A Convex Optimization Approach To Time-Optimum Path Planning

Stéphane Colas

September 6, 2009

Abstract

This project analyzes the problem of determining a good, in a time-minimizing sense, trajectory and speed profile for a car traveling along a known track. Since this problem is in general nonconvex, we investigate, and improve upon, an existing heuristic approach that simplifies the problem to the point of making it convex and allowing it to be run in an on-line scenario. We furthermore discuss applying this a proach in a receding horizon setup.

Contents

Chapter 1

Introduction

1.1 Problem Statement

We consider a car traveling along a known track. Taking into account both the dynamics of the vehicle and the track geometry, our task consists in determining a good, in a time-minimizing sense, trajectory and speed profile.
We will be looking at this problem from two perspectives: optimality and simplicity. While we would like our proposed solution to be as close to optimal as possible, it should be simple enough to be applied in a real-time receding horizon application. As is so often the case, these two objectives are competing, and we will be after a solution that reasonably trades off one against the other.

We will see that, even assuming a highly simplified car model, our most general problem formulation, whose solution we consider to be "optimal", turns out to be nonconvex. Since convexity is desirable in that it guarantees global solution optimality and generally allows for efficient problem solving, we will be considering alternative problem formulations whose solutions we know are suboptimal but allow us to solve convex problems.

1.2 Setup

1.2.1 Track & Trajectory Models

The track to which the car is constrained in its movement is modelled as follows (see fig. 1.1). We assume the track centerline to be described, in Carthesian coordinates, by a plane curve $r_0(\theta) = (x_0(\theta), y_0(\theta))^T \in \mathbf{R}^2$ parameterized in $\theta \in \mathbf{R}$.
Without loss of generality, we consider curves normalized so that $\theta \in [0, 1]$. Denoting the unit normal vector to r_0 as $n_0(\theta) \in \mathbf{R}^2$ and the track width as $w(\theta) \in \mathbf{R}$, the right and left track borders[1] are given by $r_{0,r}(\theta) = (x_{0,r}(\theta), y_{0,r}(\theta))^T = r_0(\theta) + \frac{w(\theta)}{2} n_0(\theta)$ and

[1]Without loss of generality, we assume cars to travel the track in clock-wise direction.

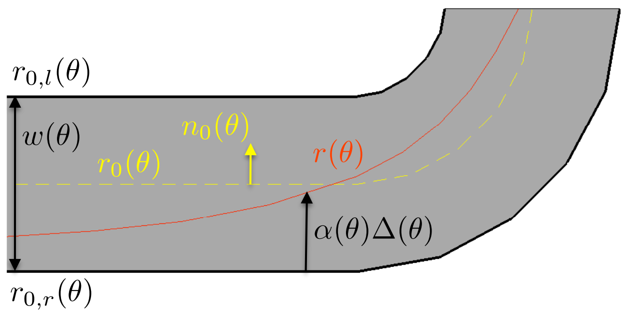


Figure 1.1: Continuous-space track and trajectory models

$r_{0,l}(\theta) = (x_{0,l}(\theta), y_{0,l}(\theta))^T = r_0(\theta) - \frac{w(\theta)}{2} n_0(\theta)$, respectively. We define the car trajectory as $r(\theta) = (x(\theta), y(\theta))^T := r_{0,r}(\theta) + \alpha(\theta)\Delta(\theta) \in \mathbf{R}^2$, where $\Delta(\theta) := r_{0,l}(\theta) - r_{0,r}(\theta) = w(\theta) n_0(\theta) \in \mathbf{R}^2$ is the vector normal to the track centerline, pointing from the right towards the left track border, of norm $w(\theta)$, and $\alpha \in [0, 1]$ measures the cars' relative distance from the right track border. Note that $\alpha(\theta)$ completely specifies the geometry of a trajectory going through the track[2]. Finally, we denote the trajectory arc length by $l(\theta) \in \mathbf{R}$ and define $l(0) = 0$ and $l(1) = L$, L denoting the total geometric length of the car trajectory.

A discretized version of this track, used for numerical calculations, is obtained by sampling θ (not necessarily uniformly) at N sampling points $\theta_i, i \in \{0, 1, \ldots, N-1\}$, which leaves us with $N-1$ track segments (1.2). Accordingly, all quantities previously param-

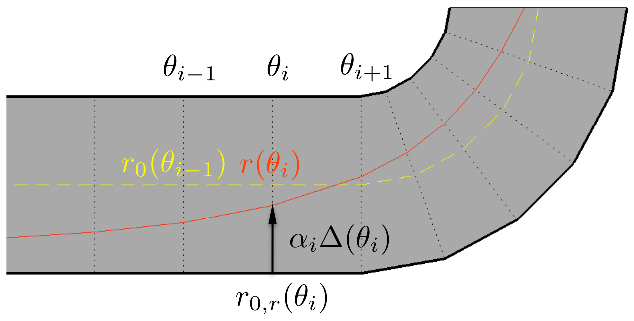


Figure 1.2: Discretized track and trajectory models

eterized in θ are now sampled at those sampling points, which is indicated by an additional sample index i. For example, we define the samples of the right track border as $(x_{r,i}, y_{r,i})^T := (x_{0,r}(\theta_i), y_{0,r}(\theta_i))^T, \forall i \in \{0, 1, \ldots, N-1\}$. These samples are collected in vectors x_r and y_r. Note that we have dropped the 0-index for better readability.
The trajectory samples are given by $(x_i, y_i)^T := (x(\theta_i), y(\theta_i))^T, \forall i \in \{0, 1, \ldots, N-1\}$ and we define $x := (x_0, x_1, \ldots, x_{N-1})^T$ and $y := (y_0, y_1, \ldots, y_{N-1})^T$ as vectors containing these samples.

[2]Equivalently, so does $r(\theta)$.

