B_i^{\bar{\mathbf{u}}}(\bar{x}_2) = \min_{k=1,\dots,4}\{b_k^i(\bar{x}_2)\} = \min\left\{\frac{37}{5}, \frac{33}{5}, \frac{34}{5}, 8\right\} = \frac{33}{5},$$

with $\bar{k} = 2$, *and*

$$\bar{z} = [F_i]^{-1}(\bar{x}_2) = \max_{j=1,\dots,3}\{[f_j^i]^{-1}(\bar{x}_2)\} = \max\left\{\frac{13}{5}, \frac{41}{5}, 8\right\} = \frac{41}{5},$$

with $\bar{j} = 2$. *Then,* $\bar{x}_3$ *is the solution of the equation* $x+3 = \frac{9}{2}x-8$, *i.e.,* $\bar{x}_3 = \frac{22}{7}$ *(See also Figure 4.2). In the third iteration we have*

$$\bar{y} = B_i^{\bar{\mathbf{u}}}(\bar{x}_3) = \min_{k=1,\dots,4}\{b_k^i(\bar{x}_3)\} = \min\left\{\frac{47}{7}, \frac{43}{7}, \frac{46}{7}, 8\right\} = \frac{43}{7},$$

with $\bar{k} = 2$, *and*

$$\bar{z} = [F_i]^{-1}(\bar{x}_3) = \max_{j=1,\dots,3}\{[f_j^i]^{-1}(\bar{x}_3)\} \qquad = \max\left\{\frac{15}{7}, \frac{43}{7}, \frac{40}{7}\right\} = \frac{43}{7},$$

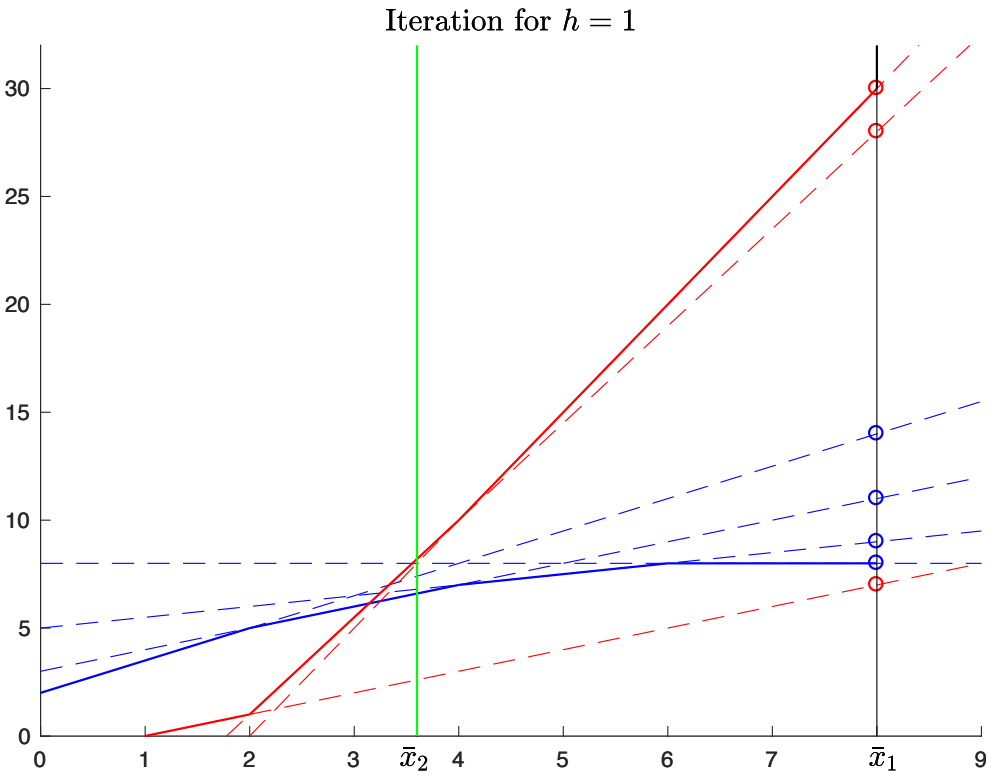


Figure 4.1: The first step of Algorithm 4. The red lines represent the linear function $[f_{\bar{j}}^i]^{-1}$ while the blue ones are the functions $b_{\bar{k}^i}$. The green line represents the solution of the one dimensional equation $f_{\bar{j}}^i(b_{\bar{k}}^i(x)) = x$.

with $\bar{j} = 2$. *Then,* $\bar{x}_4 = \bar{x}_3$ *and since* $\bar{y} = \bar{z}$ *the algorithm stops and returns the optimal solution* $\left(\frac{22}{7}, \frac{43}{7}\right)$ *(See also Figure 4.3).*

If we denote by C_b the time needed to evaluate one function b_k^i, by C_f the time needed to evaluate one function $[f_j^i]^{-1}$, and by C_{eq} the time needed to solve a one-dimensional equation $f_j^i(b_k^i(x)) = x$, we can state the complexity of Algorithm 4. Before we need to prove one lemma.

Lemma 1. *The sequence* $\{\bar{x}_h\}$ *is strictly decreasing.*

Proof. If the algorithm does not stop, then $\bar{y} < \bar{z}$, or, equivalently $b_{\bar{k}}^i(\bar{x}_h) -$

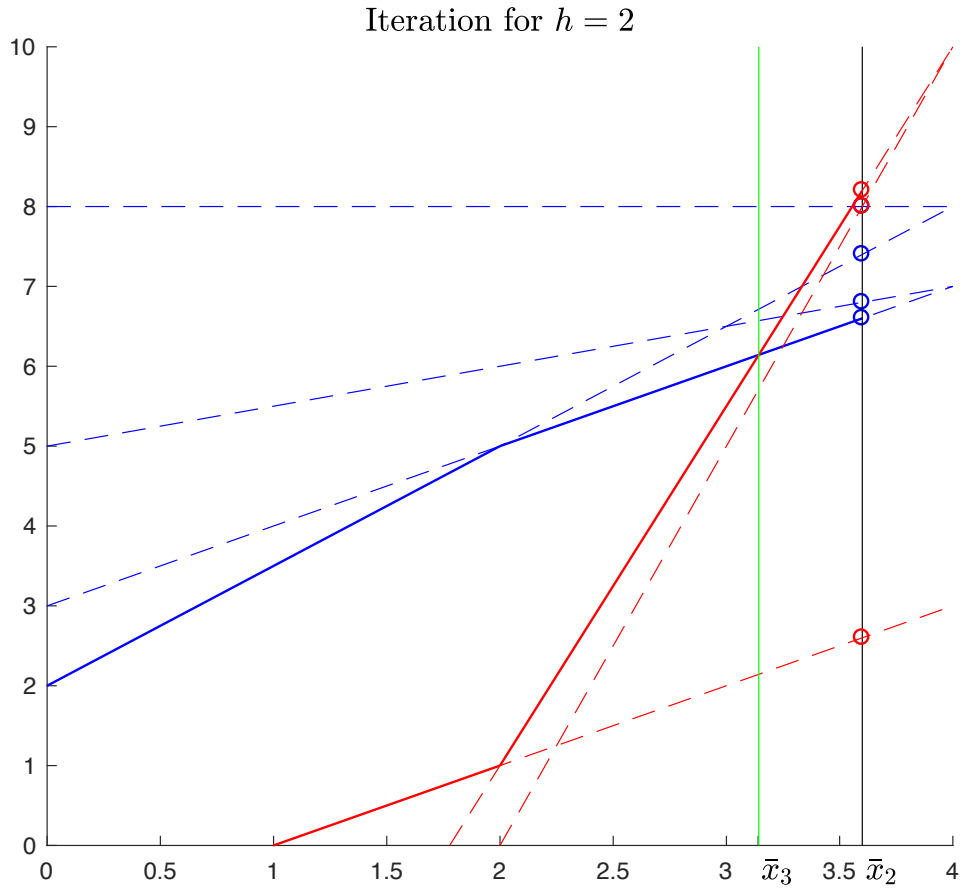


Figure 4.2: The second step of Algorithm 4.

$[f_{\bar{j}}^i]^{-1}(\bar{x}_h) < 0$. Since $[f_{\bar{j}}^i]^{-1}$ is convex, $b_k^i(x) - [f_{\bar{j}}^i]^{-1}(x)$ is concave. Moreover, $b_k^i(0) - [f_{\bar{j}}^i]^{-1}(0) > 0$. Then, there exists a unique value $x \in (0, \bar{x}_h)$ such that $b_k^i(x) - [f_{\bar{j}}^i]^{-1}(x) = 0$. Such value, lower than $\bar{x}_h$, is also the solution $\bar{x}_{h+1}$ of the one-dimensional equation $f_{\bar{j}}^i(b_k^i(x)) = x$. □

Now we are ready to prove the complexity result.

Proposition 22. *Algorithm 4 has complexity* $O(r_i t_i (t_i C_b + r_i C_f + C_{eq}))$.

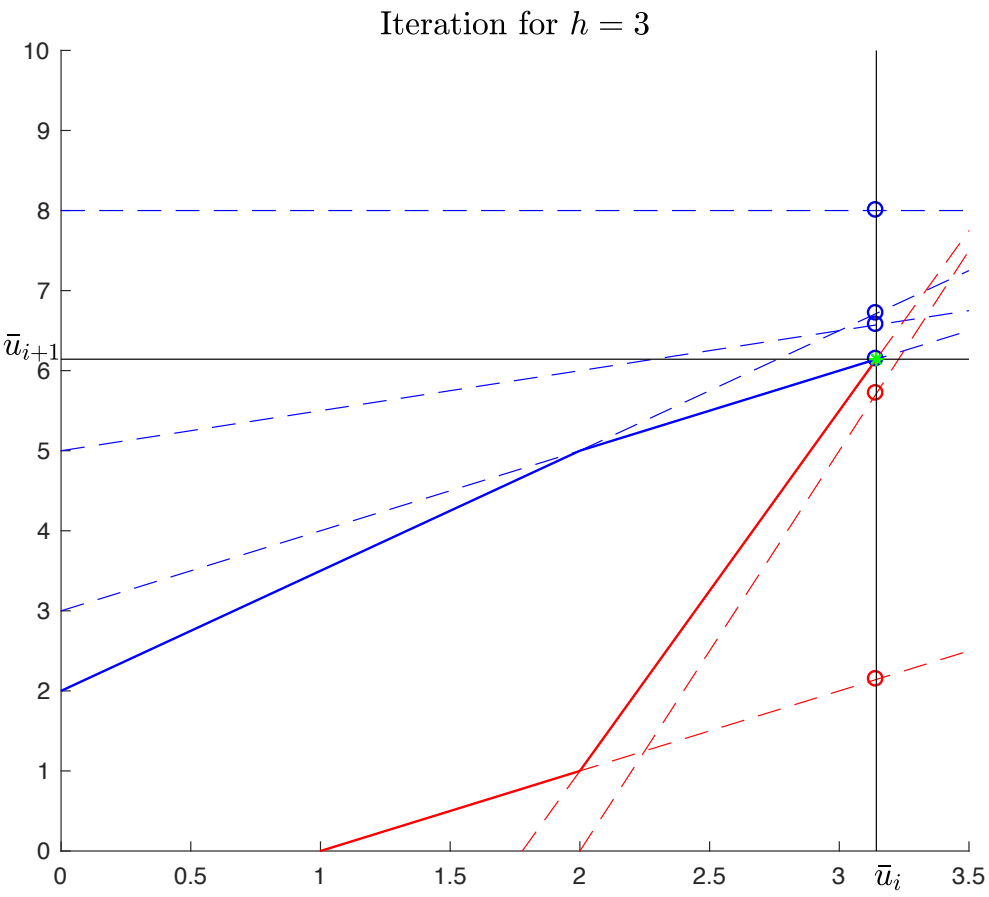


Figure 4.3: The last step of Algorithm 4. The green marker represents value $(u_i, u_{i+1}) = (\bar{x}_h, \bar{y})$ returned.

Proof. In view of Lemma 1 the sequence $\{\bar{x}_h\}$ is decreasing, which means that an equation $f_j^i(b_k^i(x)) = x$, $j = 1, \ldots, r_i$, $k = 1, \ldots, t_i + 1$, is solved at most once. Thus, the number of iterations is at most $(t_i + 1)r_i$. At each iteration we need to evaluate $B_i^{\bar{\mathbf{u}}}$ ($(t_i+1)C_b$ operations), evaluate $[F_i]^{-1}$ ($r_i C_f$ operations), and solve a one-dimensional equation (C_{eq} operations). □

Algorithm 4 and the related complexity result can be improved when the

functions b_k^i and f_j^i are linear ones. Algorithm 5 is a variant of Algorithm 4 for the linear case. In the initialization phase of Algorithm 5 the slopes m_k, $k = 1, \ldots, t_i + 1$, of the linear functions b_k^i are ordered in a decreasing way, i.e.,

$$m_k > m_{k+1}, \quad k = 1, \ldots, t_i,$$

while the slopes η_j, $j = 1, \ldots, r_i$, of the linear functions $[f_j^i]^{-1}$ are ordered in a decreasing way, i.e.,

$$\eta_j < \eta_{j+1}, \quad j = 1, \ldots, r_i - 1.$$

Note that, in case of two linear functions with the same slope, one of the two can be eliminated since it gives rise to a redundant constraint. The pointer ξ is updated in such a way that at each iteration it identifies the index $\bar{k}$ such that $\bar{y} = B_i^{\bar{\mathbf{u}}}(\bar{x}_h) = b_{\bar{k}}^i(\bar{x}_h)$, without the need of computing the value of *all* the functions b_k^i (as, instead, required in Algorithm 4) and, thus, saving the $O(t_i)$ time required by this computation. Similarly, the pointer ϕ is updated in such a way that at each iteration it identifies the index $\bar{j}$ such that $\bar{z} = [F_i]^{-1}(\bar{x}_h) = [f_{\bar{j}}^i]^{-1}(\bar{x}_h)$. We illustrate the algorithm on the previous example.

Example 2. *The slopes of the functions b_k^i are already ordered in a decreasing way, while those of the functions $[f_j^i]^{-1}$ are already ordered in an increasing way.*
In the first iteration we immediately exit the first inner `While` *cycle since $b_4^i(\bar{x}_1) < b_3^i(\bar{x}_1)$, so that at the end of the cycle we set $\bar{k} = \xi = 4$. We also immediately exit the second inner* `While` *cycle since $[f_3^i]^{-1}(\bar{x}_1) > [f_2^i]^{-1}(\bar{x}_1)$, so that at the end of the cycle we set $\bar{j} = \phi = 3$.*
In the second iteration the first inner `While` *cycle is repeated twice since*

$$b_4^i(\bar{x}_2) > b_3^i(\bar{x}_2) > b_2^i(\bar{x}_2) < b_1^i(\bar{x}_2),$$

so that at the end of the cycle we set $\bar{k} = \xi = 2$. The second inner `While` *cycle is repeated once since*

$$[f_3^i]^{-1}(\bar{x}_1) < [f_2^i]^{-1}(\bar{x}_2) > [f_1^i]^{-1}(\bar{x}_2),$$

Algorithm 5: Algorithm for the computation of the new values $\bar{u}_i$ and $\bar{u}_{i+1}$ in the linear case.

Order the slopes of the linear functions b^i_k, $k = 1, \ldots, t_i + 1$, in a decreasing way;
Order the slopes of the linear functions $[f^i_j]^{-1}$, $j = 1, \ldots, r_i$, in an increasing way;
Remove redundant constraints and update t_i and r_i accordingly;
Set $\xi = t_i + 1$ and $\phi = r_i$;
Let $\bar{x}_1 = \bar{u}_i$;
Set $\bar{y} = -\infty$, $\bar{z} = +\infty$, $h = 1$;
while $\bar{y} < \bar{z}$ **do**
    while $\xi > 1$ *and* $b^i_{\xi-1}(\bar{x}_h) < b^i_\xi(\bar{x}_h)$ **do**
        Set $\xi = \xi - 1$;
    Set $\bar{k} = \xi$ and $\bar{y} = b^i_{\bar{k}}(\bar{x}_h)$;
    while $\phi > 1$ *and* $[f^i_{\phi-1}]^{-1}(\bar{x}_h) > [f^i_\phi]^{-1}(\bar{x}_h)$ **do**
        Set $\phi = \phi - 1$;
    Set $\bar{j} = \phi$ and $\bar{z} = [f^i_{\bar{j}}]^{-1}(\bar{x}_h)$;
    Let $\bar{x}_{h+1}$ be the solution of the one-dimensional equation $f^i_{\bar{j}}(b^i_{\bar{k}}(x)) = x$;
    Set $h = h + 1$;
return $(\bar{u}_i, \bar{u}_{i+1}) = (\bar{x}_h, \bar{y})$;

so that at the end of the cycle we set $\bar{j} = \phi = 2$.
In the third iteration we immediately exit the first inner `While` *cycle since* $b_2^i(\bar{x}_1) < b_1^i(\bar{x}_3)$*, so that at the end of the cycle we set* $\bar{k} = \xi = 2$*. We also immediately exit the second inner* `While` *cycle since* $[f_2^i]^{-1}(\bar{x}_3) > [f_1^i]^{-1}(\bar{x}_3)$*, so that at the end of the cycle we set* $\bar{j} = \phi = 2$.

The following proposition establishes the complexity of Algorithm 5.

Proposition 23. *Algorithm 5 has complexity* $O(r_i \log(r_i) + t_i \log(t_i))$.