Finally, we define $\Delta_{x,i} := x_{l,i} - x_{r,i}$, $\Delta_{y,i} := y_{l,i} - y_{r,i}$ and $\Delta_i := (\Delta_{x,i}, \Delta_{y,i})^T$. Introducing the matrices $\Delta_x := \mathbf{diag}(\Delta_{x,0}, \Delta_{x,1}, \ldots, \Delta_{x,N-1})$ and $\Delta_y := \mathbf{diag}(\Delta_{y,0}, \Delta_{y,1}, \ldots, \Delta_{y,N-1})$ allows to write $x = x_r + \Delta_x\alpha$ and $y = y_r + \Delta_y\alpha$.
It is worth stressing that discretization takes place in θ, *i.e.* in the parameterization variable space, as opposed to, *e.g.*, in time t or trajectory arc length l. This choice has been made mainly for conveniency reasons.

Track Generation
All (discretized) tracks considered in this report have been generated by concatenating simple building blocks, in our case 90° curve segment and straight segment building blocks.These are easily arc length parameterized, so that uniform sampling of the parameterization variable θ leads to uniform sampling in space. The density of sampling can be chosen separately for each type of building block. Fig. 1.3 shows an example of a track generated using this

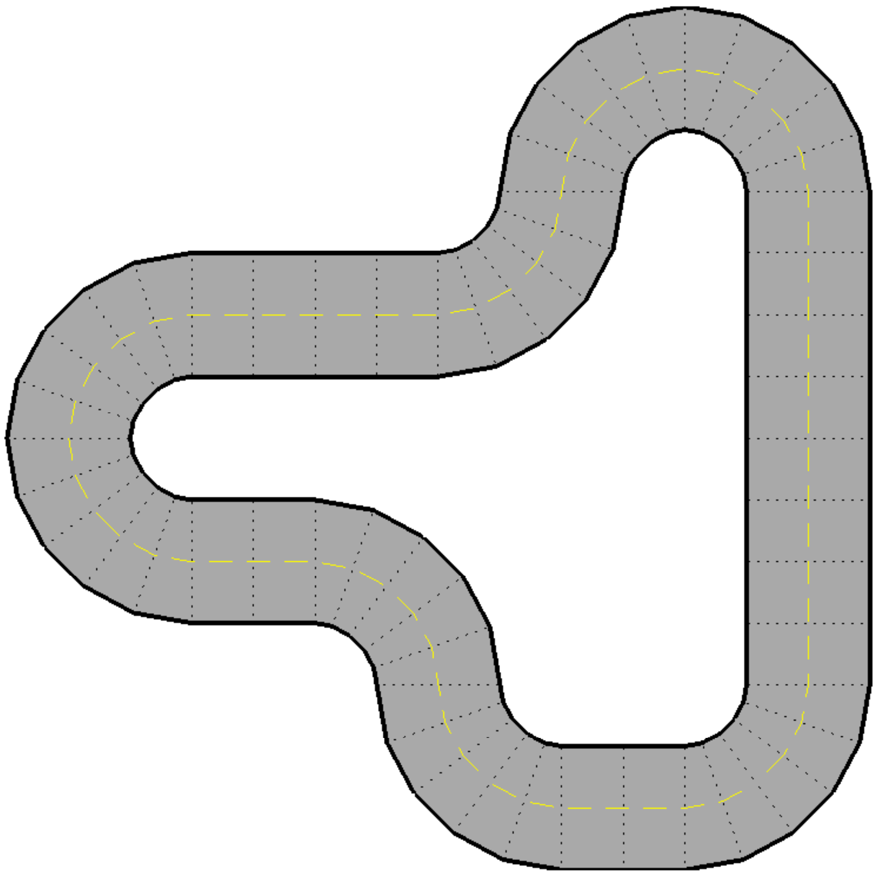


Figure 1.3: Track generation, curve sampling 5, straight sampling 2

method.

1.2.2 Car Model

We model the car as a simple point mass. As such, its movement is entirely determined by its acceleration $a(\theta) = \ddot{r}(\theta) = \frac{d^2 r(\theta)}{dt^2} \in \mathbf{R}^2$. Note that we define this acceleration to be the combined result of all forces acting on the car, both internal and external. This is an important simplifiying assumption in the sense that it hides all the complexity of transmitting forces onto the wheels, of how tires interact with the track surface *etc.*. It presumes us (the driver) to have accurate, and instant, control over the vehicle acceleration. While this choice might be deceivingly simple, it is necessary to allow for problem formulations that are anything close to convex. At the same time, this assumption can be motivated by assuming the presence of a lower-level controller that takes care of getting actual car controls right.

In order to be able to model actual cars, we need to assume constraints that take into account the main physical limitations of the vehicle:

- The vehicle speed will be upper bounded. This constraint is a consequence of a finite tire-road friction coefficient (and limited torque generation).
- The vehicle longitudinal acceleration (deceleration) will be upper (lower) bounded for similar reasons.
- The vehicle lateral acceleration will be upper bounded (in magnitude) in order to account for a maximum centripetal force developed by the tires.

After discretization, we assume constant acceleration in between sampling points, *i.e.* a linear change in velocity (and speed). We thus consider $N-1$ samples $a_i \in \mathbf{R}^2,\ i \in \{0, 1, \ldots, N-2\}$ such that $a(\theta) = a_i, \forall\, \theta \in [\theta_i, \theta_{i+1})$.

On a side note, notice that $\alpha(\theta)$, while describing the trajectory spatially, does not contain any information on the dynamics of the problem, *i.e.* how the car behaves over time. It is only in conjunction with the time dependency relation $\theta(t)$ that it contains all spatial and temporal information required to fully describe the car movement in all dimensions.