Proof. We first remark that the initial orderings of the slopes already require the computing time $O(r_i \log(r_i) + t_i \log(t_i))$, while the removal of the redundant constraints require time $O(r_i + t_i)$. Next, we remark that in the linear case C_b, C_f and C_{eq} are $O(1)$ operations. In particular, the one-dimensional equation is a linear one. Moreover, we notice that $B_i^{\bar{\mathbf{u}}}$ is a concave piecewise linear function, while $[F_i]^{-1}$ is a convex piecewise linear function. Since in view of Lemma 1 the sequence $\{\bar{x}_h\}$ is decreasing, the corresponding sequence of slopes of the function $B_i^{\bar{\mathbf{u}}}$ at points $\bar{x}_h$ is not decreasing, while the sequence of slopes of the function $[F_i]^{-1}$ is not increasing, and at each iteration at least one slope must change (otherwise we would solve the same linear equation and $\bar{x}_h$ would not change). Then, the number of different slope values and, thus, the number of iterations, can not be larger than $t_i + r_i + 1$. Moreover, by updating the two pointers ξ and ϕ , the overall number of evaluations of the functions b_k^i and $[f_j^i]^{-1}$, needed to compute the different values $\bar{y}$ and $\bar{z}$ in the outer `While` cycle can not be larger than $O(t_i + r_i)$. Consequently, the computing time of the outer `While` cycle is $O(t_i + r_i)$ and the complexity of the algorithm is determined by the initial orderings of the slopes. □

While in practice we employed Algorithm 5 in order to compute $\bar{u}_i, \bar{u}_{i+1}$, in the linear case we could also solve the linear subproblem (4.30). This can be done in linear time $O(t_i + r_i)$ with respect to the number of constraints, e.g., by Megiddo's algorithm (see [52]). Thus, we can state the following complexity result for Problem (4.27) in the linear case.

Theorem 4. *If f_j^i, b_k^i, $i = 1, \ldots, n$, $j = 1, \ldots, r_i$, and $k = 1, \ldots, t_i$, are linear functions, then Problem (4.27) can be solved in time $O(\sum_{i=1}^{n}(t_i + r_i))$ by Algorithm 3, if $\bar{u}_i, \bar{u}_{i+1}$ are computed by Megiddo's algorithm.*

Proof. The complexity result immediately follows by the observation that the most time consuming part of Algorithm 3 is the forward one with the computation of $\bar{u}_i, \bar{u}_{i+1}$. Indeed, the backward part is run in $O(\sum_{i=1}^{n} t_i)$ time (at each iteration we only need to evaluate B_i). □

4.3 Solution of the speed-planning problem

Problem 8 does not belong to the class defined in (4.27). In this section, we show that a small variation of Problem 7, followed by discretization, allows obtaining a problem that belongs to class (4.27). To this end, consider the following family of problems, depending on the positive real parameter h.

Problem 9.

$$\min_{\mathbf{q}\in C^2, a, \boldsymbol{\tau}\in C^0, w\in C^1} \int_0^{s_f} w(s)^{-1/2}\, ds,$$

such that $(\forall s \in [0, s_f],\ j = 1, \ldots, p)$

$$w(s + \lambda_j(s)) = \hat{w}_j(s) \tag{4.31}$$

$$w(s + \eta_j(s)) = \bar{w}_j(s) \tag{4.32}$$

$$d_j(s)a(s) + c_j(s)\hat{w}_j(s) + g(s) = \tau_j(s), \tag{4.33}$$

$$\gamma_j'(s)a(s) + \gamma_j''(s)\bar{w}_j(s) = \ddot{q}_j(s), \tag{4.34}$$

$$[\boldsymbol{\gamma}'(s)]^2 w(s) = \dot{\mathbf{q}}(s)^2, \tag{4.35}$$

$$w'(s) = 2a(s), \tag{4.36}$$

$$0 \le \dot{\mathbf{q}}(s)^2 \le \boldsymbol{\psi}(s)^2, \tag{4.37}$$

$$|\ddot{\mathbf{q}}(s)| \le \boldsymbol{\alpha}(s), \tag{4.38}$$

$$|\tau(s)| \le \boldsymbol{\mu}(s), \tag{4.39}$$

$$w(0) = 0,\ w(s_f) = 0, \tag{4.40}$$

Note that if $\lambda_j(s) = \eta_j(s) = 0$, then $w(s) = \hat{w}_j(s) = \bar{w}_j(s)$ ($\forall s \in [0, s_f]$ and $j = 1, \ldots, p$) and Problem 9 becomes Problem 7. For our purposes, we define functions λ_j and η_j in (4.31)-(4.32) ($\forall s \in [0, s_f]$ and $j = 1, \ldots, p$) as follows:

$$\lambda_j(s) = \begin{cases} h, \ d_j(s)c_j(s) \geq 0 \\ 0, \ d_j(s)c_j(s) < 0, \end{cases} \qquad \eta_j(s) = \begin{cases} h, \ \gamma_j'(s)\gamma_j''(s) \geq 0 \\ 0, \ \gamma_j'(s)\gamma_j''(s) < 0\,. \end{cases} \tag{4.41}$$

Note that, for $h = 0$, Problem 9 becomes Problem 7. Further, for every $h > 0$, Problem 9 has an optimal solution (this can be proved with the same arguments used for Proposition 18). Let w_h^* be the solution of Problem 9 as a function of h. Note that, by (4.34) and (4.38), $\forall h > 0, \forall s \in [0, s_f], |w'(s)| \leq L = 2\sqrt{p}(\|\boldsymbol{\alpha}\| + p\|\boldsymbol{\gamma}''\|\|\boldsymbol{\psi}\|^2)$ so that w_h^* is Lipshitz with constant L, independent of h. Thus, Ascoli-Arzelà Theorem implies that from any succession of solutions $w_{h_i}^*$, with $\lim_{i\to\infty} h_i = 0$ we can extract a convergent subsequence that converges to a solution of Problem 7.

Discretizing Problem 9 with step h, we obtain the following problem.

Problem 10.

$$\min_{\boldsymbol{a},\mathbf{w},\boldsymbol{\tau}} \ 2h \sum_{i=1}^{n-1} \frac{1}{\sqrt{w_i} + \sqrt{w_{i+1}}}, \tag{4.42}$$

such that $(i = 1, \dots, n-1)$

$$\boldsymbol{\lambda}_i w_{i+1} + (\mathbf{1} - \boldsymbol{\lambda}_i) w_i = \hat{\mathbf{w}}_i, \tag{4.43}$$

$$\boldsymbol{\eta}_i w_{i+1} + (\mathbf{1} - \boldsymbol{\eta}_i) b_i = \bar{\mathbf{w}}_i, \tag{4.44}$$

$$\mathbf{d}_i a_i \mathbf{1} + \mathbf{c}_i \hat{\mathbf{w}}_i + \mathbf{g}_i = \boldsymbol{\tau}_i, \tag{4.45}$$

$$\boldsymbol{\gamma}'_i a_i \mathbf{1} + \boldsymbol{\gamma}''_i \bar{\mathbf{w}}_i = \ddot{\mathbf{q}}_i \tag{4.46}$$

$$w_{i+1} - w_i = 2 a_i h, \tag{4.47}$$

$$\boldsymbol{\gamma}'^2_i w_i = \dot{\mathbf{q}}_i \tag{4.48}$$

$$0 \leq \dot{\mathbf{q}}_i \leq \boldsymbol{\psi}_i^2, \tag{4.49}$$

$$|\ddot{\mathbf{q}}_i| \leq \boldsymbol{\alpha}_i, \tag{4.50}$$

$$|\boldsymbol{\tau}_i| \leq \boldsymbol{\mu}_i, \tag{4.51}$$

$$w_1 = 0,\ w_n = 0. \tag{4.52}$$

$$\mathbf{w} \in \mathbb{R}^n\ \boldsymbol{a} \in \mathbb{R}^{n-1}, \boldsymbol{\tau}_i \in \mathbb{R}^p\,. \tag{4.53}$$

Here, for $j = 1, \dots, p$, $i = 1, \dots, n$,

$$\lambda_{j,i} = \begin{cases} 1, & d_{j,i} c_{j,i} \geq 0 \\ 0, & d_{j,i} c_{j,i} < 0\,, \end{cases} \quad \eta_{j,i} = \begin{cases} 1, & \boldsymbol{\gamma}'_{j,i} \boldsymbol{\gamma}''_{j,i} \geq 0 \\ 0, & \boldsymbol{\gamma}'_{j,i} \boldsymbol{\gamma}''_{j,i} < 0\,. \end{cases}$$

where $h = \frac{s_n}{n-1}$, $\boldsymbol{\alpha}_i = \alpha(s_i)$, $\boldsymbol{\psi}_i = \psi(s_i)$, $\boldsymbol{\mu}_i = \mu(s_i)$, $\boldsymbol{\gamma}'_i = \boldsymbol{\gamma}'(s_i)$, $\boldsymbol{\gamma}''_i = \boldsymbol{\gamma}''(s_i)$, $\mathbf{d}_i = D(\boldsymbol{\gamma}_i)\boldsymbol{\gamma}'_i$, $\mathbf{c}_i = D(\boldsymbol{\gamma}_i)\boldsymbol{\gamma}''_i + C(\boldsymbol{\gamma}_i, \boldsymbol{\gamma}'_i)\boldsymbol{\gamma}'_i$, and $\mathbf{g}_i = g(\boldsymbol{\gamma}_i)$, with $i = 1, \dots, n$.

Thanks to constraints (4.43)-(4.48), it is possible to eliminate variables $\hat{\mathbf{w}}_i$, $\bar{\mathbf{w}}_i$, $\boldsymbol{\tau}_i$, and a_i and use only w_i, with $i = 1, \dots, n$, as decision variables. Since Problem 10 is convex, we can easily find a solution with an interior point method (see [41]). Moreover, reference [45] shows that, since the feasible region of the problem contains its component-wise maximum (see Proposition 20), and the objective function (4.42) is strictly monotone increasing with respect to w_i, with $i = 0, \dots, n$, the latter can be substituted with $\max \sum_{i=0}^{n} w_i$ (i.e., maximize the sum of the velocities along the path) obtaining an LP problem. After solving Problem 10, it is possible to find an approximated solution of Problem 9 by quadratic interpolation. A suitable interpolation technique is proposed in [53].

4.3.1 Application of Algorithm 3 to the solution of Problem 10

In this section we show how to apply the results in Section 4.2 to Problem 10. To this end we introduce the following propositions.

Proposition 24. *Problem 10 belongs to the subclass of problems (4.27) with linear constraints.*

Proof. Since the objective function (4.42) is monotonic non increasing and the variables w_i, $i = 0, \dots, n$, are non negative and bounded by (4.58), we only need to prove that constraints (4.50) and (4.51) satisfy Assumption 5 for suitable choices of $\lambda_{j,i}$ and $\eta_{i,j}$, $j = 1, \dots, p$ and $i = 1, \dots, n$.

For the sake of simplicity consider the j-th component of the i-th sample of (4.51). Substituting variable a_i and τ_i with (4.47) and (4.45) in constraints (4.51) we have:

$$|(d_{j,i} + 2hc_{j,i}\lambda_{j,i})w_{i+1} + (-d_{j,i} + (1 - \lambda_{j,i})2hc_{j,i})w_i + 2hg_{j,i}| \leq 2h\mu_{j,i}$$

First, we discuss the cases when $d_{j,i} = 0$ or $c_{j,i} = 0$. For these cases we set $\lambda_{j,i} = 1$. If $d_{j,i} = c_{j,i} = 0$, we have $|g_{j,i}| < \mu_{j,i}$ that is always true by Assumption 4. If $d_{j,i} = 0$ and $c_{j,i} \neq 0$ we have $|c_{j,i}w_{i+1} + g_{j,i}| \leq \mu_{j,i}$ that, after combining it with constraints (4.49), becomes $0 \leq w_{i+1} \leq \min\{\psi^2_{i+1}/\boldsymbol{\gamma}'^2_{j,i+1}, (\mu_{j,i} - g_{j,i})/|c_{j,i}|\}$. Finally, if $d_{j,i} \neq 0$ and $c_{j,i} = 0$ we have $|d_{j,i}(w_{i+1} - w_i) + 2hg_{j,i}| \leq 2h\mu_{j,i}$ that satisfies Assumption 5. If $d_{j,i} \neq 0$ and $c_{j,i} \neq 0$ we have:

$$-2h(\mu_{j,i}+g_{j,i}) \leq (d_{j,i}+2hc_{j,i}\lambda_{j,i})w_{i+1}+(-d_{j,i}+(1-\lambda_{j,i})2hc_{j,i})w_i \leq 2h(\mu_{j,i}-g_{j,i}).$$

In order to satisfy Assumption 5 we choose the value of $\lambda_{j,i}$ such that $(d_{j,i} + 2hc_{j,i}\lambda_{j,i})(-d_{j,i} + (1 - \lambda_{j,i})2hc_{j,i}) < 0$. Hence, we set

$$\lambda_{j,i} = \begin{cases} 1 & \text{if } \; d_{j,i}c_{j,i} > 0 \\ 0 & \text{if } \; d_{j,i}c_{j,i} < 0. \end{cases}$$

Using this selection technique we can rewrite constraint (4.45) as follows:

$$\tau_{j,i} = \begin{cases} d_{j,i}a_i + c_{j,i}w_{i+1} + g_{j,i}, & \text{if } d_{j,i}\ c_{j,i} > 0, \\ d_{j,i}a_i + c_{j,i}w_i + g_{j,i}, & \text{if } d_{j,i}\ c_{j,i} < 0. \end{cases}$$

Moreover, we can explicit constraints (4.51) in the form presented in (4.27). In fact, if $d_{j,i}\ c_{j,i} > 0$, the constraint (4.51) becomes:

$$\begin{aligned} w_{i+1} &\leq \frac{d_{j,i}}{d_{j,i} + 2hc_{j,i}} w_i + \left| \frac{2h(\mu_{j,i} - g_{j,i})}{d_{j,i} + 2hc_{j,i}} \right|, \\ w_i &\leq \frac{d_{j,i} + 2hc_{j,i}}{d_{j,i}} w_{i+1} + \left| \frac{2h(\mu_{j,i} + g_{j,i})}{d_{j,i}} \right| \end{aligned} \tag{4.54}$$

and, with $d_{j,i}\ c_{j,i} < 0$,

$$\begin{aligned} w_{i+1} &\leq \frac{d_{j,i} - 2hc_{j,i}}{d_{j,i}} w_i + \left| \frac{2h(\mu_{j,i} + g_{j,i})}{d_{j,i}} \right| \\ w_i &\leq \frac{d_{j,i}}{d_i - 2hc_{j,i}} w_{i+1} + \left| \frac{2h(\mu_{j,i} - g_{j,i})}{d_{j,i} - 2hc_{j,i}} \right|. \end{aligned} \tag{4.55}$$

We use the same reasoning for constraints (4.50). Consider

$$|(\gamma'_{j,i} + 2h\gamma''_i\eta_{j,i})w_{i+1} + (-\gamma'_{j,i} + (1 - \eta_{j,i})2h\gamma''_{j,i})w_i| \leq 2h\alpha_{j,i}$$

Again, setting $\eta_{j,i} = 1$, we discuss the cases when $\gamma'_{j,i} = 0$ or $\gamma''_{j,i} = 0$. If $\gamma'_{j,i} = \gamma''_{j,i} = 0$, we have $|0| \leq \alpha_{j,i}$ that is always true. If $\gamma'_{j,i} = 0$ and $\gamma''_{j,i} \neq 0$ we have $|\gamma''_{j,i}b_i| \leq \alpha_{j,i}$ that becomes $0 \leq w_{i+1} \leq \alpha_{j,i}/|\gamma''_{j,i}|$. Finally, if $\gamma'_{j,i} \neq 0$ and $\gamma''_{j,i} = 0$ we have $|w_{i+1} - w_i| \leq 2h\alpha_{j,i}/|\gamma'_{j,i}|$ that satisfies Assumption 5. After that, with $\gamma'_{j,i} \neq 0$ and $\gamma''_{j,i} \neq 0$ we set

$$\eta_{j,i} = \begin{cases} 1 & \text{if } \gamma'_{j,i}\ \gamma''_{j,i} > 0 \\ 0 & \text{if } \gamma'_{j,i}\ \gamma''_{j,i} < 0 \end{cases},$$

which implies, for $\gamma'_{j,i} \cdot \gamma''_{j,i} > 0$:

$$\begin{aligned} w_{i+1} &\leq \frac{\gamma'_{j,i}}{\gamma'_{j,i} + 2h\gamma''_{j,i}} w_i + \left| \frac{2h\alpha_{j,i}}{\gamma'_{j,i} + 2h\gamma''_{j,i}} \right| \\ w_i &\leq \frac{\gamma'_{j,i} + 2h\gamma''_{j,i}}{\gamma'_{j,i}} w_{i+1} + \left| \frac{2h\alpha_{j,i}}{\gamma'_{j,i}} \right| \end{aligned}, \tag{4.56}$$

and, with $\gamma'_{j,i} \cdot \gamma''_{j,i} < 0$,

$$\begin{aligned} w_{i+1} &\leq \frac{\gamma'_{j,i} - 2h\gamma''_{j,i}}{\gamma'_{j,i}} w_i + \left| \frac{2h\alpha_{j,i}}{\gamma'_{j,i}} \right| \\ w_i &\leq \frac{\gamma'_{j,i}}{\gamma'_{j,i} - 2h\gamma''_{j,i}} w_{i+1} + \left| \frac{2h\alpha_{j,i}}{\gamma'_{j,i} - 2h\gamma''_{j,i}} \right| \end{aligned} . \tag{4.57}$$

By observing relations (4.54)-(4.57), we notice that: i) the right-hand sides of these constraints are linear and, thus, also concave functions; ii) the coefficients of w_i and w_{i+1} in the right-hand sides are positive; iii) the constant terms are positive. Thus, all conditions stated in Assumption 5 are fulfilled. □

Then, we can prove the following theorem.

Theorem 5. *Algorithm 3 returns the optimal solutions of Problem 10 in time $O(np)$ if the 2-D LP problems are solved by Megiddo's algorithm.*

Proof. The fact that Algorithm 3 returns the optimal solution of Problem 10 is an immediate consequence of Propositions 21 and 24. Moreover, by Theorem 4, Algorithm 3 applied to Problem 10, where $t_i + r_i = 2p$, $i = 1, \ldots, n$, has time complexity $O(pn)$ if 2-D LP problems are solved by Megiddo's Algorithm. □

Remark 8. *The time complexity stated in Theorem 5 is optimal for Problem 10 in the sense that each solution algorithm for such problem requires reading its input data whose size is exactly $O(pn)$. Thus, Algorithm 3 could be possibly outperformed by another algorithm but in such case the computing times of the latter would differ with respect to those of Algorithm 3 by a constant factor as p and n vary.*

Remark 9. *Rather than using Megiddo's Algorithm, whose implementation is not straightforward, in our experiments we solve the 2-D LP problems by using Algorithm 4, whose worst-case complexity, when applied to the 2-D LP problems arising from Problem 10, is $O(p^2)$ (but, as we will see in the experiments, much better in practice), so that the overall worst-case complexity is $O(np^2)$, and*

by using Algorithm 5 whose worst-case complexity is $O(p\log(p))$, so that the overall complexity is $O(np\log(p))$. Such complexities are slightly worse than the optimal time complexity stated in Theorem 5 but in practice they are quite efficient and, as we will see in the experiments in Section 4.4, both Algorithm 4 and Algorithm 5 to solve 2-D LPs guarantee a better performance than the simplex algorithm.