1.3 Continuous-Space Global Problem

We now formulate the problem of determining both the optimal car trajectory, $\alpha(\theta)$, and the optimal time dependency relation, $\theta(t)$, simultaneously.
Without loss of generality, we assume travelling to start at $t = 0$, with a total travel time of T, *i.e.* $\theta(0) = 0$ and $\theta(T) = 1$. The goal then is to minimize travel time, $\int_0^T dt$, subject to some constraints we will define shortly. Our objective function can be rewritten as

$$\text{min.} \quad \int_0^T dt = \int_0^1 \frac{dt}{d\theta} d\theta = \int_0^1 \frac{d\theta}{\dot{\theta}}. \tag{1.1}$$

We consider two sets of constraints:

- those forcing us to stay within the track, or *positional constraints*
- those imposed by car dynamics, or *dynamics constraints*

The positional constraints are straight forward. Since any trajectory is entirely specified by $\alpha(\theta),\ \theta \in [0,1]$, all positional constraints can be summarized as

$$0 \leq \alpha(\theta) \leq 1 \quad \forall\, \theta \in [0,1]. \tag{1.2}$$

In case we have $r(\theta)$ as a state variable in our optimization problem, we need one additonal constraint linking $\alpha(\theta)$ and $r(\theta)$,

$$r(\theta) = r_{0,r}(\theta) + \alpha(\theta)\Delta(\theta) \quad \forall\, \theta \in [0,1]. \tag{1.3}$$

In order to determine the dynamics constraints, we start by establishing relations between the car trajectory $r(\theta)$ and vehicle velocity $v(\theta) \in \mathbf{R}^2$, speed $s(\theta) \in \mathbf{R}$ and acceleration $a(\theta) \in \mathbf{R}^2$:

$$\begin{array}{l} v(\theta) = \dot{r}(\theta) = r'(\theta)\dot{\theta} \\ s(\theta) = ||v(\theta)|| = ||r'(\theta)|||\dot{\theta}| = ||r'(\theta)||\dot{\theta} \\ a(\theta) = \ddot{r}(\theta) = r''(\theta)\dot{\theta}^2 + r'(\theta)\ddot{\theta} \end{array} \tag{1.4}$$

We can furthermore split $a(\theta)$ into its longitudinal and lateral components (with respect to the trajectory):

$$\begin{array}{rcl} a(\theta) & = & a_{||}(\theta) + a_{\perp}(\theta) \\ & = & a_{\text{long}}(\theta)n_{||}(\theta) + a_{\text{lat}}(\theta)n_{\perp}(\theta), \end{array} \tag{1.5}$$

where $n_{||}$ and $n_{\perp}$ are unit vectors parallel and perpendicular[3] to the trajectory, respectively. The two acceleration components can be found as

$$\begin{array}{rcl} a_{\text{long}}(\theta) & = & \dot{s}(\theta) = \frac{r''(\theta)^T r'(\theta)}{||r'(\theta)||}\dot{\theta}^2 + ||r'(\theta)||\ddot{\theta} \\ a_{\text{lat}}(\theta) & = & \frac{||r''(\theta)\times r'(\theta)||}{||r'(\theta)||}\dot{\theta}^2 = \frac{||r''(\theta)||||r'(\theta)||| \sin(\phi)|}{||r'(\theta)||}\dot{\theta}^2 = ||r''(\theta)||| \sin(\phi)|\dot{\theta}^2, \end{array} \tag{1.6}$$

where ϕ is the angle formed by $r'(\theta)$ and $r''(\theta)$. Having formulated all the constraints, we can write the optimization problem as

$$\begin{array}{ll} \text{min.} & \int_0^1 \frac{d\theta}{\dot{\theta}} \\ \text{s.t.} & 0 \leq \alpha(\theta) \leq 1 \quad \forall\, \theta \in [0,1] \\ & r(\theta) = r_{0,r}(\theta) + \alpha(\theta)\Delta(\theta) \\ & a_{\text{long,min}}(\theta) \leq a_{\text{long}}(\theta) \leq a_{\text{long,max}}(\theta) \\ & a_{\text{lat,min}}(\theta) \leq a_{\text{lat}}(\theta) \leq a_{\text{lat,max}}(\theta) \\ & s_{\text{min}}(\theta) \leq s(\theta) \leq s_{\text{max}}(\theta), \end{array} \tag{1.7}$$

with optimization variables $r''(\theta)$, $\ddot{\theta}$, differential state variables $r'(\theta)$, $r(\theta)$, $\dot{\theta}$ and algebraic state variables $s(\theta)$, $a(\theta)$, $\alpha(\theta)$.

Using the relation

$$\dot{\theta} = \frac{d\theta}{dt} = \frac{d\theta}{dl}\frac{dl}{dt} = \frac{d\theta}{\sqrt{dx^2 + dy^2}}s(\theta) = \frac{s(\theta)}{\sqrt{x'^2 + y'^2}} = \frac{s(\theta)}{||r'(\theta)||}, \tag{1.8}$$

[3]Pointing towards the center of curvature of the path.

we can exchange the pseudo-speed $\dot{\theta}$ for the actual vehicle speed $s(\theta)$, so that we can reformulate the objective function as

$$\int_0^1 \frac{||r'(\theta)||}{s(\theta)} d\theta \tag{1.9}$$

and the acceleration components as

$$\begin{aligned} a_{\text{long}} &= \dot{s} = \tfrac{ds}{dt} = \tfrac{ds}{d\theta}\tfrac{d\theta}{dt} = \tfrac{s's}{||r'||} \\ a_{\text{lat}} &= \tfrac{||r'' \times r'||}{||r'||^3} s^2 \overset{\diamondsuit}{=} \kappa s^2 = \tfrac{s^2}{R}. \end{aligned} \tag{1.10}$$

Here κ and R denote trajectory curvature and radius of curvature, respectively and $\diamondsuit$ assumes the standard definition of curvature. Finally, we introduce $\tilde{s} := s^2$ as a variable, in which case the acceleration can be written as

$$\begin{aligned} a_{\text{long}} &= \tfrac{s's}{||r'||} = \tfrac{1}{2}\tfrac{(s^2)'}{||r'||} = \tfrac{1}{2}\tfrac{\tilde{s}'}{||r'||} \\ a_{\text{lat}} &= \tfrac{\tilde{s}}{R}, \end{aligned} \tag{1.11}$$

which eliminates the term $s's$ from the constraints. Summarizing, we can formulate our original problem as