In the current definition of Problem 10 we only imposed nonnegativity of the velocities. In practice, we might have also nonnegative lower bounds ℓ_i, $i = 1, \ldots, p$, for them. In such case Problem 10 is not necessarily feasible. However, Algorithm 3 can be employed also in this case, as stated in the following proposition.

Proposition 25. *If we impose nonnegative lower bounds ℓ_i, $i = 1, \ldots, p$, for the velocities, Algorithm 3 returns an optimal solution of Problem 10 if such problem is feasible, otherwise it is able to detect unfeasibility of the problem.*

Proof. One can solve the problem by Algorithm 3 only imposing nonnegativity of the velocities. Once the optimal solution w_i^*, $i = 1, \ldots, n$, has been obtained, if $w_i^* \geq \ell_i \; \forall i$, then it is also the optimal solution for the problem with the given lower bounds for the velocities. Instead, if for some i, $w_i^* < \ell_i$ holds, noting that, in view of Proposition 20, $w_i \leq w_i^*$ must hold at each feasible solution of Problem 10, we can immediately conclude that in such case the problem is unfeasible. □

Remark 10. *Note that Problems 9 and 10 can still have a solution even if Assumption 4 does not hold. In this case, in Assumption 5, some of conditions $f_j^i(0) > 0$, $i = 1, \ldots, n-1, \quad j = 1, \ldots, r_i$, $b_k^i(0) > 0$, $i = 1, \ldots, n-1, \quad k = 1, \ldots, t_i$ may be violated. As a consequence, the following anomalies can arise during the execution of Algorithm 3:*

- *the set of $x \in \mathbb{R}$ such that $F_i \circ B_i(x) > x$ may be empty,*
- *vector $\bar{u}$ computed by Algorithm 3 may have negative values.*

Note that if any of these two conditions is met during the solution of Algorithm 3, then Problem 10 is not feasible. On the other hand, if these conditions are not encountered, the solution $\bar{u}$ reported by Algorithm 3 is still an optimal solution of Problem 10, despite the fact that Assumption 4 is not satisfied. In other words, it is possible to modify Algorithm 3 to handle cases in which Assumption 4 does not hold. In view of this, Algorithm 3 needs be appropriately changed in order to detect the failure conditions reported above. Anyway, we prefer to keep Assumption 4 in order to maintain the current simple formulation of Algorithm 3. Note that TOPP-RA algorithm in [46] can correctly handle cases in which Assumption 4 is not satisfied.

4.3.2 Comparison with TOPP-RA algorithm

As already mentioned in the introduction, a very recent and interesting work, closely related to ours, is [46]. In that paper a backward-forward approach is proposed. In the backward phase a controllable set is computed for each discretization point. This is an interval that contains all possible states for which there exists at least a sequence of controls leading to the final assigned state. The computation of each interval requires the solution of two LP problems with two variables. Next, a forward phase is performed where a single LP with two variables is solved for each discretization point. The final result is a feasible solution which, however, is optimal under the assumption that no zero-inertia points (i.e., values of s such that $a(s) = 0$ in Problem 9) are present. In the presence of zero-inertia points a solution is returned whose objective function value differs from the optimal one by a quantity proportional to the discretization step h.

In what follows we outline the main differences between our approach and the one proposed in [46]. (i) The approach in [46] solves $2n$ 2-D LPs and n 1-D LPs, while ours solves n 2-D LPs. (ii) In [46] the 2-D LPs are solved by the simplex method while we proposed alternative methods which turn out to be more efficient. Indeed, our computational experiments will show that the computation times are reduced considerably when using our alternative

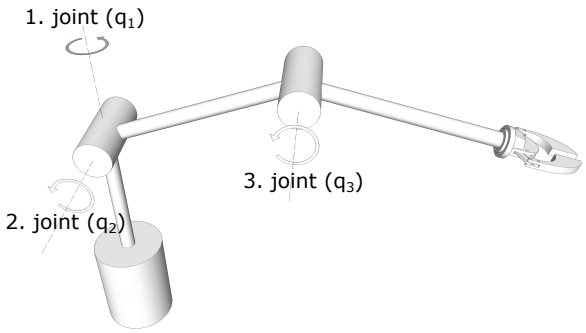


Figure 4.4: The 3-DoF manipulator with the three revolute joints ($\mathbf{q} = [q_1,\ q_2,\ q_3]^T$).

methods. (iii) In [46] it is observed that the practical (say, average) complexity of solving 2-D LPs is linear with respect to the number of constraints. In fact, we observed that such complexity is not only the practical one but also the worst-case one. (iv) Finally, in our approach we deal with the presence of zero-inertia points through the addition of the displacements (4.41). Introducing these displacement, we are able to return an exact solution of the discretized problem.

4.4 Experimental results

In this section, we consider a motion planning problem for a 3-DoF manipulator and compare the computation time of the proposed solver to other methods existing in literature a further real application on a 6-DOF robotic manipulator is reported in [53]. We consider the robot presented in [54] (Chapter 4, example 4.3). This robot is a serial chain robot (see Figure 4.4), composed of 3 links connected with 3 revolute joints (the first link is connected with a fixed origin). Table 4.1 reports the robot parameters. Namely, for link i, $i = 1, \ldots, 3$, l_i is the length, and r_i is the distance between the gravity center of the link and the joint that connects it to the previous link in the chain (see Figure 4.5). Parameters I_{xi}, I_{yi}, I_{zi}, m_i are the diagonal components of the inertia matrix

Table 4.1: Kinematic and dynamic parameters for the 3-DoF manipulator.

Link	(I_{xi}, I_{yi}, I_{zi}) $[kg\ m^2]$	m_i $[kg]$	l_i $[m]$	r_i $[m]$
1	(7.5, 7.5, 7.5)	1.5	0.2	0.08
2	(5.7, 5.7, 5.7)	1.2	0.3	0.12
3	(4.75, 4.75, 4.75)	1.0	0.325	0.13

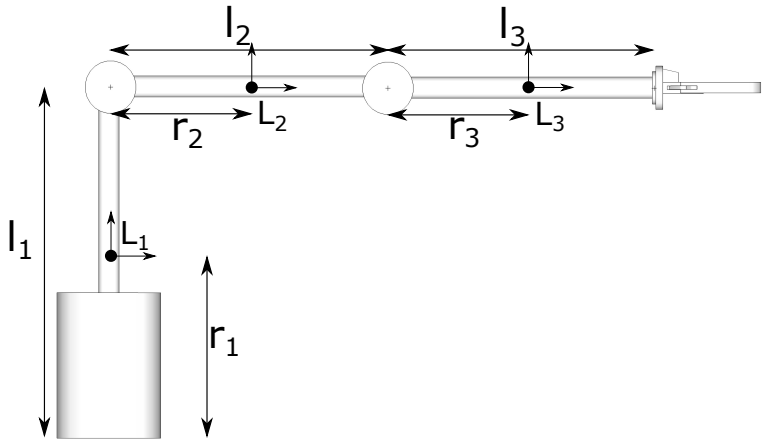


Figure 4.5: Kinematic and dynamic parameters for the 3-DoF manipulator. L_i indicates the coordinate frame attached to the gravity center of the link.

and the mass of link i.

We consider an instance of Problem 9, where the reference curve $\boldsymbol{\gamma} : [0, 1] \rightarrow \mathbb{R}^3$ is defined as a cubic spline that interpolates the points shown in Table 4.2. The mass matrix $\boldsymbol{D}$, the Coriolis matrix $\boldsymbol{C}$ and the external forces term $\mathbf{g}$ that we consider are reported in [54] (Chapter 4, example 4.3).

The following kinematic and dynamic bounds are applied for the presented

Table 4.2: Points interpolated in the configuration space.

s	γ_1	γ_2	γ_3
0	0	0	0
0.25	1.288	-0.2864	-0.2982
0.5	2.59	-0.03045	-0.5995
0.75	4.374	-0.04647	-0.582
1	5.334	-0.1657	-0.4504

test case ($\forall s \in [0, 1]$)

$$\begin{aligned}\boldsymbol{\psi}(s) =&[2.0,\ 2.0,\ 2.0]^T,\\ \boldsymbol{\alpha}(s) =&[1.5,\ 1.5,\ 1.5]^T,\\ \boldsymbol{\mu}(s) =&[9,\ 9,\ 9]^T.\end{aligned}$$

We find an approximated solution of Problem 9 by solving Problem 10 with five different methods:

1. a SOCP solver which solves the SOCP reformulation presented in Equation (74)-(86) of [41];

2. a LP solver which solves the LP reformulation presented in Equation (23) of [45];

3. Algorithm 3 using simplex method to solve the 2-D LP subproblems (4.30).

4. Algorithm 3 using Algorithm 4 to solve the 2-D LP subproblems (4.30).

5. Algorithm 3 using Algorithm 5 to solve the 2-D LP subproblems (4.30).

In the first and second method we use Gurobi solver [50] while, for the other methods, we use a C++ implementation of Algorithm 3. We measure the performance on a 2.4 GHz Intel Core i7-3630QM CPU.

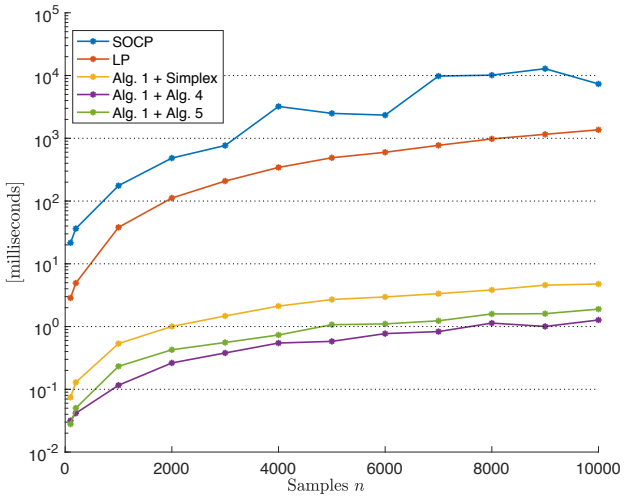


Figure 4.6: Computation times obtained using different approaches as a function of the number of sample points.

The results are presented in Figure 4.6 for values of n up to $n = 10,000$. Such results show that the Algorithm 3 with Algorithm 4 and Algorithm 5 employed to solve the 2-D LP subproblems perform quite similarly (the former slightly better than the latter). Both significantly outperform the first two methods (by approximately four and two order of magnitudes, respectively), and perform better than Algorithm 3 with the simplex method. In Figure 4.7, we show how the travelling time varies with the number n of discretization points. Note that the time tends to converge rather quickly as n increases and that the computed travelling time for each value of n is the same for all approaches we tested.

In order to evaluate the impact of the number of degrees of freedom p on the solution time, we ran a second set of tests, with a fixed number of samples $n = 500$ and where p varies from 10 to 100, along the lines of a similar test presented in [46]. For each test, we obtain the reference path $\boldsymbol{\gamma} : [0, 1] \to \mathbb{R}^p$ by randomly generating 5 vectors in $\mathbb{R}^p$, whose components $\gamma_i,\ i = 1, \ldots, p$

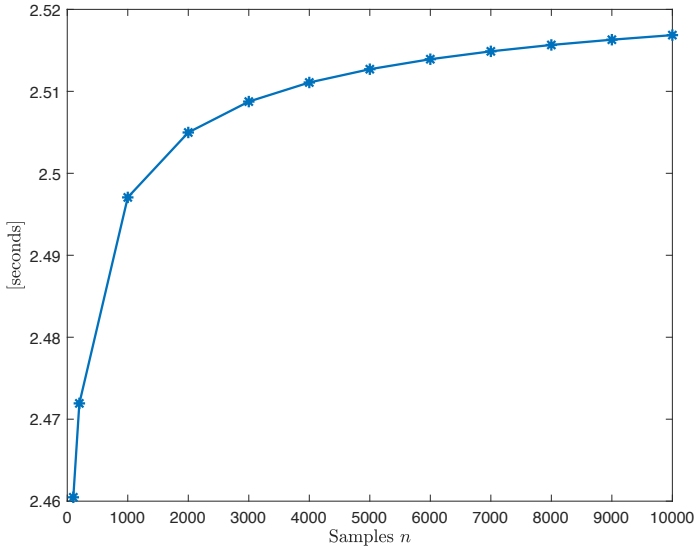


Figure 4.7: Minimum travelling time as a function of the the number of steps n.

are uniformly sampled in interval $[-5, 5]$. After that, we interpolate these 5 waypoints with a cubic-spline. In these tests, we do not impose any constraint on generalized forces and consider constant speed and accelerations constraints, given by $\psi_i(s) = 2.0$ and $\alpha_i(s) = 1.5$, for $s \in [0, 1]$ and $i = 1 \dots, p$. We restricted the attention to Algorithm 3 with 2-D LPs solved by the simplex algorithm, Algorithm 4, and Algorithm 5, respectively (we do not report the computing times for the SOCP and LP solver since these, as previously seen, are significantly worse).

The results are displayed in Figure 4.8. They confirm that both Algorithm 4 and Algorithm 5 outperform the simplex method, but, interestingly, it turns out that Algorithm 4 performs significatively better than Algorithm 5 as p increases, in spite of the fact that its worst-case complexity, namely $O(p^2)$, is inferior to the $O(p \log(p))$ complexity of Algorithm 5. This can be explained by observing that the complexity of Algorithm 5 is not only the worst-case one but also the practical one, since it depends on the sorting operations which

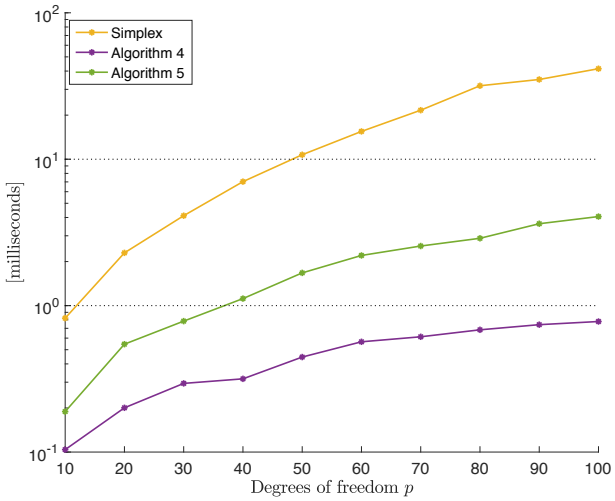


Figure 4.8: Computation times obtained using different approaches with $n = 500$ and increasing p.

always need to be performed. Instead, the average performance of Algorithm 4 can be much better than its worst-case performance and, in fact, Figure 4.8 suggests a linear increase with respect to p. Note that we employed our own implementation of the primal simplex algorithm for 2-D LPs. We did not employ commercial solvers since, for such low dimensional problems, they require a much larger model creation times than the (very small) solution times, thus making their use in this context inefficient. Of course, our implementation may be improved. However, the larger computing times of the primal simplex algorithm are probably due to the fact that its starting point is the origin, which is the lowermost feasible point, while the optimal solution of these 2-D LPs is always the uppermost feasible point, so that, on average, an $O(p)$ number of iterations is needed before reaching the optimal solution. We did not compute directly the solution with the TOPP-RA algorithm presented in [46], however, note that the computational time of TOPP-RA is comparable with Algorithm 3 using simplex method to solve the 2-D LP subproblems (4.30) (actually higher,

since TOPP-RA solves $3n$ LP problems, while our approach solves only n LP problems).

4.5 Appendix

Proof of Proposition 18 Let $\mathcal{D}$ be the subset of $C^1([0, s_f], \mathbb{R})$ that satisfies conditions (4.11)-(4.17). Set $f(w) = \int_0^{s_f} w(s)^{-1/2}\, ds$ and $f^* = \inf\{f(b) : w \in \mathcal{D}\}$, then there exists a sequence of $w_i : [0, s_f] \to \mathbb{R}$, $i \in \mathbb{N}$, such that $w_i \in \mathcal{D}$ and $\lim_{i\to\infty} f(w_i) = f^*$. By Ascoli-Arzelà theorem, if the sequence $\{w_i\}$ is uniformly bounded and differentiable with $\{w_i'\}$ uniformly bounded, then there exists a subsequence $\{w_{i_k}\}$ that uniformly converges on $[0, s_f]$. Since $(\forall s \in [0, s_f])\ \|\boldsymbol{\gamma}'(s)\| = 1$, there exists an index $i(s) \in \{1, \dots, p\}$ such that $\gamma'_{i(s)}(s)^2 \geq \frac{1}{p}$.Then, we define a function $\beta : [0, s_f] \to \mathbb{R}_+$ as the most restrictive upper bound of w along the path. Hence, from constraint (4.15) we can write the following relation

$$0 \leq w(s) \leq \beta(s). \tag{4.58}$$

Since each function w_i is uniformly bounded (by (4.15) and boundedness of β) and differentiable, it remains to show that w_i' is uniformly bounded (i.e., there exists a real constant C such that, $\forall s \in [0, s_f]$, $|w_i'(s)| \leq C$). Consider constraint (4.16)

$$-2(\boldsymbol{\alpha}(s) + \boldsymbol{\gamma}''(s)w(s)) \leq\leq \boldsymbol{\gamma}'(s)w'(s) \leq 2(\boldsymbol{\alpha}(s) - \boldsymbol{\gamma}''(s)w(s)).$$

Since $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are bounded functions, $\bar{\alpha} = \|\boldsymbol{\alpha}\|_\infty < +\infty$ and $\bar{\beta} = \|\boldsymbol{\beta}\|_\infty < +\infty$. Moreover, $\boldsymbol{\gamma}''$ is a continuous function on the compact set $[0, s_f]$, then there exists the component-wise maximum $\bar{\gamma}'' = \|\boldsymbol{\gamma}''\|_\infty$. Hence, we bound $\boldsymbol{\gamma}'(s)w'(s)$ as follows

$$-2(\bar{\alpha} + \bar{\gamma}''\bar{\beta})\mathbf{1} \leq \boldsymbol{\gamma}'(s)w'(s) \leq 2(\bar{\alpha} + \bar{\gamma}''\bar{\beta})\mathbf{1}.$$

Since $\forall s \in [0, s_f]\ \|\boldsymbol{\gamma}'(s)\|_2 = 1$, then there exists an index $i(s) \in \{1, \dots, p\}$ such that $\|\boldsymbol{\gamma}'_{i(s)}(s)\|_\infty \geq \frac{1}{\sqrt{p}}$, which implies

$$-2\sqrt{p}(\bar{\alpha}+\bar{\gamma}''\bar{\beta}) \leq -2\frac{(\bar{\alpha}+\bar{\gamma}''\bar{\beta})}{\|\gamma'_{i(s)}(s)\|_\infty} \leq w'(s) \leq \frac{(\bar{\alpha}+\bar{\gamma}''\bar{\beta})}{\|\gamma'_{i(s)}(s)\|_\infty} \leq 2\sqrt{p}(\bar{\alpha}+\bar{\gamma}''\bar{\beta}).$$

Hence, $|w'|$ is uniformly bounded by the real constant $C = 2\sqrt{p}(\bar{\alpha}+\bar{\gamma}''\bar{\beta})$.