---

Formulation 1 *Global Problem - Trajectory and Speed Profile*

$$\begin{array}{ll} \text{min.} & \int_0^1 \frac{||r'(\theta)||}{\sqrt{\tilde{s}(\theta)}} d\theta \\ \text{s.t.} & 0 \leq \alpha(\theta) \leq 1 \quad \forall\, \theta \in [0,1] \\ & r(\theta) = r_{0,r}(\theta) + \alpha(\theta)\Delta(\theta) \\ & a_{\text{long,min}}(\theta) \leq \frac{1}{2}\frac{\tilde{s}'}{||r'(\theta)||} \leq a_{\text{long,max}}(\theta) \\ & a_{\text{lat,min}}(\theta) \leq \frac{\tilde{s}}{R(\theta)} \leq a_{\text{lat,max}}(\theta) \\ & s^2_{\text{min}}(\theta) \leq \tilde{s}(\theta) \leq s^2_{\text{max}}(\theta), \end{array}$$

with optimization variables $r''(\theta)$, $\tilde{s}(\theta)$, differential state variables $r'(\theta)$, $r(\theta)$, and algebraic state variable $\alpha(\theta)$.

Note that, in the course of this project, we have been treating the dynamics constraint bounds as constants, *i.e.* dropping the θ dependency. One should at some point look at more elaborate ways of modeling these bounds, see section 4.1.

Formulation 1 is optimal in the sense that it does not assume any additional simplifications concerning the objective function or the constraints. Unfortunately, the formulation is not convex in its optimization variables: both the objective function and the acceleration constraints are nonconvex. The rest of this report will be devoted to examining simplified problem formulations whose solutions are suboptimal but potentially of practical interest.

1.4 Related Work

In essence, our trajectory planning problem is a time optimal control problem as it can be found in numerous other applications, *e.g.* in robot path planning REFS.
More specifically, for car path planning as we consider it, Braghin et al. [REF] propose an approach of splitting the global nonconvex problem into convex subproblems. While being heuristic, this approach turns out to be strikingly simple, to the point of making it attractive for on-line application. For that reason, a large part of this project was devoted to analyzing, and improving upon, this approach.
In fact, most approaches proposed in some way or another split a nonconvex problem into subproblems that are easier to handle. For instance, X et al. [REF] propose an analytical method to determine an optimal speed profile given a known car trajectory.
We would like to mention another approach, chosen by Quoc & Diehl [REF]. Starting with a general problem formulation very similar to ours (formulation 1), they apply *sequential convex programming* (SCP) REF to it . Essentially, the idea of this approach is to linearize any nonconvex constraints, making the entire problem convex and relatively easy to solve. This linearization step is repeated iteratively, hopefully leading to convergence to a global minimum. Due to its non-heuristic nature, we would expect this strategy to do well in terms of optimality, thus serving as a good reference to compare other methods against. Unfortunately, during this project, we only had an early-stage and somewhat unstable implementation of this method at our disposal. We therefore were not able to do a meaningful comparison with the approaches we analyzed.

Chapter 2

Solutions

This section presents two ways of approximately solving formulation 1. As mentioned before, given the nonconvexity of that initial problem, we will look at ways to simplify it and to come up with alternative formulations that are both convex and computationally tractable. For now, we only introduce those formulations, deferring their actual analysis to chapter 3.

On a side note, one straight forward method to handle our global problem would be to simply discretize it, *e.g.* using the direct transcription method, and to solve the discretized problem via *dynamic programming* (DP). The problem would be deterministic as well as finite-horizon, and simple backward induction could be used REF. While this solution would allow to quantify the suboptimality of other methods, we have not implemented it in the course of this project, mainly due to time constraints. Further more, a solution of this kind would be computationally demanding, allowing it only to be used as a reference and not in an on-line application scenario[1].

2.1 Two-Step Solution Approach

The approach we look at most extensively is based on work by Braghin et al. [REF].

2.1.1 Approach

The key idea of this approach is to separate the original problem into two convex subproblems that are solved once each, in sequential order. While problem formulation 1 optimizes over both the trajectory and acceleration profile simultaneously, we start by solving a purely geometric problem for a 'good' trajectory and in a second step, assuming the trajectory resulting from the first step, solve for the corresponding optimal acceleration profile.

[1]It turns out the combined input and state space would be of dimension 5. Assuming a (very humble) gridding with 10 grid points in each dimension, this amounts to up to 10^5 points to check in each DP iteration (of which there will be, depending on the track length and sampling, up to multiple dozens), which quickly becomes computationally unbearable.

It is worth emphasizing that car dynamics come into play exlusively at the stage of solving for the speed profile. We are effectively minimizing lap time given some trajectory found withoutconsidering car dynamics. Therefore, we expect this simple approach to most likely be suboptimal.
As an attempt to overcome this inherent limitation, we will add an additional optimization layer that links the car dynamics into the trajectory problem, using a simple iterative approach, see 2.1.3.

2.1.2 Optimum Path

We first present different ways of solving the geometric problem for determining good trajectories. It is a priori not clear what "good" means in this context: given that we are not yet considering car dynamics, we are not able to estimate lap times.
We would ultimately like to minimize travel time, which can be roughly seen as $\frac{\text{distance}}{\text{speed}}$. Intuition thus tells us that one could consider two objectives:

- minimizing trajectory length
- maximizing speed along the trajectory

It turns out these two objectives are competing. According to (1.11), the upper bound on vehicle speed is proportional to the trajectory radius of curvature, *i.e.* curvature should be low to allow for high speeds. Short trajectories tend to maximize curvature in curves, which drives down speeds locally and, assuming finite longitudinal acceleration, globally.