To show that $f^*(w) \leq U < +\infty$, where U is a constant depending on the problem data, it is sufficient to find $w \in \mathcal{D}$ such that $f(w) < +\infty$. To this end, set for $\delta \geq 0$

$$w_\delta(s) = \begin{cases} \delta s(2\delta - s), & s \in [0, \delta), \\ \delta^3, & s \in [\delta, s_f - \delta], \\ \delta(s_f - s)(s - s_f + 2\delta), & s \in (s_f - \delta, s_f]. \end{cases}$$

Its derivative is

$$w'_\delta(s) = \begin{cases} 2\delta(\delta - s), & s \in [0, \delta), \\ 0, & s \in [\delta, s_f - \delta], \\ 2\delta(s_f - s - \delta), & s \in (s_f - \delta, s_f]. \end{cases}$$

Note that (4.17) is obviously satisfied by w_δ, moreover, $w_\delta \in \mathcal{D}$ if $\delta = 0$.

The maximum value of b_δ is δ^3. By Assumption 3, there exists the minimum $\hat{\psi} = \min_{i=1,\dots,p} \min\{\boldsymbol{\psi}_i(s) : s \in [0, s_f]\} > 0$. Since $\boldsymbol{\gamma} \in C^2([0, s_f], \mathbb{R}^p)$ it follows that $\hat{\gamma}' = \|\boldsymbol{\gamma}'\|^2_\infty < +\infty$. Hence, setting $\hat{\delta} = (\hat{\psi}/\hat{\gamma}')^{\frac{2}{3}}$, (4.15) is satisfied for any $\delta \in [0, \hat{\delta}]$. After that, we have that $(\forall s \in [0, s_f])$

$$\left|\frac{1}{2}\mathbf{d}(s)w'_\delta(s) + \mathbf{c}(s)w_\delta(s) + \mathbf{g}(s)\right| \leq \frac{1}{2}|\mathbf{d}(s)|\,|w'_\delta(s)| + |\mathbf{c}(s)|w_\delta(s) + |\mathbf{g}(s)|.$$

By Assumption 4 it follows that

$$\begin{aligned} &\frac{1}{2}|\mathbf{d}(s)|\,|w'_\delta(s)| + |\mathbf{c}(s)|w_\delta(s) + |\mathbf{g}(s)| < \\ &< \frac{1}{2}|\mathbf{d}(s)|\,|w'_\delta(s)| + |\mathbf{c}(s)|w_\delta(s) + \boldsymbol{\mu}(s) - \varepsilon\mathbf{1}. \end{aligned}$$

There exists $\bar{\delta} > 0$ such that $(\forall s \in [0, s_f])$

$$\frac{1}{2}|\mathbf{d}(s)||w'_{\bar{\delta}}(s)| + |\mathbf{c}(s)|w_{\bar{\delta}}(s) \leq \varepsilon \mathbf{1},$$

which implies that for any $\delta \in [0, \bar{\delta}]$

$$\left|\frac{1}{2}\mathbf{d}(s)w'_{\delta}(s) + \mathbf{c}(s)w_{\delta}(s) + \mathbf{g}(s)\right| \leq \boldsymbol{\mu}(s),$$

i.e., w_δ satisfies constraints (4.14). Analogously one can see that constraint (4.16) is fulfilled for each $\delta \in [0, \tilde{\delta}]$ with a sufficiently small $\tilde{\delta} > 0$. Hence, for each $\delta \in [0, \delta^*]$ with $\delta^* = \min\{\hat{\delta}, \bar{\delta}, \tilde{\delta}\}$, it follows that $w_\delta \in \mathcal{D}$. Finally, by direct computation, it is straightforward to see that $f(w_\delta) < +\infty$, with $\delta > 0$.

Chapter 5

A sequential algorithm for jerk limited minimum-time speed planning

In the previous chapters we carried out the problem of finding the speed profile for autonomous vehicles and robotic manipulators in order to minimize the traversed time on an assigned path. Both these problems were formulated taking into account only the constraints on the speed, on the accelerations and on the torques. However, in a practical scenario the resulting speed profile produced by the previously proposed approaches may not be sufficiently smooth. A certain degree of smoothness is a feature that is generally requested when we want to guarantee some comfort constraints, to avoid mechanical solicitations due to the saturation of the actuators, and to obtain better control performance.

The aim of this chapter is to add jerk constraints to the minimum time speed planning problem considered in Chapter 2, i.e., to add a limitation on the second derivative of the longitudinal speed, in order to generate a sufficiently smooth profile. We propose an efficient algorithm which is able to find a solution to a finite dimensional version of the minimum time problem.

5.1 Problem formulation

In Section 2.1 we showed how to formulate a minimum time speed planning problem by using the arc-length parametrization of the assigned path $\boldsymbol{\gamma}$. The problem has been converted into a convex one by the change of variable (2.5). We recall that through this formulation, the longitudinal acceleration a_L along the curve $\boldsymbol{\gamma}$ can be expressed as

$$a_L(t) = \frac{1}{2} w'(s(t)),$$

where s is the curvilinear coordinate and w is the square of the speed. By this parametrization, we can derive the jerk as follows

$$j_L(t) = \dot{a}_L(t) = \frac{1}{2} w''(s(t)) \sqrt{w(s(t))}.$$

Then, the minimum-time problem becomes:

Problem 11 (Smooth minimum-time speed planning problem: continuous version)**.**

$$\min_{w \in C^2} \int_0^{s_f} w(s)^{-1/2} \, ds$$

such that

$$\begin{aligned}
& w(0) = 0, \quad w(s_f) = 0, \\
& 0 \leq w(s) \leq \mu^+(s), && s \in [0, s_f], \\
& \frac{1}{2} \left| w'(s) \right| \leq A, && s \in [0, s_f], && (5.1) \\
& \frac{1}{2} \left| w''(s) \sqrt{w(s)} \right| \leq J, && s \in [0, s_f], && (5.2)
\end{aligned}$$

where μ^+ is the square speed upper bound depending on the shape of the path, i.e.,

$$\mu^+(s) = \min \left\{ v_{\max}^2(s), \frac{A_N}{|k(s)|} \right\},$$

with $v_{\max}$, A_N and k be the maximum allowed speed of the vehicle, the maximum normal acceleration and the curvature of the path, respectively. Parameters A and J are the bounds representing the limitations on the acceleration and the jerk, respectively. As in Chapter 2 we solve a finite dimensional version of the problem, namely:

Problem 12 (Smooth minimum-time speed planning problem: discretized version)**.**

$$\min_{\mathbf{w}\in\mathbb{R}^n} 2h \sum_{i=1}^{n-1} \frac{1}{\sqrt{w_{i+1}} + \sqrt{w_i}} \tag{5.3}$$

such that

$$0 \leq \mathbf{w} \leq \mathbf{u}, \tag{5.4}$$

$$w_{i+1} - w_i \leq 2hA, \qquad i = 1, \ldots, n-1, \tag{5.5}$$

$$w_i - w_{i+1} \leq 2hA, \qquad i = 1, \ldots, n-1, \tag{5.6}$$

$$(w_{i-1} - 2w_i + w_{i+1})\sqrt{\frac{w_{i+1} + w_{i-1}}{2}} \leq 2h^2 J \qquad i = 2, \ldots, n-1, \tag{5.7}$$

$$-(w_{i-1} - 2w_i + w_{i+1})\sqrt{\frac{w_{i+1} + w_{i-1}}{2}} \leq 2h^2 J \qquad i = 2, \ldots, n-1, \tag{5.8}$$

where $u_1 = 0$ and $u_n = 0$, since we are assuming that the initial and final speed are equal to 0. Constraints (5.7) and (5.8) are obtained by using a second-order central finite difference to approximate w'', while w is approximated by the arithmetical mean. Due to jerk constraints (5.7) and (5.8), it follows that Problem 12 becomes a non-convex problem and loses the special structure of the problems we solved in Chapters 2 and 4. That makes even the detection of a local minimizers much harder with respect to those problems.

In the next sections, we propose to solve Problem 12 with a line-search algorithm based on the sequential solution of convex problems. The main contribution of the chapter lies in the procedure that we provide to efficiently solve such convex problems. We will use the results presented in Chapters 2

and 4 and in references [22, 55] in order to exploit as much as possible their properties.

5.2 A sequential algorithm based on constraint linearization

To account for the non-convexity of Problem 12 we propose a line search method (see Algorithm 6, which flow chart is shown in Figure 5.1). Let us denote by Ω the feasible region of Problem 12. At each iteration k, we replace the current $\mathbf{w}^{(k)} \in \Omega$ with $\mathbf{w}^{(k)}+\alpha^{(k)}\boldsymbol{\delta}\mathbf{w}^{(k)} \in \Omega$, where the step-size $\alpha^{(k)} \in [0,1]$ is obtained by a *line search* along the descent direction $\boldsymbol{\delta}\mathbf{w}^{(k)}$. Such descent direction $\boldsymbol{\delta}\mathbf{w}^{(k)}$ is obtained thorugh the solution of a convex problem (see Problem 13), which constraints are linear approximations of (5.4)-(5.8) around $\mathbf{w}^{(k)}$, while the objective function is the original one. Then, the problem we consider to compute the direction $\boldsymbol{\delta}\mathbf{w}^{(k)}$ is the following:

Problem 13.

$$\min_{\boldsymbol{\delta}\mathbf{w}\in\mathbb{R}^n} 2h \sum_{i=1}^{n-1} \frac{1}{\sqrt{w_{i+1}+\delta w_{i+1}}+\sqrt{w_i+\delta w_i}} \tag{5.9}$$

such that

$$\mathbf{l_B} \leq \boldsymbol{\delta}\mathbf{w} \leq \mathbf{u_B}, \tag{5.10}$$

$$\delta w_{i+1} - \delta w_i \leq b_{A_i}, \qquad i = 1,\dots,n-1, \tag{5.11}$$

$$\delta w_i - \delta w_{i+1} \leq b_{Di}, \qquad i = 1,\dots n-1, \tag{5.12}$$

$$\delta w_i - \eta_i \delta w_{i-1} - \eta_i \delta w_{i+1} \leq b_{N_i}, \qquad i = 2,\dots,n-1. \tag{5.13}$$

$$\eta_i \delta w_{i-1} + \eta_i \delta w_{i+1} - \delta w_i \leq b_{P_i}, \qquad i = 2,\dots,n-1, \tag{5.14}$$

where $\mathbf{l_B} = -\mathbf{w}^{(k)}$ and $\mathbf{u_B} = \mathbf{u} - \mathbf{w}^{(k)}$, while parameters $\boldsymbol{\eta}$, $\mathbf{b}_\mathrm{A}$ $\mathbf{b}_\mathrm{D}$, $\mathbf{b}_\mathrm{N}$ and $\mathbf{b}_\mathrm{P}$ depend on the point $\mathbf{w}^{(k)}$ around which the constraints (5.4)-(5.8) are linearized (see proof of Proposition 26). Note that the proposed approach follows some standard ideas of sequential quadratic approaches employed in

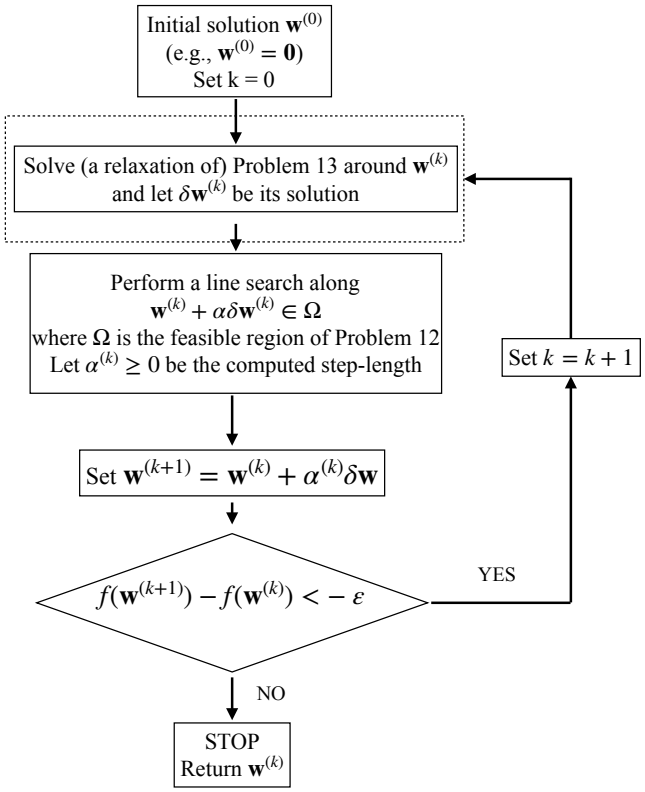


Figure 5.1: Flow chart of Algorithm 6. The dashed block corresponds to a call of the procedure `ComputeUpdate`, proposed to solve Problem 13, which represents the main contribution of this chapter.

the literature about nonlinearly constrained problems, but a quite relevant difference is that the true objective function (5.3) is employed in the problem to compute the direction, rather than a quadratic approximation of such function. This choice comes from the fact that the objective function (5.3) has some features (in particular, convexity and non-increasingness), which, combined with the structure of the linearized constraints, allow for an efficient solution of Problem 13. We make the following observation.

Proposition 26. *All parameters* $\boldsymbol{\eta}$, $\mathbf{b}_A$ $\mathbf{b}_D$, $\mathbf{b}_N$ *and* $\mathbf{b}_P$ *are non negative for*

$h \to 0$.

Proof. Let us consider constraints (5.5). By linearizing them around $\mathbf{w} \in \Omega$ we have

$$\delta w_{i+1} - \delta w_i \leq b_{\mathrm{A}_i},$$

where $b_{\mathrm{A}_i} = 2hA - w_{i+1} + w_i \geq 0$ because of the feasibility of $\mathbf{w}$. Analogously we can prove that $b_{\mathrm{D}i} = 2hA - w_i + w_{i+1} \geq 0$.

After that, by linearizing constraints (5.7) around $\mathbf{w}$ we have:

$$\eta_i = \frac{3(w_{i+1} + w_{i-1}) - 2w_i}{4(w_{i+1} + w_{i-1})}$$

and

$$b_{\mathrm{P}_i} = \frac{\sqrt{2}h^2 J - (w_{i-1} - 2w_i + w_{i+1})\sqrt{w_{i+1} + w_{i-1}}}{2\sqrt{w_{i+1} + w_{i-1}}}.$$

By continuity of w we have that, for $h \to 0$, $\eta_i \to \frac{1}{2}$, whereas for the feasibility of $\mathbf{w}$ we have $b_{\mathrm{P}_i} \geq 0$. Analogously we can prove that $b_{\mathrm{N}_i} \geq 0$. □

Algorithm 6: Sequential Convex algorithm

```
1 Find w^(1) ∈ Ω;
2 for k = 1... do
3     Define Problem 13 depending on w^(k);
4     Set δw^(k) = ComputeUpdate(w^(k));
5     Find α^(k) such that w^(k) + α^(k) δw^(k) ∈ Ω;
6     Set w^(k+1) = w^(k) + α^(k) δw^(k);
7     if f(w^(k+1)) − f(w^(k)) < −ε then
8         return w^k
```

Problem 13 is a convex problem with a non-empty feasible region ($\boldsymbol{\delta}\mathbf{w} = \mathbf{0}$ is always a feasible solution) and, consequently, can be solved by existing non-linear solvers. However, such solvers tend to increase computing times since

they need to be called many times within an iterative algorithm (see Algorithm 6). The main contribution of this chapter is routine `computeUpdate` (see dashed block in Figure 5.1), which is able to solve Problem 13 and efficiently returns a descent direction $\boldsymbol{\delta}\mathbf{w}^{(k)}$. In fact, to be more precise, we will solve a *relaxation* of Problem 13. Such relaxation as well as the routine to solve it, will be detailed in Sections 5.3 and 5.4. In Section 5.3 we present efficient approaches to solve some subproblems including proper subsets of the constraints. Then, in Section 5.4 we address the solution of the relaxation of Problem 13.