Shortest Length
The continuous-space formulation for the problem of minimizing trajectory length L can be formulated as

$$\begin{array}{ll} \text{min.} & \int_0^L dl = \int_0^1 \sqrt{x'(\theta)^2 + y'(\theta)^2} d\theta \\ \text{s.t.} & 0 \leq \alpha(\theta) \leq 1 \quad \forall\, \theta \in [0,1] \\ & r(\theta) = r_{0,r}(\theta) + \alpha(\theta)\Delta(\theta), \end{array} \tag{2.1}$$

with optimization vector $r(\theta) = (x(\theta), y(\theta))^T \in \mathbf{R}^2$. The only constraints in this case are the positional constraints. We discretize the problem by approximating $x'(\theta) = \frac{x(\theta+d\theta)-x(\theta)}{d\theta}$ in between two sampling points by $x'(\theta) \approx \frac{x(\theta_i+\Delta\theta_i)-x(\theta_i)}{\Delta\theta_i}$, $\theta \in [\theta_i; \theta_i + \Delta\theta_i]$, same for y', which gives us

$$\begin{array}{ll} \text{min.} & L = \sum_{i=0}^{N-2} l_i = \sum_{i=0}^{N-2} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \\ \text{s.t.} & 0 \leq \alpha_i \leq 1 \quad \forall\, i \in \{0, 1, \ldots, N-1\} \\ & r_i = r_{r,i} + \alpha_i \Delta_i, \end{array} \tag{2.2}$$

effectively approximating the length of the trajectory, L, by a finite sum of $N-1$ segment lengths l_i.
The problem in its current form is convex and can be formulated as a *second order cone program* (SOCP). In order to reduce computational complexity, we go one step further and slightly modify it to minimize the sum of *squared* euclidean distances, $\sum_{i=0}^{N-2} l_i^2$, which gives

us a convex formulation with a quadratic objective function.
Finally, we observe that

$$\begin{aligned} x_{i+1} - x_i &= x_{r,i+1} - x_{r,i} + \alpha_{i+1}\Delta_{x,i+1} + \alpha_i\Delta_{x,i} \\ &= \Delta_{r,x,i} + \alpha_{i+1}\Delta_{x,i+1} + \alpha_i\Delta_{x,i} \\ y_{i+1} - y_i &= y_{r,i+1} - y_{r,i} + \alpha_{i+1}\Delta_{y,i+1} + \alpha_i\Delta_{y,i} \\ &= \Delta_{r,y,i} + \alpha_{i+1}\Delta_{y,i+1} + \alpha_i\Delta_{y,i}, \end{aligned} \tag{2.3}$$

where we have defined $\Delta_{r,x,i} := x_{r,i+1} - x_{r,i}$, same for y. This means that the components of the trajectory difference vectors can be expressed as linear combinations of α's, from which follows that

$$\begin{aligned} \tilde{L} &= \textstyle\sum_{i=0}^{N-2} l_i^2 = \sum_{i=0}^{N-2} \bar{\alpha}_i^T H_{\tilde{L},i}\bar{\alpha}_i + B_{\tilde{L},i}\bar{\alpha}_i + \text{const.} \\ &= \textstyle\sum_{i=0}^{N-2} \bar{\alpha}^T E_i^T H_{\tilde{L},i} E_i\bar{\alpha} + B_{\tilde{L},i}E_i\bar{\alpha} + \text{const.} \\ &= \bar{\alpha}^T H_{\tilde{L}}\bar{\alpha} + B_{\tilde{L}}\bar{\alpha} + \text{const.} \end{aligned} \tag{2.4}$$

where $\bar{\alpha}_i := (\alpha_{i+1}, \alpha_i)^T$, $\bar{\alpha} := (\alpha_0, \alpha_1, \dots, \alpha_{N-1})^T$, and $H_{\tilde{L},i} \in \mathbf{R}^{2\times 2}$, $B_{\tilde{L},i} \in \mathbf{R}^{1\times 2}$, $E_i \in \mathbf{R}^{2\times N}$, $H_{\tilde{L}} \in \mathbf{R}^{N\times N}$, $B_{\tilde{L}} \in \mathbf{R}^{1\times N}$ such that

$$\begin{aligned} H_{\tilde{L},i} &= \begin{bmatrix} \Delta_{x,i+1}^2 + \Delta_{y,i+1}^2 & -\Delta_{x,i}\Delta_{x,i+1} - \Delta_{y,i}\Delta_{y,i+1} \\ -\Delta_{x,i}\Delta_{x,i+1} - \Delta_{y,i}\Delta_{y,i+1} & \Delta_{x,i}^2 + \Delta_{y,i}^2 \end{bmatrix} \\ B_{\tilde{L},i} &= 2\begin{bmatrix} \Delta_{r,x,i}\Delta_{x,i+1} + \Delta_{r,y,i}\Delta_{y,i+1} & -\Delta_{r,x,i}\Delta_{x,i} - \Delta_{r,y,i}\Delta_{y,i} \end{bmatrix} \\ E_i &= \begin{bmatrix} 0 & \cdots & 0 & 1_i & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 1_{i+1} & 0 & \cdots & 0 \end{bmatrix} \\ H_{\tilde{L}} &= \textstyle\sum_{i=0}^{N-2} E_i^T H_{\tilde{L},i} E_i \\ B_{\tilde{L}} &= \textstyle\sum_{i=0}^{N-2} B_{\tilde{L},i} E_i. \end{aligned} \tag{2.5}$$

Summarizing, the modified shortest length problem can be formulated as

Formulation 2 *Shortest "Length" Trajectory*

$$\begin{aligned} \text{min.} \quad & \tilde{L} = \bar{\alpha}^T H_{\tilde{L}}\bar{\alpha} + B_{\tilde{L}}\bar{\alpha} + \text{const.} \\ \text{s.t.} \quad & 0 \le \alpha_i \le 1 \quad \forall\, i \in \{0, 1, \dots, N-1\}, \end{aligned}$$

with optimization variables $\alpha_i,\ i \in 0, 1, \dots, N-1$.

The problem in this form is convex, more precisely a quadratic program (QP): our objective function is quadratic, with $H_{\tilde{L}}$ positive semidefinite, and all constraints are linear inequality constraints.