Remark 11. *It is possible to see that if one of the constraints* (5.7)-(5.8) *is active at* $\mathbf{w}^{(k)}$*, then along the direction* $\boldsymbol{\delta}\mathbf{w}^{(k)}$ *computed through the solution of the linearized Problem 13, it holds that* $\mathbf{w}^{(k)} + \alpha\boldsymbol{\delta}\mathbf{w}^{(k)} \in \Omega$ *for any sufficiently small* $\alpha > 0$*. In other words, small perurbations of the current solution* $\mathbf{w}^{(k)}$ *along direction* $\boldsymbol{\delta}\mathbf{w}^{(k)}$ *do not lead outiside the feasible region* Ω*. This fact is illustrated in Figure 5.2. Let us rewrite constraints* (5.7)-(5.8) *as follows:*

$$|(x - 2y)\sqrt{x}| \leq C, \tag{5.15}$$

where $x = w_{i+1} + w_{i-1}$*,* $y = w_i$ *and* C *is a constant. The feasible region associated to constraint* (5.15) *is reported in Figure 5.2. In particular, it is the region between the blue and the red line. Suppose that constraint* $y \leq \frac{x}{2} + \frac{C}{2\sqrt{x}}$ *is active at* $\mathbf{w}^{(k)}$ *(the case when* $y \leq \frac{x}{2} - \frac{C}{2\sqrt{x}}$ *is active can be dealt with in a completely analogous way). If we linearize such constraint around* $\mathbf{w}^{(k)}$*, then we obtain a linear constraint (black line in Figure 5.2) which defines a region completely contained into the one defined by the nonlinear constraint* $y \leq \frac{x}{2} + \frac{C}{2\sqrt{x}}$*. Hence, for each direction* $\boldsymbol{\delta}\mathbf{w}^{(k)}$ *feasible with respect to the linearized constraint, we are always able to perform sufficiently small steps, without violating the original nonlinear constraints, i.e., for* $\alpha > 0$ *small enough, it holds that* $\mathbf{w}^{(k)} + \alpha\boldsymbol{\delta}\mathbf{w}^{(k)} \in \Omega$*.*

Remark 12. *In the rest of this chapter we call constraints* (5.11)-(5.12) *acceleration constraints, while constraints* (5.13) *and* (5.14) *are called Negative*

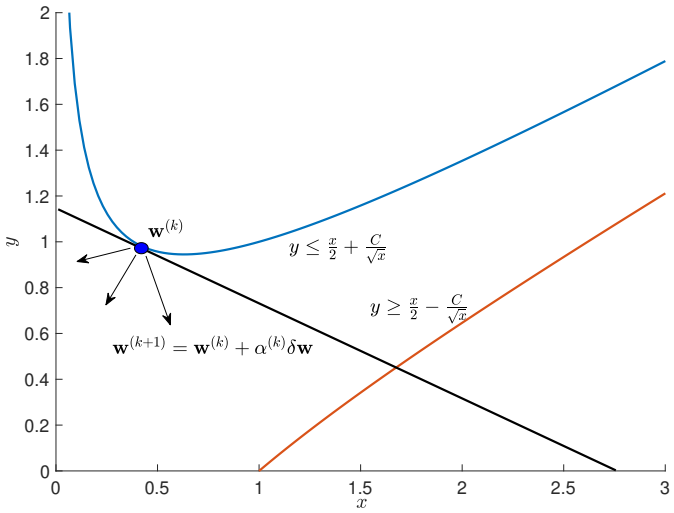


Figure 5.2: Constraints (5.7)-(5.8) and their linearization.

Acceleration Rate (PAR) and Positive Acceleration Rate (NAR) constraints, respectively.

5.3 The cases with acceleration and NAR constraints

In this section we show that if we consider only the subset of constraints (5.10)-(5.13), then Problem 13 can be efficiently solved. In what follows, besides the lower and upper bound constraints (5.10), we carry out the case including only the acceleration constraints (5.11) and (5.12), the case including only the NAR constraints (5.13), and, finally, the case including all constraints (5.10)-(5.13).

5.3.1 Acceleration constraints

The simplest case is the one where we only consider the acceleration constraints (5.11) and (5.12), besides constraints (5.10) with upper bound vector $\mathbf{y}$. The problem to be solved is:

Problem 14.

$$\min_{\boldsymbol{\delta}\mathbf{w}\in\mathbb{R}^n} 2h\sum_{i=1}^{n-1}\frac{1}{\sqrt{w_{i+1}+\delta w_{i+1}}+\sqrt{w_i+\delta w_i}}$$

such that

$$\begin{aligned}
&\mathbf{l_B} \le \boldsymbol{\delta}\mathbf{w} \le \mathbf{y}, && \\
&\delta w_{i+1} - \delta w_i \le b_{A_i}, && i = 1,\dots,n-1,\\
&\delta w_i - \delta w_{i+1} \le b_{Di}, && i = 1,\dots n-1.
\end{aligned}$$

It belongs to the class of Problem 4, presented in Chapter 2. As already proved, we are able to detect its optimal solution by Algorithm 7, which computational complexity is $O(n)$.

Algorithm 7: Routine `SolveAcc` for the solution of the problem with acceleration constraints

input : The upper bound $\mathbf{y}$
output: The descent direction $\boldsymbol{\delta}\mathbf{w}$

1 $\delta w_1 = 0$, $\delta w_n = 0$;
2 **for** $i = 1$ **to** $n-1$ **do**
3 $\quad \delta w_{i+1} = \min\left\{\delta w_i + b_{\mathrm{A}_i}, y_{i+1}\right\}$
4 **for** $i = n-1$ **to** 1 **do**
5 $\quad \delta w_i = \min\left\{\delta w_{i+1} + b_{\mathrm{D}i}, y_i\right\}$
6 **return** $\boldsymbol{\delta}\mathbf{w}$

5.3.2 NAR constraints

Now, we consider the problem only including NAR constraints (5.13) and constraints (5.10) with upper bound vector $\mathbf{y}$:

Problem 15.

$$\min_{\boldsymbol{\delta}\mathbf{w}\in\mathbb{R}^n} 2h\sum_{i=1}^{n-1}\frac{1}{\sqrt{w_{i+1}+\delta w_{i+1}}+\sqrt{w_i+\delta w_i}}$$

such that

$$0 \le \delta w_i \le y_i, \qquad i = 1,\dots,n \quad (5.16)$$

$$\delta w_i \le \eta_i(\delta w_{i-1}+\delta w_{i+1}) + b_{N_i}, \qquad i = 2,\dots,n-1, \quad (5.17)$$

where $y_1 = y_n = 0$ because of the boundary conditions. We first prove the following proposition.

Proposition 27. *The optimal solution of Problem 15 is the maximum componentwise of its feasible region.*

Proof. It follows from the fact that Problem 15 belongs to the class of problems addressed in [55]. Indeed, we can write constraints (5.16) and (5.17) as follows

$$\delta w_i \le \min\left\{\eta_i(\delta w_{i-1}+\eta_i\delta w_{i+1}) + b_{\mathrm{N}_i},\ y_i\right\}, \quad i = 2,\dots,n-1,$$

which, thanks to the nonnegativity of the coefficients η_i and b_{N_i}, fulfill the assumptions required in [55]. □

By Proposition 27 it follows that Problem 15 can be solved by using the graph-based approach presented in [22, 55]. However, reference [22] shows that, by exploiting the structure of a simpler version of the NAR constraints, it is possible to develop an algorithm more efficient than the graph-based one. Our purpose is to extend the results presented in reference [22] to a case with different and more challenging NAR constraints, in order to develop an efficient algorithm outperforming the graph-based one. We first make the following remark.

Remark 13. *As a consequence of Proposition 27 it follows that* $\delta w_1^* = \delta w_n^* = 0$ *and, for each* $i = 2,\dots,n-1$*, either* $\delta w_i^* \le y_i$ *or* $\delta w_i^* \le \eta_i(\delta w_{i+1}^* + \delta w_{i-1}^*) + b_{N_i}$

holds as an equality. Indeed, in case both inequalities were strict for some i, then we could decrease the objective function value by increasing the value of δw_i^*.

In view of Remark 13 we have that, given two indexes s and t, with $s < t$, if $\delta w_s^* = y_s$ and $\delta w_t^* = y_t$, then there exists an index j, with $s < j \leq t$, such that $\delta w_j^* = y_j$ and such that each NAR constraint between $s+1$ and $j-1$ is active. If index j is known, then the following observation allows to return the componets of the optimal solutions between s and j.

Observation 1. *Let $\boldsymbol{\delta}\mathbf{w}^*$ be the optimal solution of Problem 15 and let $s < j$. If constraints $\delta w_s^* \leq y_s$, $\delta w_j^* \leq y_j$, and $\delta w_i^* \leq \eta_i(\delta w_{i+1}^* + \delta w_{i-1}^*) + b_{N_i}$, for $i = s+1, \ldots, j-1$, are all active, then $\delta w_{s+1}^*, \ldots, \delta w_{j-1}^*$ are obtained by the solution of the following tridiagonal linear system:*

$$\mathbf{A} \begin{bmatrix} \delta w_{s+1}^* \\ \vdots \\ \delta w_{j-1}^* \end{bmatrix} = \mathbf{q}, \tag{5.18}$$

where

$$\mathbf{A} = \begin{bmatrix} 1 & -\eta_{s+1} & 0 & \cdots & 0 \\ -\eta_{s+2} & 1 & -\eta_{s+2} & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -\eta_{j-2} & 1 & -\eta_{j-2} \\ 0 & \cdots & 0 & -\eta_{j-1} & 1 \end{bmatrix} \text{ and } \mathbf{q} = \begin{bmatrix} b_{Ns+1} + \eta_{s+1} y_s \\ b_{Ns+2} \\ \vdots \\ b_{Nj-2} \\ b_{Nj-1} + \eta_{j-1} y_j \end{bmatrix}.$$

Note that $\mathbf{A}$ is the square submatrix of the NAR constraints restricted to rows $s+1$ up to $j-1$ and the related columns.

In order to detect the index $j \in \{s+1, \ldots, t-1\}$ such that all NAR constraints $s+1, \ldots, j-1$ are active, we propose Algorithm 8, described in what follows. We initially set $j = t$. Then, at each iteration we solve linear system (5.18). Let $\bar{\mathbf{x}} = (\bar{x}_{s+1}, \ldots, \bar{x}_{j-1})$ be its solution. We check if it is feasible

and optimal. Namely, if there exists $k \in \{s+1, \dots, j-1\}$ such that either $\bar{x}_k < 0$ or $\bar{x}_k > u_k$, then $\bar{\mathbf{x}}$ is unfeasible. Conversely, if $0 \leq \bar{x}_k \leq u_k$, for $k = s+1, \dots, j-1$, then we verify if $\bar{\mathbf{x}}$ is the best possible solution over the interval $\{s+1, \dots, j-1\}$, namely, if there does not exist a better solution such that at least one NAR constraint is not active in such interval. Let $\bar{\mathbf{u}} = (u_{s+1}, \dots u_{j-1})$, and consider the following problem

$$\begin{aligned} \max_{\boldsymbol{\epsilon} \geq 0} \quad & \mathbf{1}^T(\bar{\mathbf{x}} + \boldsymbol{\epsilon}) \\ & \mathbf{A}(\bar{\mathbf{x}} + \boldsymbol{\epsilon}) \leq \mathbf{q} \\ & \mathbf{0} \leq \bar{\mathbf{x}} + \boldsymbol{\epsilon} \leq \bar{\mathbf{u}}. \end{aligned} \tag{5.19}$$

By construction we have $\mathbf{A}\bar{\mathbf{x}} = \mathbf{q}$, so that problem (5.19) can be rewritten as follows:

$$\begin{aligned} \max_{\boldsymbol{\epsilon} \geq 0} \quad & \mathbf{1}^T \boldsymbol{\epsilon} \\ & \mathbf{A}\epsilon \leq 0, \\ & -\bar{\mathbf{x}} \leq \boldsymbol{\epsilon} \leq \bar{\mathbf{u}} - \bar{\mathbf{x}}. \end{aligned} \tag{5.20}$$

If $\boldsymbol{\epsilon} = \mathbf{0}$ is the optimal solution of problem (5.20), then $\bar{\mathbf{x}}$ is optimal over the interval $\{s+1, \dots, j-1\}$. Indeed, recall that the optimal solution is the component-wise maximum over the feasible region. So, we need to verify whether $\boldsymbol{\epsilon} = \mathbf{0}$ is a KKT point. Since at $\boldsymbol{\epsilon} = \mathbf{0}$ constraints $-\mathbf{z} \leq \boldsymbol{\epsilon} \leq \bar{\mathbf{u}} - \mathbf{z}$ cannot be active, then the KKT conditions for problem (5.20) are the following ones:

$$\mathbf{A}^T \boldsymbol{\lambda} = \mathbf{1}, \ \boldsymbol{\lambda} \geq \mathbf{0}. \tag{5.21}$$

If conditions (5.21) do not hold, then $\bar{\mathbf{x}}$ is not optimal. In this case and in the previously discussed unfeasible case, we reduce j by 1. Otherwise, if conditions (5.21) are satisfied, then $\bar{\mathbf{x}}$ is optimal over the interval $\{s+1, \dots, j-1\}$ and we can set $\delta w_k^* = \bar{x}_k$ for $k = s+1, \dots, j-1$. After that, we recursively call the routine `SolveNAR` on the remaining subinterval $\{j, \dots, t\}$ in order to obtain the solution over the full interval.

Remark 14. *In Algorithm 8 routine* `isFeasible` *is the routine used to verify if, for* $k = s+1, \dots, j-1$, $0 \leq \bar{x}_k \leq u_k$, *while* `isOptimal` *is the procedure to check optimality of* $\bar{\mathbf{x}}$ *over the interval* $\{s+1 \dots, j-1\}$.

Now, we are ready to prove that Algorithm 8 solves Problem 15.

Proposition 28. *The call* `solveNAR(`$\mathbf{y}$`,1,n)` *of Algorithm 8 returns the optimal solution of Problem 15.*

Proof. The proof is straightforward in view of Remark 13 and observing that $\delta w_1^* = y_1 = 0$ and $\delta w_n^* = y_n = 0$. □

Algorithm 8: `SolveNAR`$(\mathbf{y}, s, t)$

input : Upper bound $\mathbf{y}$ and two indices s and t with $1 \leq s < t \leq n$
output: The solution $\boldsymbol{\delta}\mathbf{w}^*$ of Problem 15 if $\delta w_s^* = y_s$ and $\delta w_t^* = y_t$.

```
Set j = t;
δw* = y;
while j ≥ s + 1 do
    Compute the solution x̄ of the linear system (5.18);
    if isFeasible(x̄) and isOptimal(x̄) then
        Break;
    else
        Set j = j − 1;
for i = s + 1, ..., j − 1 do
    Set δw*_i = x̄_i;
return δw* = min{δw*, SolveNAR(δw*, j, t)};
```

Note that Algorithm 8 involves solving a significant amount of linear systems, both to compute $\bar{\mathbf{x}}$ and to verify its optimality (see (5.18) and (5.21)). In what follows we propose some implementation details which improve the performance of Algorithm 8.

Remark 15. *Let* $\mathbf{Ax} = \mathbf{d}$ *be a tridiagonal linear system with* $\mathbf{A} \in \mathbb{R}^{m \times m}$ *such that each equation can be written as*

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i, \quad i = 1, \ldots, m,$$

with $a_1 = c_m = 0$. Then, it can be solved with $O(m)$ operations through so called Thomas algorithm [56] (see Algorithm 9).

Algorithm 9: Thomas algorithm

input : a, **b**, **c**, **d**

output: $\bar{\mathbf{x}}$

1 Let m be the dimension of **d**

/* Forward phase */

2 **for** $i = 2, \dots, m$ **do**

3 Set $\delta_i = \frac{a_i}{b_{i-1}}$;

4 Set $b_i = b_i - \delta_i c_{i-1}$;

5 Set $d_i = d_i - \delta_i d_{i-1}$;

6 Set $\alpha_i = \frac{d_i}{b_i}$;

7 Set $\beta_i = \frac{c_i}{b_i}$;

/* Backward phase */

8 Set $\bar{x}_m = \alpha_m$;

9 **for** $i = m-1, \dots, 1$ **do**

10 Set $\bar{x}_i = \alpha_i - \beta_i \bar{x}_{i+1}$;

Then, each (tridiagonal) linear system (5.18) can be solved in $O(n)$ operations. Moreover, we can directly check the feasibility of $\bar{\mathbf{x}}$ during the backward phase of Thomas Algorithm (see lines 8-10 of Algorithm 9), namely we declare unfeasibility as soon as $0 \leq \bar{x}_i \leq u_i$ does not hold, without completing the backward propagation. Moreover, coefficients α_i and β_i, $i = 2, \dots, m$ do not change with j, so that the forward phase of Thomas algorithm can be performed only once at the beginning of the procedure `solveNAR` for the whole interval $\{s, \dots, t\}$. Finally, Thomas algorithm can also be employed to solve the (tridiagonal) linear system (5.21), needed to verify optimality of $\bar{\mathbf{x}}$. The following proposition states the worst-case complexity of `solveNAR(y,1,n)`.

Proposition 29. *Problem 15 can be solved with $O(n^3)$ operations by running*

the procedure `SolveNAR`$(\mathbf{y}, 1, n)$ *and by using Thomas algorithm for the solution of each linear system.*

Proof. In the worst case, at each iteration $j = s + 1$, so that each call of `SolveNAR` requires $O(n^2)$ operations, while the total number of recursive calls is $O(n)$, thus making an overall effort of $O(n^3)$ operations. □

In fact, the practical complexity of the algorithm is much better, namely $O(n^2)$.

5.3.3 Acceleration and NAR constraints

Now we discuss the problem with acceleration and NAR constraints, with upper bound vector $\mathbf{y}$, i.e.:

Problem 16.

$$\min_{\boldsymbol{\delta}\mathbf{w}\in\mathbb{R}^n} 2h \sum_{i=1}^{n-1} \frac{1}{\sqrt{w_{i+1} + \delta w_{i+1}} + \sqrt{w_i + \delta w_i}}$$

such that

$$\begin{aligned}
&\mathbf{l_B} \leq \boldsymbol{\delta}\mathbf{w} \leq \mathbf{y}, \\
&\delta w_{i+1} - \delta w_i \leq b_{A_i}, && i = 1, \ldots, n-1, \\
&\delta w_i - \delta w_{i+1} \leq b_{Di}, && i = 1, \ldots n-1, \\
&\delta w_i - \eta_i \delta w_{i-1} - \eta_i \delta w_{i+1} \leq b_{N_i}, && i = 2, \ldots, n-1.
\end{aligned}$$

We first prove the following proposition.

Proposition 30. *The optimal solution of Problem 16 is the maximum componentwise of its feasible region.*

Proof. The proof is the same as the one for Proposition 27. □

As for the case with the NAR constraints, we can solve Problem 16 by using the graph-based approach proposed in reference [55]. However, reference [22]

shows that, if we adopt a very efficient procedure to solve the case with acceleration and the case with NAR constraints separately, then it is worth splitting the full problem into two separated ones and use an iterative approach (see Algorithm 10). Indeed, the solution of one of the two problems is a valid upper bound for the other and viceversa. Thus, by alternating the solutions of the routine `solveACC` and `solveNAR` we have that the sequence of solutions converges from above to the optimal solution of the problem including acceleration and NAR constraints.

Proposition 31. *Algorithm 10 returns the optimal solution for the case including the acceleration and the NAR constraints.*

Proof. See Section 5.3 of [22]. □

Algorithm 10: Algorithm `SolveACCNAR` for the solution of Problem 16.

input : The upper bound $\mathbf{y}$ and the tolerance ε
output: The optimal descent direction $\boldsymbol{\delta}\mathbf{w}^*$ and the objective value f^*

```
δw_Acc = SolveACC(y);
while true do
    δw* = SolveNAR(δw_Acc, 1, n);
    if ‖δw* − δw_Acc‖ < ε then
        return δw*, evaluateObj(δw*)
    δw_Acc = SolveACC(δw*);
```

5.4 A descent method for the case of acceleration, PAR and NAR constraints

Unfortunately, PAR constraints (5.14) do not satisfy the assumptions requested in [55] in order to guarantee that the component-wise maximum of the feasible

region is the optimal solution of Problem 13. However, in Section 5.3 we have shown that Problem 16 can be efficiently solved by Algorithm 10. Our purpose then is to separate the acceleration and NAR constraints from the PAR constraints. This approach is a further extension of the results shown by [22].

Definition 6. *Let $\psi : \mathbb{R}^n \to \mathbb{R}$ be the objective function of Problem 13 and let $\mathcal{D}$ be the region defined by the acceleration and NAR constraints. We define the function $F : \mathbb{R}^n \to \mathbb{R}$ as follows*

$$F(\mathbf{y}) = \min \{\psi(\mathbf{x}) \,|\, \exists \mathbf{x} \in \mathcal{D}, \mathbf{x} \leq \mathbf{y}\}.$$

Namely, F is the optimal value function of Problem 16 when the upper bound is $\mathbf{y}$.

Proposition 32. *Function F is a convex function.*

Proof. Since Problem 16 is convex, then the optimal value function F is convex (see Section 5.6.1 of [42]) . □

Now, let us introduce the following problem:

Problem 17.

$$\min_{\mathbf{y} \in \mathbb{R}^n} F(\mathbf{y}) \tag{5.22}$$

such that

$$\eta_i(y_{i-1} + y_{i+1}) - y_i \leq b_{P_i}, \quad i = 2, \ldots, n-1, \tag{5.23}$$

$$\mathbf{l_B} \leq \mathbf{y} \leq \mathbf{u_B}. \tag{5.24}$$

such problem is a relaxation of Problem 13. Indeed, each feasible solution of Problem 13 is also feasible for Problem 17 and the value of F at such solution is equal to the value of the objective function of Problem 13 at the same solution. We will solve Problem 17 rather than Problem 13 to compute the new

displacement $\boldsymbol{\delta}\mathbf{w}$. More precisely, if $\mathbf{y}^*$ is the optimal solution of Problem 17, then we will set

$$\boldsymbol{\delta}\mathbf{w} = \arg \min_{\mathbf{x}\in\mathcal{D},\mathbf{x}\leq\mathbf{y}^*} \psi(\mathbf{x}).$$

Before solving Problem 17 we need to show that it is possible to perform a step along the computed direction $\boldsymbol{\delta}\mathbf{w}$ without exiting the feasible region Ω, provided that the step-length is small enough. That was observed to hold true for the solution of Problem 13 (see Remark 11). Since we solve a relaxation of Problem 13, we need to prove that this still holds true under mild conditions. We first need to prove a proposition.

Proposition 33. *Let* $\mathbf{w}^{(k)}$ *be the current point and let*

$$\mathcal{A}(\mathbf{w}^{(k)}) = \left\{ j : (w_{j-1}^{(k)} - 2w_j^{(k)} + w_{j+1}^{(k)})\sqrt{\frac{w_{j+1}^{(k)} + w_{j-1}^{(k)}}{2}} = 2h^2 J \right\},$$

be the set of PAR active constraints at $\mathbf{w}^{(k)}$*. Then, if*

$$\delta w_{j-1} + \delta w_{j+1} \leq 2\left(w_{j-1}^{(k)} + w_{j+1}^{(k)}\right), \quad j \in \mathcal{A}(\mathbf{w}^{(k)}), \tag{5.25}$$

then,

$$\eta_j \delta w_{j-1} + \eta_j \delta w_{j+1} - \delta w_j \leq b_{P_j} \quad j \in \mathcal{A}(\mathbf{w}^{(k)}), \tag{5.26}$$

i.e., the linearized PAR constraints is satisfied.

Proof. Let $j \in \mathcal{A}(\mathbf{w}^{(k)})$. Since $\boldsymbol{\delta}\mathbf{w}$ satisfies the acceleration and NAR constraints, we have that

$$\delta w_j \leq \delta w_{j+1} + b_{D_j}$$

$$\delta w_j \leq \delta w_{j-1} + b_{A_j}$$

$$\delta w_j \leq \eta_j(\delta w_{j+1} + \delta w_{j-1}) + b_{N_j}$$

$$\delta w_j \leq y_j^*.$$

At least one of these constraints must be active, otherwise δw_j could be increased, thus contradicting optimality. If the active constraint is $\delta w_j \leq$

$\eta_j(\delta w_{j+1} + \delta w_{j-1}) + b_{N_j}$, then constraint (5.26) is obviously satisfied and is not active. If $\delta w_j = y_j^*$, then

$$\eta_j(\delta w_{j-1} + \delta w_{j+1}) \leq \eta_j(y_{j-1}^* + y_{j+1}^*) \leq y_j^* + b_{P_j} = \delta w_j + b_{P_j},$$

where the second inequality follows from the assumption that $\mathbf{y}^*$ satisfies the PAR constraints. Now, let $\delta w_j = \delta w_{j+1} + b_{D_j}$ (the case when $\delta w_j \leq \delta w_{j-1} + b_{A_j}$ is active can be dealt with in a completely analogous way). First we observe that $\delta w_j \geq \delta w_{j-1} - b_{D_{j-1}}$. Then,

$$2\delta w_j \geq \delta w_{j+1} + \delta w_{j-1} + b_{D_j} - b_{D_{j-1}}.$$

In view of the definitions of b_{D_j} and $b_{D_{j-1}}$ this can also be written as

$$2\delta w_j \geq \delta w_{j+1} + \delta w_{j-1} + w_{j+1}^{(k)} - 2w_j^{(k)} + w_{j-1}^{(k)}.$$

Since $j \in \mathcal{A}(\mathbf{w}^{(k)})$, it also holds that

$$w_{j+1}^{(k)} - 2w_j^{(k)} + w_{j-1}^{(k)} = \frac{\Delta}{\sqrt{w_{j+1}^{(k)} + w_{j-1}^{(k)}}},$$

where $\Delta = \sqrt{2}h^2 J$. Then, the inequality can be rewritten as

$$2\delta w_j \geq \delta w_{j+1} + \delta w_{j-1} + \frac{\Delta}{\sqrt{w_{j+1}^{(k)} + w_{j-1}^{(k)}}}. \tag{5.27}$$

Now, we observe that if $j \in \mathcal{A}(\mathbf{w}^{(k)})$, then

$$\eta_j = \frac{1}{2} + \frac{\Delta}{4\left(w_{j+1}^{(k)} + w_{j-1}^{(k)}\right)^{\frac{3}{2}}}, \quad b_{P_j} = 0,$$

so that, (5.26) can be rewritten as

$$2\delta w_j \geq \delta w_{j+1} + \delta w_{j-1} + \frac{\Delta}{2\left(w_{j+1}^{(k)} + w_{j-1}^{(k)}\right)^{\frac{3}{2}}}(\delta w_{j+1} + \delta w_{j-1}).$$

Now, taking into account (5.27) such inequality holds if

$$\frac{\Delta}{\sqrt{w_{j+1}^{(k)} + w_{j-1}^{(k)}}} \geq \frac{\Delta}{2\left(w_{j+1}^{(k)} + w_{j-1}^{(k)}\right)^{\frac{3}{2}}}(\delta w_{j+1} + \delta w_{j-1}),$$

or equivalently, if $2\left(w_{j+1}^{(k)} + w_{j-1}^{(k)}\right) \geq (\delta w_{j+1} + \delta w_{j-1})$, as we wanted to prove. □

Note that, assumption (5.25) is mild since we are basically requiring that no steps larger than twice as much the current values can be taken. Under such assumption, the computed direction $\boldsymbol{\delta}\mathbf{w}$ fulfils the linearized PAR constraints so that according to Remark 11 it is possible to take a positive step-length along this direction without violating the corresponding nonlinear constraints. Since this is obviously also true for inactive nonlinear constraints, even if $\boldsymbol{\delta}\mathbf{w}$ might not satisfy their linearized version, then we can conclude that a positive step-length along the computed $\boldsymbol{\delta}\mathbf{w}$ can be taken without exiting the feasible region Ω. In order to fulfil the mild assumption (5.25) one can impose restrictions on δw_{j+1} and δw_{j-1} for $j \in \mathcal{A}(\mathbf{w}^{(k)})$, like, e.g.,

$$\delta w_{j-1} \leq w_{j-1}^{(k)} + \frac{w_{j-1}^{(k)} + w_{j+1}^{(k)}}{2},$$

and a similar restriction for δw_{j+1} which guarantee that assumption (5.25) is satisfied. However, in the computational experiments we did not impose such restriction unless a positive step-length along the compute direction $\boldsymbol{\delta}\mathbf{w}$ could not be taken (which, however, never occurred in our experiments).

Now, let us turn our attention towards the solution of Problem 17. In order to solve it, we propose a descent method. We can exploit the information provided by the dual optimal solution $\boldsymbol{\nu} \in \mathbb{R}_+^n$ associated to the upper bound constraints of Problem 16. Indeed, from the sensitivity theory, we know that the dual solution is related to the gradient of the optimal value function F (see Definition 6) and provides information about how it changes its value for small

perturbations of the upper bound values (for further details see Sections 5.6.2 and 5.6.5 in [42]). Let $\mathbf{y}^{(t)}$ be a feasible solution of Problem 17 and $\boldsymbol{\nu} \in \mathbb{R}^n_+$ be the Lagrange multipliers of the upper bound constraints of Problem 16, when the upper bound is $\mathbf{y}^{(t)}$. Let:

$$\beta_i = b_{\mathrm{P}_i} - \eta_i \left(y^{(t)}_{i-1} + y^{(t)}_{i+1} \right) + y^{(t)}_i, \quad i = 2, \ldots, n-1.$$

Then, a *feasible descent direction* $\mathbf{d}^{(t)}$ can be obtained by solving the following problem:

Problem 18.

$$\min_{\mathbf{d} \in \mathbb{R}^n} -\boldsymbol{\nu}^T \mathbf{d} \tag{5.28}$$

such that

$$\eta_i \left(d_{i-1} + d_{i+1} \right) - d_i \leq \beta_i,\ i = 2, \ldots, n-1, \tag{5.29}$$

$$\mathbf{l_B} \leq \mathbf{y}^{(t)} + \mathbf{d} \leq \mathbf{u_B}, \tag{5.30}$$

where the objective function (5.28) imposes that $\mathbf{d}^{(t)}$ is a descent direction while constraints (5.29) and (5.30) guarantee feasibility with respect to Problem 17. Problem 18 is a linear problem so it can be easily solved by using a standard linear solver (see [50]). Unfortunately, since the information provided by the dual optimal solution $\boldsymbol{\nu}$ is local and related to small perturbations of the upper bounds, it might happen that $F(\mathbf{y}^{(t)} + \mathbf{d}^{(t)}) \geq F(\mathbf{y}^{(t)})$. To overcome this issue we amend Problem 18 with a trust-region constraint. So, let $\sigma^{(t)} \in \mathbb{R}_+$ be the radius of the trust-region at iteration t, then we have:

Problem 19.

$$\min_{\mathbf{d} \in \mathbb{R}^n} -\boldsymbol{\nu}^T \mathbf{d} \tag{5.31}$$

such that

$$\eta_i \left(d_{i-1} + d_{i+1} \right) - d_i \leq \beta_i,\ i = 2, \ldots, n-1, \tag{5.32}$$

$$\bar{\mathbf{l}}_\mathbf{B} \leq \mathbf{d} \leq \bar{\mathbf{u}}_\mathbf{B}. \tag{5.33}$$

where $\bar{l}_{B_i} = \min\{l_{B_i} - y_i^{(t)}, -\sigma^{(t)}\}$ and $\bar{u}_{B_i} = \max\{u_{B_i} - y_i^{(t)}, \sigma^{(t)}\}$ for $i = 1, \ldots, n$. After each iteration of the descent algorithm, we change the the radius $\sigma^{(t)}$ according to the following rules:

- if $F(\mathbf{y}^{(t)} + \mathbf{d}^{(t)}) \geq F(\mathbf{y}^{(t)})$, then we tight the trust-region by decreasing $\sigma^{(t)}$ by a factor $\tau \in (0, 1)$ and solve again Problem 19;
- if $F(\mathbf{y}^{(t)} + \mathbf{d}^{(t)}) < F(\mathbf{y}^{(t)})$, then we make the update $\mathbf{y}^{(t+1)} = \mathbf{y}^{(t)} + \mathbf{d}^{(t)}$ and enlarge the radius $\sigma^{(t)}$ by a factor $\rho > 1$.

The proposed descent algorithm is sketched in Figure 5.3, which reports the flow chart of the procedure `ComputeUpdate` used in Algorithm 6. We ini-

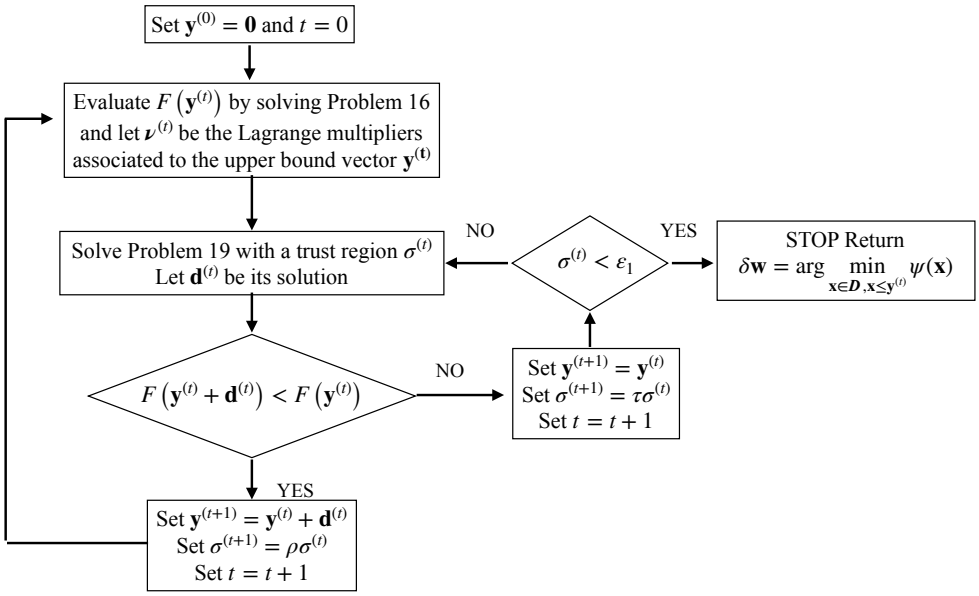


Figure 5.3: Flow chart of the routine `ComputeUpdate`

tially set $\mathbf{y}^{(0)} = \mathbf{0}$. At each iteration t we evaluate the objective function $F(\mathbf{y}^t)$ by solving Problem 16 and upper bound vector $\mathbf{y}^{(t)}$ through a call of the routine `solveACCNAR` (Algorithm 10). Then, we compute the Lagrange multipliers $\boldsymbol{\nu}^{(t)}$ associated to the upper bound constraints. After that, we compute

a candidate descent direction $\mathbf{d}^{(t)}$ by solving Problem 19. If $\mathbf{d}^{(t)}$is a descent direction, then we set $\mathbf{y}^{(t+1)} = \mathbf{y}^{(t)} + \mathbf{d}^{(t)}$ and enlarge the radius of the trust region, otherwise we tight the trust region and solve again Problem 19. The descent algorithm stops as the radius of the trust region is smaller then a fixed tolerance ε_1.

Remark 16. *Note that we initially set* $\mathbf{y}^{(0)} = \mathbf{0}$. *But any feasible solution of Problem 19 does the job and, actually, starting with a good initial solution may enhance the performance of the algorithm.*

5.4.1 A heuristic procedure for computing a suboptimal descent direction

In this section we propose a heuristic procedure for the detection of a suboptimal solution to Problem 19. Indeed, we observe that: i) an optimal descent direction is not strictly required; ii) a heuristic approach allows to reduce the time needed to get a descent direction. We first need to introduce a definition.

Definition 7. *Given a vector* $\mathbf{d} \in \mathbb{R}^N$ *the set of critical point associated to such vector is*

$$Q(\mathbf{d}) = \{p \ : \ \eta_i(d_{p-1} + d_{p+1}) - d_p - \beta_p > 0\},$$

i.e., the set of points where constraints (5.32) are violated at $\mathbf{d}$.