Least Curvature

We will later see that just minimizing the distance covered by the car is in most cases far suboptimal. As noted earlier, high speed trajectories tend to trade off shortness against smoothness. Thus, an alternative to minimizing trajectory length would be to maximize speed or, equivalently, radius of curvature.
We need to somehow quantify and minimize a measure of the total trajectory curvature. As mentioned earlier, curvature $\kappa \in \mathbf{R}$ is defined as

$$\kappa(\theta) = \frac{||r''(\theta) \times r'(\theta)||}{||r'(\theta)||^3}. \tag{2.6}$$

Our problem of minimizing the "aggregate" trajectory curvature can thus be formulated as

$$\begin{array}{ll} \text{min.} & \int_0^1 \kappa(\theta) d\theta \\ \text{s.t.} & 0 \leq \alpha(\theta) \leq 1 \quad \forall\, \theta \in [0,1] \\ & r(\theta) = r_{0,r}(\theta) + \alpha(\theta)\Delta(\theta). \end{array} \tag{2.7}$$

Discretizing, we obtain

$$\begin{array}{ll} \text{min.} & C = \sum_{i=0}^{N-2} c_i = \sum_{i=0}^{N-2} \kappa(\theta_i)\Delta\theta_i \\ \text{s.t.} & 0 \leq \alpha_i \leq 1 \quad \forall\, i \in \{0, 1, \ldots, N-1\} \\ & r_i = r_{r,i} + \alpha_i \Delta_i. \end{array} \tag{2.8}$$

The question at this point is how to measure curvature. First of all, in order to get an idea of what curvature looks like at the sampling points, we need to make assumptions on how the trajectory looks in between points, *i.e.* how to interpolate. When determining the discretized shortest length problem formulation, we assumed a piece-wise linear trajectory. However, linear interpolation is clearly not an option here since it would yield infinite curvature at the sample points. Also, according to REF, approximating curvature using differences of adjacent slopes does not lead to good results.
A very elegant and, as we will see, convenient choice of interpolating in between trajectory points are cubic splines REF. Spline interpolation is a kind of interpolation where the interpolant is a piecewise polynomial, in our case a piecewise cubic polynomial. Since we are considering plane curves, our trajectory $r\theta) = (x(\theta), y(\theta))^T$ will be represented by two splines, one for each component, both being parameterized in θ. The following properties make cubic splines especially attractive for our purposes:

- They are first and second order continuous, thus ensuring continuity in curvature all along the curve. A piece-wise linear interpolation on the other hand would have yielded infinite curvature at the sampling points while zero curvature in between points.

- Amongst all twice continuously differentiable functions $f(x)$, cubic splines, under certain boundary conditions, yield the least oscillating interpolation of a set of knots, minimizing the functional $\int |f''(x)|^2 dx$. This is desireable in the sense that, given a set of sufficiently closely spaced points to pass, a vehicle would also tend to choose the smoothest path connecting those points.

- On a more practical note, they allow computationally efficient calculation of second derivatives at their knots, the meaning of which will become clear soon.

Problem (2.8) is not convex, again due to nonconvexity of its objective function. In order to circumvent this, we make two assumptions.

The first assumption is crucial in that it will greatly simplify the problem formulation but at the same time prove to be quite significant in its effect on the quality of our solution. Looking at expression (2.6) for curvature, it seems hard to make this convex in $(x(\theta), y(\theta))^T$. However, assuming *arc length parameterization* of the trajectory, the expression simplifies to

$$\kappa(\theta) = \frac{||r''(\theta) \times r'(\theta)||}{||r'(\theta)||^3} = \frac{||r''(\theta)||||r'(\theta)||| \sin(\phi)|}{||r'(\theta)||^3} = ||r''(\theta)||. \tag{2.9}$$

This follows from the fact that any plane curve $\gamma(\theta)$, where θ parameterizes the curve and measures arc length, has unit curve velocity, *i.e.* $||\gamma'(\theta)|| = 1$, and its curve velocity and curve acceleration vectors are perpendicular, *i.e.* $\gamma'(\theta) \perp \gamma''(\theta)$. We would like to stress what it means to assume arc length parameterization of the trajectory. Since we fix the parameterization of the trajectory to coincide with the track centerline parameterization before solving the optimization problem, this assumption clearly is far-fetched. Indeed, whenever any feasible trajectory's arc length parameterization differs from our track centerline parameterization, its aggregate curvature will not be correctly measuredbut over- or underestimated. In fact, the unique trajectory where this assumption holds in a strict sense and where the curvature is precisely captured is the one whose arc length parameterization is in fact the track centerline parameterization. Consequently, this approach compares tracks based on their curvature but fails to correctly measure curvature in all but one (trivial) case. We thus expect to obtain suboptimal solutions, at least from a curvature minimization point of view. We will analyze these aspects more carefully in chapter 3.

The second assumption made is similar to what we did for the shortest-length problem. Instead of minimizing a finite sum of κ samples, we minimize the sum of *squared* samples, realizing that this will certainly lead to a distorted solution but hoping for the distortion to be negligeable. In fact, it might even be beneficial to consider squared curvature since, intuitively, we would sometimes[2] like to disproportionally penalize very large curvature values that may cause the vehicle to slow down considerably.
Combining the two assumptions, we obtain a new objective function

$$\tilde{C} = \sum_{i=0}^{N-2} \kappa^2(\theta_i)\Delta\theta_i = \sum_{i=0}^{N-2} ||r''(\theta_i)||^2 \Delta\theta_i = \sum_{i=0}^{N-2} \left([x''(\theta_i)]^2 + [y''(\theta_i)]^2\right) \Delta\theta_i. \tag{2.10}$$

As mentioned before, one advantage of using cubic splines for interpolating is the ease of sampling second derivatives at the spline knots. It turns out that the samples of $x''(\theta)$

[2]For instance for cars that are weak on (longitudinal and/or lateral) acceleration.

and $y''(\theta)$, which we denote by $z_{x,i} := x''(\theta_i)$ and $z_{y,i} := y''(\theta_i),\ i \in \{0, 1, \dots, N-1\}$, can be found by solving simple tridiagonal linear systems. The exact format of these systems depends on the boundary conditions we impose on our splines. For instance, if we choose to fix $x''(\theta_0) = 0$, $y''(\theta_0) = 0$ and $x''(\theta_{N-1}) = 0$, $y''(\theta_{N-1}) = 0$, which defines so-called *natural* cubic splines[3], we obtain the following. Defining $h_i := \theta_{i+1} - \theta_i = \Delta\theta_i$ and $b_{x,i} := \frac{1}{h_i}(x_{i+1} - x_i)$, $b_{y,i} := \frac{1}{h_i}(y_{i+1} - y_i)$, we can write the systems as

$$\tilde{H}_s \tilde{z}_x = \tilde{b}_x \qquad \tilde{H}_s \tilde{z}_y = \tilde{b}_y, \tag{2.11}$$

where

$$\begin{aligned}
\tilde{H}_s &:= \begin{pmatrix} 2(h_0+h_1) & h_1 & & & & \\ h_1 & 2(h_1+h_2) & h_2 & & & \\ & h_2 & 2(h_2+h_3) & h_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & h_{N-4} & 2(h_{N-4}+h_{N-3}) & h_{N-3} \\ & & & & h_{N-3} & 2(h_{N-3}+h_{N-2}) \end{pmatrix} \\
\tilde{z}_x &:= \begin{pmatrix} z_{x,1} \\ z_{x,2} \\ \vdots \\ z_{x,N-2} \end{pmatrix} \qquad \tilde{z}_y := \begin{pmatrix} z_{y,1} \\ z_{y,2} \\ \vdots \\ z_{y,N-2} \end{pmatrix} \\
\tilde{b}_x &:= 6\begin{pmatrix} b_{x,1}-b_{x,0} \\ b_{x,2}-b_{x,1} \\ \vdots \\ b_{x,N-2}-b_{x,N-3} \end{pmatrix} \qquad \tilde{b}_y := 6\begin{pmatrix} b_{y,1}-b_{y,0} \\ b_{y,2}-b_{y,1} \\ \vdots \\ b_{y,N-2}-b_{y,N-3} \end{pmatrix}.
\end{aligned} \tag{2.12}$$

Depending on the type of spline considered, we will augment $\tilde{H}_s$ to become $H_s \in \mathbf{R}^{N\times N}$ and $\tilde{z}_x$ and $\tilde{b}_x$ to become $z_x, b_x \in \mathbf{R}^{N\times 1}$, same for y. For instance, with natural cubic splines, we would define

$$H_s := \begin{pmatrix} 1 & 0_{1\times N-2} & 0 \\ h_0 & & 0_{N-3\times 1} \\ 0_{N-3,1} & \tilde{H}_s & h_{N-2} \\ 0 & 0_{1\times N-2} & 1 \end{pmatrix} \quad z_x := \begin{pmatrix} z_{x,0} \\ \tilde{z}_x \\ z_{x,N-1} \end{pmatrix} \quad b_x := \begin{pmatrix} 0 \\ \tilde{b}_x \\ 0 \end{pmatrix}. \tag{2.13}$$

Independently of what kind of spline we are considering, H_s will be strictly diagonally dominant and thus invertible. The second derivative sample vectors z_x and z_y can thus be found as

$$z_x = H_s^{-1} b_x \qquad z_y = H_s^{-1} b_y. \tag{2.14}$$

[3]Alternatively, one could fix the first derivatives at the end points (*clamped* spline) or, in case of a closed spline, make the first and last points be continuous in first and second derivative (*periodic* spline). All three forms have been used in this project.

Note that H_s will only depend on the parameterization[4] of the spline, which we assume to be constant and not part of the optimization problem.
Next we observe that the entries of the b_x and b_y vectors are linear combinations of x and y samples. Again considering the natural cubic spline case, we obtain

$$b_x = B_s x \qquad b_y = B_s y, \tag{2.15}$$

where

$$B_s := 6 \begin{pmatrix} 0 & 0 & 0 & & & & \\ \frac{1}{h_0} & -\left(\frac{1}{h_0}+\frac{1}{h_1}\right) & \frac{1}{h_1} & & & & \\ & \frac{1}{h_1} & -\left(\frac{1}{h_1}+\frac{1}{h_2}\right) & \frac{1}{h_1} & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \frac{1}{h_{N-3}} & -\left(\frac{1}{h_{N-3}}+\frac{1}{h_{N-2}}\right) & \frac{1}{h_{N-2}} \\ & & & 0 & 0 & 0 \end{pmatrix} \tag{2.16}$$

depends only on the spline parameterization.
We can thus write

$$z_x^T z_x + z_y^T z_y = x^T (H_s^{-1} B_s)^T (H_s^{-1} B_s) x + y^T (H_s^{-1} B_s)^T (H_s^{-1} B_s) y, \tag{2.17}$$

which almost corresponds to our objective function (2.10). To correct for the missing weighting factors $\Delta\theta_i$, we introduce the diagonal matrix

$$D_s := \begin{pmatrix} h_0 & & & & \\ & h_1 & & & \\ & & \ddots & & \\ & & & h_{N-2} & \\ & & & & 0 \end{pmatrix}^{\frac{1}{2}} \tag{2.18}$$

that allows us to write

$$\tilde{C} = x^T G_s^T G_s x + y^T G_s^T G_s y, \tag{2.19}$$

where

$$G_s := D_s H_s^{-1} B_s. \tag{2.20}$$

Writing x and y as a function of $\bar{\alpha}$, *i.e.* $x = x_r + \Delta_x \bar{\alpha}$ and $y = y_r + \Delta_y \bar{\alpha}$, we finally obtain

$$\tilde{C} = \bar{\alpha}^T H_{\tilde{C}} \bar{\alpha} + B_{\tilde{C}} \bar{\alpha} + C_{\tilde{C}}, \tag{2.21}$$

where

$$\begin{aligned} H_{\tilde{C}} &:= \Delta_x^T G_s^T G_s \Delta_x + \Delta_y^T G_s^T G_s \Delta_y \\ B_{\tilde{C}} &:= 2(x_r^T G_s^T G_s \Delta_x + y_r^T G_s^T G_s \Delta_y) \\ C_{\tilde{C}} &:= x_r^T G_s^T G_s x_r + y_r^T G_s^T G_s y_r. \end{aligned} \tag{2.22}$$

Summarizing, the modified least “curvature” problem can be formulated as

[4]And, in case of the clamped spline, on the first derivative values set at the beginning and end points.