Now, the heuristic is detailed in Algorithm 11. Its purpose is to sequentially remove all the critical points p of the upper bound $\bar{\mathbf{u}}_\mathbf{B}$ by activating a sequence of constraints (5.32) in the neighbourhood of p itself. We initially set $\mathbf{d} = \bar{\mathbf{u}}_\mathbf{B}$ and compute the related set $Q(\bar{\mathbf{u}}_\mathbf{B})$ of critical points. After that, we consider the most violated critical point $p \in Q(\bar{\mathbf{u}}_\mathbf{B})$ and define Δ_p as its associated violation. Then, we define the *propagation function* from p. To this aim we

first define a function $\mathbf{y} : [0,1] \to \mathbb{R}^n$ such that:

$$y_j(\alpha;\mathbf{d},p) = \begin{cases} d_p & j = p \\ d_{p-1} - \alpha\Delta_p & j = p-1 \\ d_{p+1} - (1-\alpha)\Delta_p & j = p+1 \\ \eta_j^{-1}(\beta_j + y_j(\alpha;\mathbf{d},p)) - y_{j+1}(\alpha;\mathbf{d},p), & j = p-1,\dots,1 \\ \eta_k^{-1}(\beta_j + y_j(\alpha;\mathbf{d},p)) - y_{j-1}(\alpha;\mathbf{d},p), & j = p+1,\dots,n. \end{cases} \tag{5.34}$$

Then, let

$$k_1 = \max\{k < p \ : \ y_k(\alpha;\mathbf{d},p) \geq d_k\}$$

$$k_2 = \min\{k > p \ : \ y_k(\alpha;\mathbf{d},p) \geq d_k\}.$$

We define the propagation function around p as follows: $\mathbf{x}(\cdot;\mathbf{d},p) : [0,1] \to \mathbb{R}^n$ as follows:

$$x_j(\alpha;\mathbf{d};p) = \begin{cases} y_j(\alpha;\mathbf{d},p) & j = k_1+1,\dots,k_2-1 \\ d_j & \text{otherwise.} \end{cases} \tag{5.35}$$

Basically, $\mathbf{x}$ decreases the components of the current vector $\mathbf{d}$ around the critical point p in order to remove locally the violations, without decreasing all other components. The choice of not decreasing the remaining components comes from the fact that the objective function (5.31) is monotonic non-increasing. After having defined the propagation function from a *critical point* p, we search for the best α that, in addition to activating a sequence of constraints (5.32) around p, gives the best possible solution with respect to the objective function (5.31). Then, we consider the following problem:

$$\alpha^* \in \arg\min_{\alpha\in[0,1]} \left\{-\boldsymbol{\nu}^T\mathbf{x}(\alpha), |, \mathbf{x}(\alpha) \geq \bar{\mathbf{l}}_{\mathbf{B}}\right\}. \tag{5.36}$$

We employ the ternary search algorithm to efficiently find its optimal solution [57]. Note that the ternary search needs to consider the lower bound constraints. Actually, this issue can be easily overcome by setting to $+\infty$ the value of the objective function if $\mathbf{x}(\alpha) \not\geq \bar{\mathbf{l}}_{\mathbf{B}}$. If an optimal solution α^* exists,

we set $\mathbf{d} = \mathbf{x}(\alpha^*)$, compute its set of critical points $Q(\mathbf{d})$, and repeat the above procedure until Q is empty. If an optimal solution α^* does not exist, i.e., problem (5.36) is unfeasible, then we remove the critical point p from Q and repeat the above procedure by considering the next most violated critical point.

Remark 17. *The procedure described in this section is a heuristic since we do not have any proof of correctness and optimality. Moreover, it may happen that:*

- *α^* does not exist for all critical points contained in the set Q, i.e., we are unable to remove all violations;*
- *the solution returned by the heuristic might not a descent direction, i.e., $-\boldsymbol{\nu}^T\mathbf{d} > 0$.*

For this reason, if one of these two cases occurs, we do not consider the result computed by the heuristic and solve Problem 19 by using a linear solver.

Algorithm 11: The heuristic procedure to compute a descent direction $\mathbf{d}$ for Problem 19

```
1  Set d = ū_B;
2  Let U = Q(d);
3  while U ≠ ∅ do
4      Let p be the most violated critial point in U;
5      Let x(α; d; p) be defined as in (5.35);
6      if ∃ α* ∈ arg min_{α∈[0,1]} {−ν^T x(α; d, p) | x(α; d, p) ≥ l̄_B} then
7          Set d = x(α*; d, p);
8          Let U = Q(d)
9      else
10         U = U\{p}
11 return d
```

5.5 Computational Experiments

In this section we present various computational experiments performed in order to evaluate the approaches proposed in Sections 5.3 and 5.4.

In the first experiment, we show that the routine `solveACCNAR` (Algorithm 10) proposed in Section 5.3 to solve Problem 16, is significantly faster than the graph-based algorithm proposed by reference [55].

After that, we compare the computational time of IPOPT, a general purpose nonlinear solver [58], with that of Algorithm 6 over some randomly generated instances of Problem 12. In particular, we tested two different versions of Algorithm 6. The first version, called *ALGO6-H* in what follows, employs the heuristic described in Section 5.4.1. As we claimed in Remark 17 the heuristic procedure may fail and in such case we also need a linear solver. In particular, in our experiments, we used GUROBI whenever the heuristic did not produce either a feasible solution to Problem 19 or a descent direction. In the second version, called *ALGO6-G* in what follows, we always employed GUROBI to solve Problem 19.

As a final experiment we compare the performance of the two implemented versions of Algorithm 6 applied to two specific paths and see their behaviour as the number n of discretized points increases.

All tests have been performed on a Macbook Pro equipped with 2.3 GHz Intel Core i5. Both for IPOPT and Algorithm 6 the null solution was chosen as starting point. The parameters used within Algorithm 6 were $\varepsilon = 1e^{-8}$, $\varepsilon_1 = 1e^{-11}$ (tolerance parameters), $\rho = 1.10$ and $\tau = 0.25$ (trust-region update parameters). We remark that Algorithm 6 has been implemented in MATLAB, so we expect better performance after a C/C++ implementation.

Experiment 1 As a first experiment we compared the computational time of routine `solveACCNAR` (Algorithm 10) with the graph-based approach proposed in [55] for solving Problem 16. Note that the graph-based approach is a general purpose one for a wide class of problems, while routine `solveACCNAR`

is tailored to the problem with acceleration and NAR constraints. We considered the same path proposed in Example 1 of Chapter 2 with the same constraints on the longitudinal and normal acceleration and speed. Moreover, we set $J = 2$ ms^{-3} and the interpolation condition to zero, i.e., $u_1 = 0$ and $u_n = 0$. These algorithms have been implemented in C++. The results are presented in Figure 5.4 for values of n ranging from 100 to 1000. The routine `solveACCNAR` outperforms the graph-based approach proposed by [55] in solving the problem with acceleration and NAR constraints. Note that Figure 5.4 reports only the results obtained using one of the policies, namely FIFO, for the graph-based methods. We chose this policy since in [55] it was experimentally proved to be the most efficient.

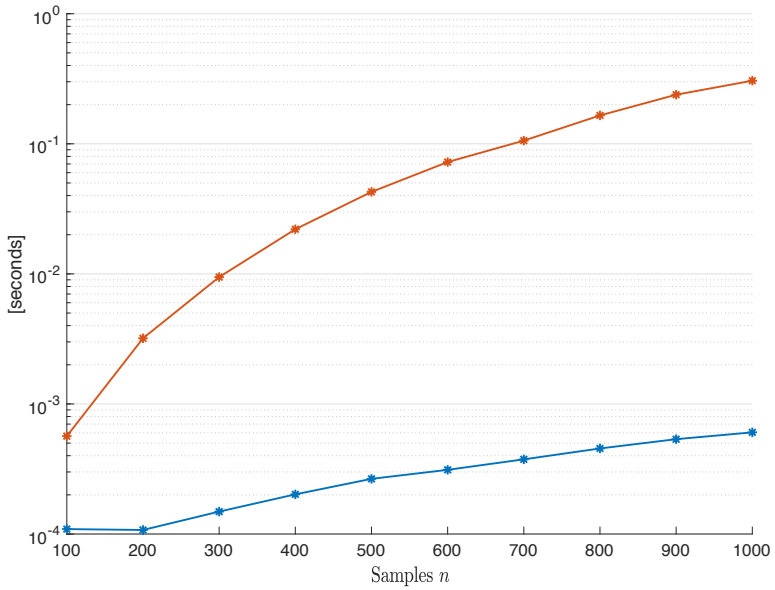
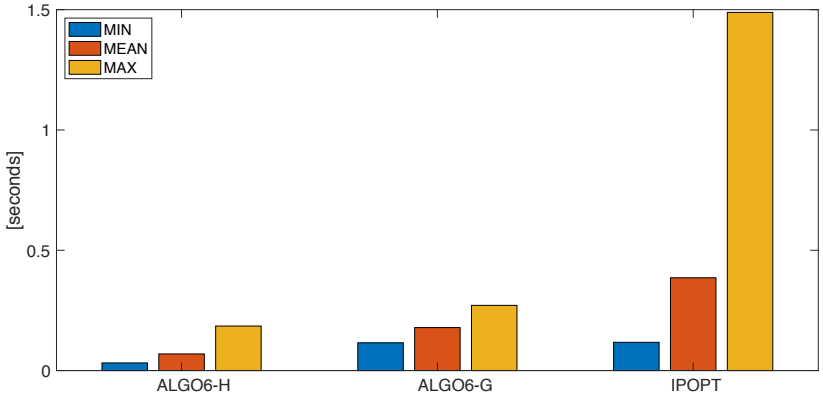


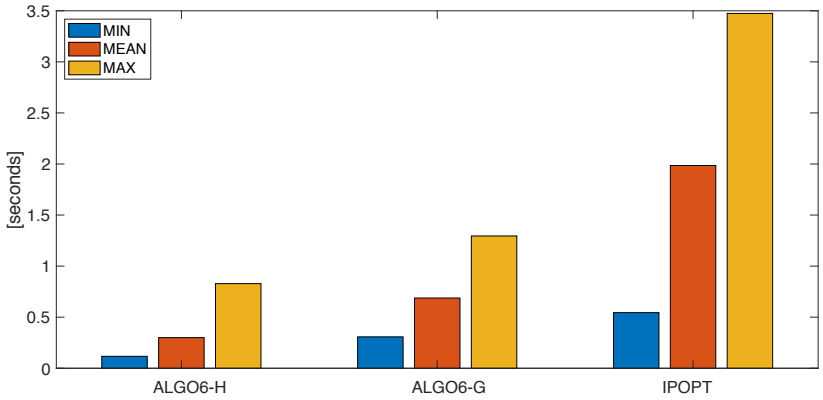
Figure 5.4: Computational times, as a function of sample points n, using `solveACCNAR` (blue line) and the graph-based approach [55] (red line).

Experiment 2 As a second experiment we compared the performance of Algorithm 6 with the nonlinear solver IPOPT. We made the experiments over a set of 50 different paths, each of which was discretized setting $n = 100$, $n = 500$ and $n = 1000$. The instances were generated by assuming that the traversed path was divided into 7 intervals over which the curvature of the path was assumed to be constant. Thus, the n-dimensional upper bound vector $\mathbf{u}$ was generated as follows. First, we fixed $u_1 = u_n = 0$, i.e., the initial and final speed must be equal to 0. Next, we partitioned the set $\{2, \ldots, n-1\}$ into 7 subintervals $I_j,\ j \in \{1, \ldots, 7\}$, which correspond to intervals with constant curvature. Then, for each subinterval we randomly generated a value $u_j \in (0, \tilde{u}]$, where $\tilde{u}$ is the maximum upper bound (which was set equal to 100). Finally, for each $j \in \{1, ..., 7\}$ we set $u_k = \tilde{u_j}\ \forall k \in I_j$. The parameter A was set to 2.78 ms^{-2} and J to 1 ms^{-3}, while the path length was $s_f = 60$ m. The results are reported in Figure 5.5 in which we show the minimum, maximum and average computational times. The results show that Algorithm 6 is significally faster than IPOPT. Moreover, it results that for the case addressed, *ALGO6-H* is faster than *ALGO6-G*. For what concerns the objective function values returned by *ALGO6-H* and *ALGO6-G*, we noted that they differed by at most $\pm 0.1\%$. Moreover, we also noted that the values returned by IPOPT were generally worse than those obtained by Algorithm 6. In particular, we observed that IPOPT hardly ever converged to a local solution (it did not return its optimality flag) and returned a feasible solution with a worse objective function value, with, in some cases, a 1% difference with respect to the results returned by Algorithm 6. Thus, it seems that Algorithm 6 is more robust than IPOPT.

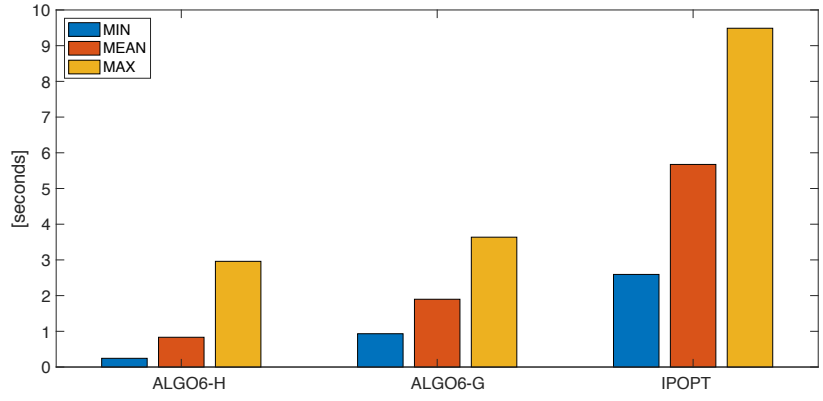
Experiment 3 We compared again the performance of Algorithm 6 with the nonlinear solver IPOPT over some different paths. Again, we made the experiments over a set of 50 different paths, each of which was discretized using $n = 100$, $n = 500$ and $n = 1000$ variables. These new instances were randomly generated such that the traversed path was divided into up to 5 interval over which the curvature could be zero, linear with respect to the arc-



(a) Samples $n = 100$



(b) Samples $n = 500$



(c) Samples $n = 1000$

Figure 5.5: Experiment 2. Computational results for Experiment 2.

length or constant. We chose this kind of paths since they are able to represent the curvature of a road trip (see [6]). An example of the generated curvature is shown in Figure 5.6. The total length of the paths was fixed to $s_f = 1000$ m, while parameter A was set to 2.78 ms^{-2}, J to 0.1 ms^{-3}and $A_N = 1$ ms^{-2}. The maximum speed μ along the paths was set 15 ms^{-1}. The results are

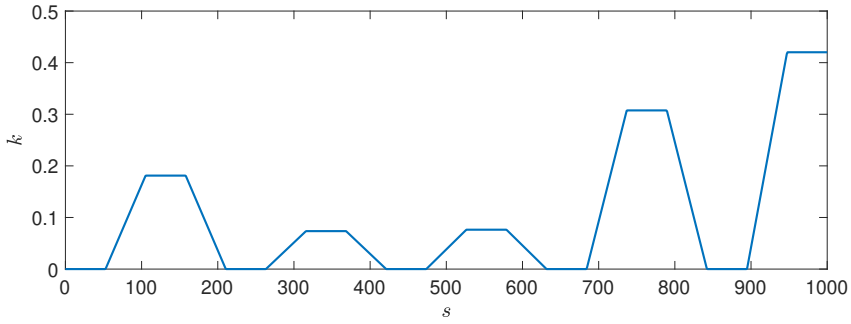
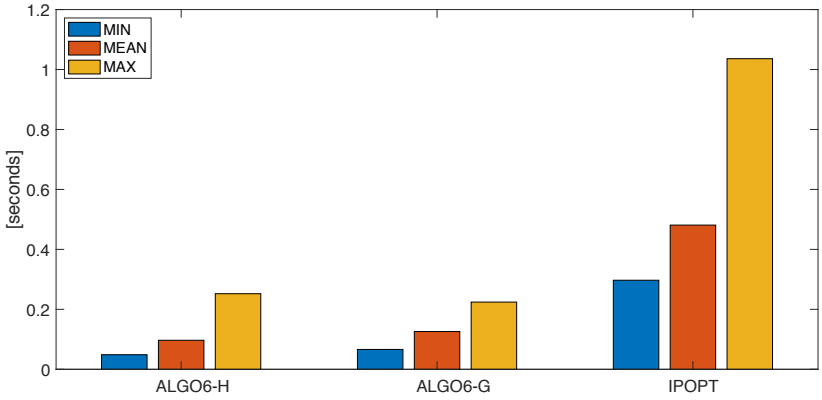


Figure 5.6: Example of a randomly generated curvature.

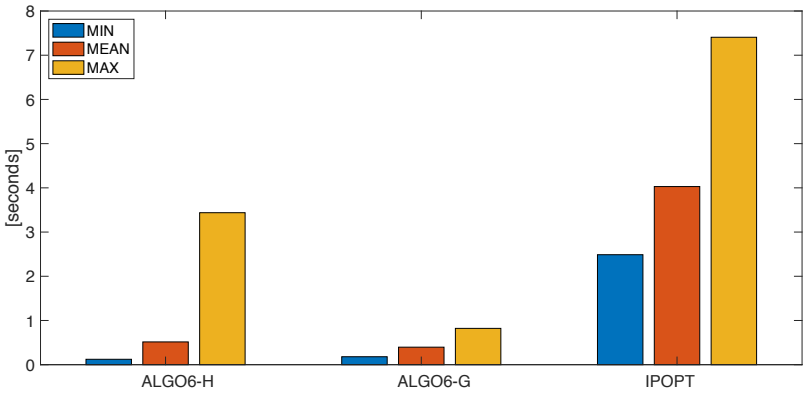
reported in Figure 5.5 in which we display the minimum, maximum and average computational times. As already shown in Experiment 2, Algorithm 6 appears faster than IPOPT. However, looking at the results shown in Figure 5.7, we note that, for this kind of paths, *ALGO6-H* appears less performant than *ALGO6-G* (see Figure 5.7b). We can give two possible motivations:

- The directions computed by the heuristic procedure are not necessarily good descent directions, so routine `computeUpdate` slowly converged to a solution.
- The heuristic procedure often fails and it was in any case necessary to call GUROBI.

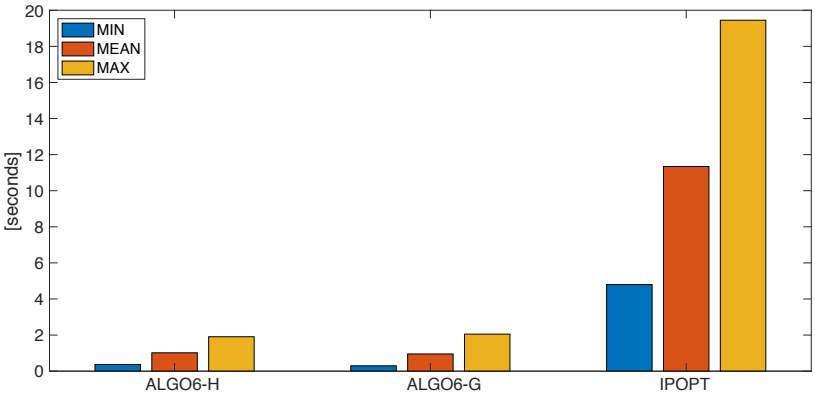
Additionally, probably for the same reasons listed above, and differently by Experiment 2, also the quality of the solutions returned by *ALGO6-H* (in terms of objective function values) is in some cases poorer with respect to *ALGO6-G*, but not as poor as those returned by IPOPT.



(a) Samples $n = 100$



(b) Samples $n = 500$



(c) Samples $n = 1000$

Figure 5.7: Computational results for Experiment 3.

Experiment 4 In our forth experiment we compared the performance of the two proposed approaches (*ALGO6-H* and *ALGO6-G*) over two possible driving scenarios as the number n of samples increases. First we considered the same path used in Experiment 1 of Chapter 2 (see Figure 2.2) to which we added jerk constraints with $J = 2$ ms^{-3}. Next, we considered a path of length $s_f = 60$ m (see Figure 5.8) whose curvature was defined according the following function

$$k(s) = \frac{1}{5}\sin\left(\frac{s}{10}\right), \quad s \in [0, s_f],$$

and parameter A, A_N and J were set to 2.78 ms^{-2}, 4.5 ms^{-2} and 0.1 ms^{-3}, respectively. The computational results are reported in Figure 5.9 and Figure 5.10 for values of n that grows from 100 to 1000. Figures 5.9 and 5.10 show

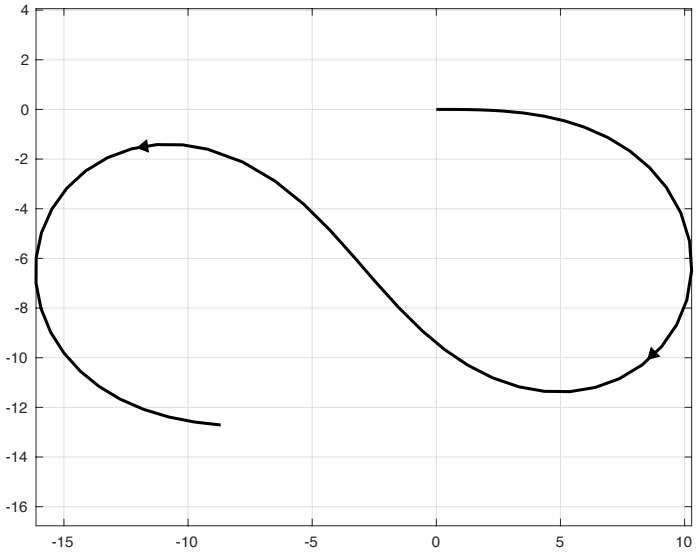


Figure 5.8: Experiment 4. Path used for the second instance.

two opposite results which confirm what we stated about Experiment 2 and 3, namely, *ALGO6-H* could be either faster or slower than *ALGO6-G*, depending on the path. In particular, it seems that the heuristic performs in a poorer way when the number of critical points tend to be large, which is the case

in the second instance. Thus, it appears that the performance of the different implementations of Algorithm 6 strictly depends on the complexity of the path considered.

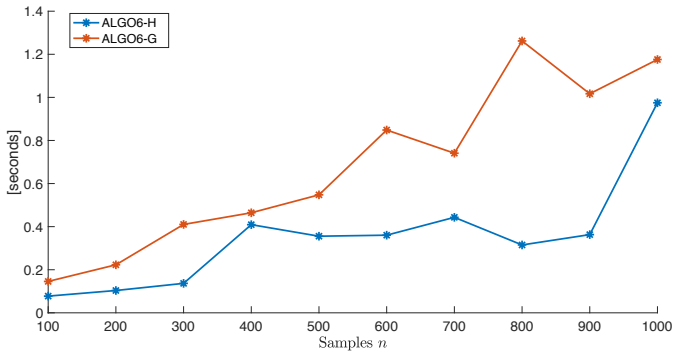


Figure 5.9: Computational times as a function of the number n of sample points for the two tested versions of Algorithm 6 over the path displayed in Figure 2.2

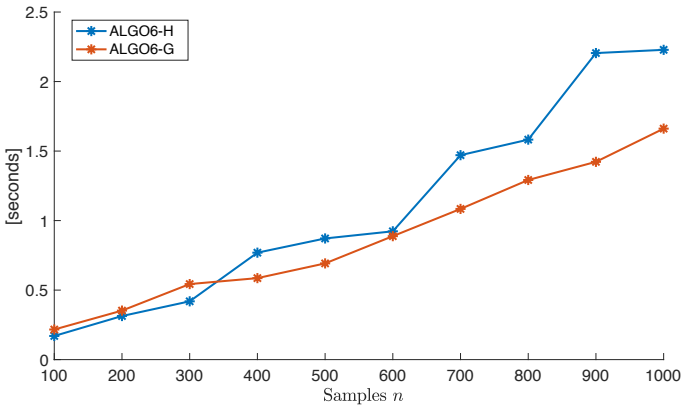


Figure 5.10: Computational times as a function of the number n of sample points for the two tested versions of Algorithm 6 over the path displayed in Figure 5.8

Finally, Figure 5.11 shows the speed profiles computed with and without

the jerk constraints for the instance associated to the path shown in Figure 2.2. It is straightforward to observe how the jerk bounded speed profile is smoother than the one obtained without the jerk limitation.

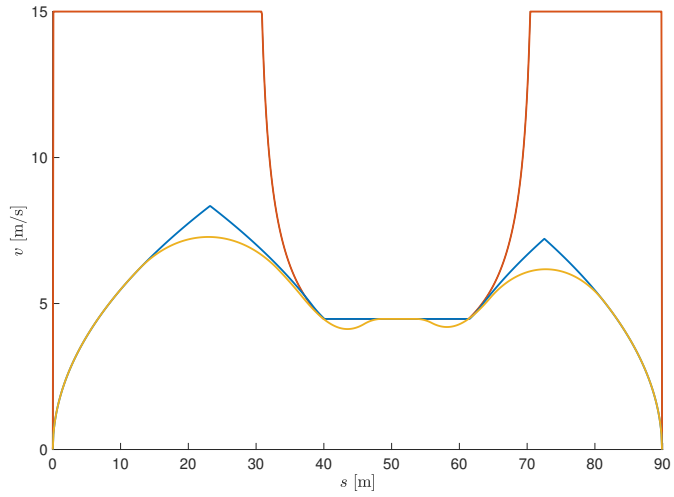


Figure 5.11: Red line represents the maximum allowed speed along the path as a function of the arc length. The blue line is the optimal speed profile without jerk constraints. The orange line is the optimal speed profile with jerk constraints.

Conclusions

This thesis dealt with the generation of the reference speed profile to be followed by a mobile robot or an industrial manipulator in order to complete an assigned task in minimum-time.
In particular, the aim of this work was to provide efficient algorithms that are able to return the optimal minimum-time speed reference profile.
This thesis proposed algorithms which exploit the structure of each finite dimensional problem formulated in order to get the optimal solution with the best possible time complexity and trying as much as possible to avoid using external solvers. Moreover, experimental tests have shown that the proposed algorithms outperform the state of the art algorithms for speed planning.
To summarize, the main contributions of this thesis were:

- In Chapter 2, for a mobile robot that has to satisfy speed and longitudinal and normal acceleration constraints, an algorithm that computes the optimal solution to the problem with linear time complexity was presented. The complexity of such algorithm was proved being optimal.

- Chapter 3 was an extension of the results presented in Chapter 2. The exact continuous-time solution of the minimum-time speed planning problem was computed without performing a finite-dimensional reduction. Moreover, Chapter 3 presented a necessary and sufficient condition for the feasibility of the minimum-time speed planning problem for a mobile robot.

- Chapter 4 addressed the speed planning problem for industrial manipulators. The proposed algorithm was proved to be optimal and it can achieve linear-time complexity $O(np)$, where n is the number of discretization points and p is the number of degrees of freedom of the robot. After that, some implementation details to speed up the proposed algorithm were provided.

- In Chapter 5 the minimum-time problem for a mobile robot was again solved, but it was updated with the addition of further smoothness constraints. The proposed algorithm is based on a standard line search method based on the sequential solution of convex problems. The main contribution lied in the procedure that was proposed to efficiently solve such problems.

Future extensions of this thesis could be:

- Add jerk constraints to the speed planning problem for industrial manipulators and apply a solution strategy similar to that proposed in Chapter 5.

- Improve the performance of the algorithm proposed in Chapter 5 trying to avoid the use of a commercial solver like GUROBI. In particular, it would be interesting to find a different heuristic procedure, with respect to the one proposed in Section 5.4.1, which could efficiently return a feasible solution to Problem 19.

- Instead of considering the travel time alone, a fuel or energy consumption term could be added to the objective function. It would be interesting to study if the new optimization problem preserves, at least partially, the special structure of the problems discussed in this thesis, or if new efficient solution approaches should be developed.

Bibliography

[1] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, New York, NY, USA, 2006.

[2] K. Kant and S.W. Zucker. Toward efficient trajectory planning: the path-velocity decomposition. *Int. J. of Robotics Research*, 5(3):72–89, 1986.

[3] Thierry Fraichard and Christian Laugier. Path-velocity decomposition revisited and applied to dynamic trajectory planning. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 40–45. IEEE, 1993.

[4] O. von Stryk and R. Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 37(1):357–373, Dec 1992.

[5] A. Piazzi, C. Guarino Lo Bianco, M. Bertozzi, A. Fascioli, and A. Broggi. Quintic G^2-splines for the iterative steering of vision-based autonomous vehicles. *IEEE Trans. on Intelligent Transportations Systems*, 3(1):27–36, March 2002.

[6] T. Fraichard and A. Scheuer. From Reeds and Shepp's to continuous-curvature paths. *IEEE Trans. on Robotics*, 20(6):1025–1035, Dec. 2004.

[7] A. Piazzi, C. Guarino Lo Bianco, and M. Romano. η^3-splines for the smooth path generation of wheeled mobile robots. *IEEE Transactions on Robotics*, 23(5):1089–1095, October 2007.

[8] C. G. L. Bianco and M. Raineri. An automatic system for the avoidance of wrist singularities in anthropomorphic manipulators. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pages 1302–1309, Aug 2017.

[9] T. Fraichard and T.M. Howard. Iterative motion planning and safety issue. In A. Eskandarian, editor, *Handbook of Intelligent Vehicles*, chapter 55, pages 1433–1458. Springer-Verlag, London, 2012.

[10] Nitin R Kapania, John Subosits, and J Christian Gerdes. A sequential two-step algorithm for fast generation of vehicle racing trajectories. *Journal of Dynamic Systems, Measurement, and Control*, 138(9):091005, 2016.

[11] F. Gravot, Y. Hirano, and S. Yoshizawa. Generation of "optimal" speed profile for motion planning. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 4071–4076, Oct 2007.

[12] G. Lini, A. Piazzi, and L. Consolini. Algebraic solution to minimum-time velocity planning. *International Journal of Control, Automation, and Systems*, 11(4):805–814, 2013.

[13] G. Lini, L. Consolini, and A. Piazzi. Minimum-time constrained velocity planning. In *Control and Automation, 2009. MED '09. 17th Mediterranean Conference on*, pages 748–753, June 2009.

[14] V. Muñoz, A. Ollero, M. Prado, and A. Simón. Mobile robot trajectory planning with dynamic and kinematic constraints. In *Proc. of the 1994 IEEE Int. Conf. on Robotics and Automation*, volume 4, pages 2802–2807, San Diego, CA, May 1994.

[15] R. Solea and U. Nunes. Trajectory planning with velocity planner for fully-automated passenger vehicles. In *IEEE Intelligent Transportation Systems Conference, ITSC '06*, pages 474 –480, September 2006.

[16] J. Villagra, V. Milanés, J. Pérez, and J. Godoy. Smooth path and speed planning for an automated public transport vehicle. *Robotics and Autonomous Systems*, 60:252–265, 2012.

[17] C. Chen, Y. He, C. Bu, J. Han, and X. Zhang. Quartic Bézier curve based trajectory generation for autonomous vehicles with curvature and velocity constraints. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6108–6113, May 2014.

[18] X. Li, Z. Sun, A. Kurt, and Q. Zhu. A sampling-based local trajectory planner for autonomous driving along a reference path. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pages 376–381, June 2014.

[19] C. Guarino Lo Bianco, A. Piazzi, and M. Romano. Velocity planning for autonomous vehicles. In *IEEE Intelligent Vehicles Symposium, IV2004*, pages 413–418, Parma, Italy, June 2004.

[20] S. Liu. An on-line reference-trajectory generator for smooth motion of impulse-controlled industrial manipulators. In *Advanced Motion Control, 2002. 7th International Workshop on*, pages 365–370, 2002.

[21] M. Brezak and I. Petrović. Time-optimal trajectory planning along predefined path for mobile robots with velocity and acceleration constraints. In *2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 942–947, July 2011.

[22] Federico Cabassi, Luca Consolini, and Marco Locatelli. Time-optimal velocity planning by a bound-tightening technique. *Computational Optimization and Applications*, 70(1):61–90, May 2018.

[23] M. Raineri, S. Perri, and C. G. L. Bianco. Online velocity planner for laser guided vehicles subject to safety constraints. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6178–6184, Sep. 2017.

[24] M. Raineri and C. Guarino Lo Bianco. Jerk limited planner for real-time applications requiring variable velocity bounds. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 1611–1617, Aug 2019.

[25] E. Velenis and P. Tsiotras. Minimum-time travel for a vehicle with acceleration limits: Theoretical analysis and receding-horizon implementation. *Journal of Optimization Theory and Applications*, 138(2):275–296, 2008.

[26] M. Frego, E. Bertolazzi, F. Biral, D. Fontanelli, and L. Palopoli. Semi-analytical minimum time solutions for a vehicle following clothoid-based trajectory subject to velocity constraints. In *2016 European Control Conference (ECC)*, pages 2221–2227, June 2016.

[27] Marco Frego, Enrico Bertolazzi, Francesco Biral, Daniele Fontanelli, and Luigi Palopoli. Semi-analytical minimum time solutions with velocity constraints for trajectory following of vehicles. *Automatica*, 86:18 – 28, 2017.

[28] J.E. Bobrow, S. Dubowsky, and J.S. Gibson. Time-optimal control of robotic manipulators along specified paths. *The International Journal of Robotics Research*, 4(3):3–17, 1985.

[29] Friedrich Pfeiffer and Rainer Johanni. A concept for manipulator trajectory planning. *IEEE Journal on Robotics and Automation*, 3(2):115–123, 1987.

[30] Kang Shin and N McKay. Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions on Automatic Control*, 30(6):531–541, 1985.

[31] J. J. E. Slotine and H. S. Yang. Improving the efficiency of time-optimal path-following algorithms. *IEEE Transactions on Robotics and Automation*, 5(1):118–124, Feb 1989.

[32] Zvi Shiller and Hsueh-Hen Lu. Computation of path constrained time optimal motions with dynamic singularities. *Journal of Dynamic Systems, Measurement, and Control*, 114(1):34–40, 1992.

[33] Quang-Cuong Pham. A general, fast, and robust implementation of the time-optimal path parameterization algorithm. *IEEE Transactions on Robotics*, 30(6):1533–1540, 2014.

[34] Quang-Cuong Pham, Stéphane Caron, and Yoshihiko Nakamura. Kinodynamic planning in the configuration space via velocity interval propagation. *Robotics: Science and System*, 2013.

[35] Quang-Cuong Pham and Olivier Stasse. Time-optimal path parameterization for redundantly actuated robots: A numerical integration approach. *IEEE/ASME Transactions on Mechatronics*, 20(6):3257–3263, 2015.

[36] Hung Pham and Quang-Cuong Pham. On the structure of the time-optimal path parameterization problem with third-order constraints. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 679–686. IEEE, 2017.

[37] Kang Shin and N McKay. A dynamic programming approach to trajectory planning of robotic manipulators. *IEEE Transactions on Automatic Control*, 31(6):491–500, 1986.

[38] S Singh and Ming-Chuan Leu. Optimal trajectory generation for robotic manipulators using dynamic programming. *Journal of Dynamic Systems, Measurement, and Control*, 109(2):88–96, 1987.

[39] M. Oberherber, H. Gattringer, and A. Müller. Successive dynamic programming and subsequent spline optimization for smooth time optimal robot path tracking. *Mechanical Sciences*, 6(2):245–254, 2015.

[40] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.

[41] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl. Time-optimal path tracking for robots: A convex optimization approach. *IEEE Transactions on Automatic Control*, 54(10):2318–2327, Oct 2009.

[42] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[43] Thomas Lipp and Stephen Boyd. Minimum-time speed optimisation over a fixed path. *International Journal of Control*, 87(6):1297–1311, 2014.

[44] Kris Hauser. Fast interpolation and time-optimization with contact. *The International Journal of Robotics Research*, 33(9):1231–1250, 2014.

[45] Ákos Nagy and István Vajk. LP-based velocity profile generation for robotic manipulators. *International Journal of Control*, 91:582–592, 2018.

[46] H. Pham and Q. Pham. A new approach to time-optimal path parameterization based on reachability analysis. *IEEE Transactions on Robotics*, 34(3):645–659, June 2018.

[47] Frederik Debrouwere, Wannes Van Loock, Goele Pipeleers, Quoc Tran Dinh, Moritz Diehl, Joris De Schutter, and Jan Swevers. Time-optimal path following for robots with convex–concave constraints using sequential convex programming. *IEEE Transactions on Robotics*, 29(6):1485–1495, 2013.

[48] Gábor Csorvási, Ákos Nagy, and István Vajk. Near time-optimal path tracking method for waiter motion problem. *IFAC-PapersOnLine*, 50(1):4929–4934, 2017.

[49] B.A. Davey and H.A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2002.

[50] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2016.

[51] F.M. Arscott and A.F. Filippov. *Differential Equations with Discontinuous Righthand Sides: Control Systems*. Mathematics and its Applications. Springer Netherlands, 2013.

[52] Nimrod Megiddo. Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems. *SIAM Journal on Computing*, 12(4):759–776, 1983.

[53] Luca Consolini, Marco Locatelli, Andrea Minari, Akos Nagy, and Istvan Vajk. A fast speed planning algorithm for robotic manipulators. *arXiv preprint arXiv:1802.03294*, 2018.

[54] Richard M Murray, Zexiang Li, and S Shankar Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[55] Luca Consolini, Mattia Laurini, and Marco Locatelli. Graph-based algorithms for the efficient solution of optimization problems involving monotone functions. *Computational Optimization and Applications*, 73(1):101–128, May 2019.

[56] Nicholas J Higham. *Accuracy and stability of numerical algorithms*, volume 80. Siam, 2002.

[57] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[58] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, Mar 2006.