Formulation 3 *Least "Curvature" Trajectory*

$$\begin{array}{ll}\text{min.} & \tilde{C} = \bar{\alpha}^T H_{\tilde{C}} \bar{\alpha} + B_{\tilde{C}} \bar{\alpha} + \text{const.} \\ \text{s.t.} & 0 \leq \alpha_i \leq 1 \quad \forall\, i \in \{0, 1, \ldots, N-1\},\end{array}$$

with optimization variables $\alpha_i,\ i \in 0, 1, \ldots, N-1$.

This is, as for the shortest length formulation, a QP.

We will further consider two variations on this formulation:

- We will choose to neglect the weighting of the curvature samples in (2.10). This can be motivated in two ways. First, in order to be meaningful, the weighting factors would have to represent spline arc lengths in between sampling points, which they clearly don't, given that they are kept fixed at the centerline parameterization. Related to this is that the center line parameterization is usually[5] sampled uniformly, in which case the weighting has no effect at all. Secondly, the least curvature objective is supposed to compete with the shortest length one. It thus makes sense not to have any measures of distance enter this objective.
- Instead of minimizing the (weighted) sum of squared curvature samples, we can look at minimizing the maximum squared curvature sample, *i.e.* minimizing $||\kappa^2||_\infty$ instead of $||\kappa^2||_1$.

Optimum Speed Profile

Once we have determined a trajectory, *i.e.* $\alpha(\theta)$ and $r(\theta)$ are known, we can compute the corresponding speed profile.
We denote by $\tilde{\theta} \in [0, 1]$ the arc length parameterization of our known trajectory. The continuous-space problem of minimizing lap time can then be formulated as an optimization problem of the following form:

$$\begin{array}{ll}\text{min.} & T = \int_0^T dt = \int_0^1 \frac{d\tilde{\theta}}{s(\tilde{\theta})} \\ \text{subject to} & a_{\text{long,min}}(\tilde{\theta}) \leq a_{\text{long}}(\tilde{\theta}) \leq a_{\text{long,max}}(\tilde{\theta}) \quad \forall\, \tilde{\theta} \in [0, 1] \\ & 0 \leq |a_{\text{lat}}(\tilde{\theta})| \leq a_{\text{lat,max}}(\tilde{\theta}) \\ & s_{\min}(\tilde{\theta}) \leq s(\tilde{\theta}) \leq s_{\max}(\tilde{\theta}),\end{array} \tag{2.23}$$

where

$$\begin{array}{l}|a_{\text{lat}}| = \frac{s^2}{R} \\ a_{\text{long}} = \dot{s} = \frac{ds}{dt} = \frac{ds}{d\tilde{\theta}} \frac{d\tilde{\theta}}{dt} \overset{(a)}{=} s's = \frac{1}{2}(s^2)',\end{array} \tag{2.24}$$

with optimization variable $s(\tilde{\theta})$. Note that (a) follows from the fact that $\tilde{\theta}$ is the arc length of our trajectory.

[5]Straight and curve building blocks are each sampled uniformly.

The discretized problem is obtained by sampling in space, assuming constant acceleration in between discretization points. We obtain the following formulation:

$$\begin{aligned}
\text{min.} \quad & T = \textstyle\sum_{i=0}^{N-2} \Delta t_i = \sum_{i=0}^{N-2} \frac{\Delta\tilde{\theta}_i}{\frac{s_{i+1}+s_i}{2}} \\
\text{s.t.} \quad & a_{\text{long,min},i} \leq a_{\text{long},i} \leq a_{\text{long,max},i} \quad \forall\, i \in \{0, 1, \ldots, N-2\} \\
& 0 \leq |a_{\text{lat},i}| \leq a_{\text{lat,max},i} \\
& s_{\text{min},i} \leq s_i \leq s_{\text{max},i} \quad \forall\, i \in \{0, 1, \ldots, N-1\},
\end{aligned} \tag{2.25}$$

where

$$\begin{aligned}
|a_{\text{lat},i}| &= \tfrac{s_i^2}{R_i} \\
a_{\text{long},i} &= \tfrac{\Delta s_i}{\Delta t_i} = \tfrac{\Delta s_i}{\Delta\tilde{\theta}_i}\tfrac{\Delta\tilde{\theta}_i}{\Delta t_i} = \tfrac{s_{i+1}-s_i}{\Delta\tilde{\theta}_i}\tfrac{s_{i+1}+s_i}{2} = \tfrac{(s_{i+1}^2-s_i^2)}{2\Delta\tilde{\theta}_i},
\end{aligned} \tag{2.26}$$

with optimization variables $s_i,\ i \in \{0, 1, \ldots, N-1\}$.
Introducing the new variables $\tilde{s}_i := s_i^2,\ i \in \{0, 1, \ldots, N-1\}$, we can reformulate the speed profile problem as

---

Formulation 4 *Speed Profile*

$$\begin{aligned}
\text{min.} \quad & T = \textstyle\sum_{i=0}^{N-2} \frac{2\Delta\tilde{\theta}_i}{\sqrt{\tilde{s}_{i+1}}+\sqrt{\tilde{s}_i}} \\
\text{s.t.} \quad & a_{\text{long,min},i} \leq a_{\text{long},i} \leq a_{\text{long,max},i} \quad \forall\, i \in \{0, 1, \ldots, N-2\} \\
& 0 \leq |a_{\text{lat},i}| \leq a_{\text{lat,max},i} \\
& s_{\text{min},i}^2 \leq \tilde{s}_i \leq s_{\text{max},i}^2 \quad \forall\, i \in \{0, 1, \ldots, N-1\},
\end{aligned}$$

where

$$\begin{aligned}
|a_{\text{lat},i}| &= \tfrac{\tilde{s}_i}{R_i} \\
a_{\text{long},i} &= \tfrac{\tilde{s}_{i+1}-\tilde{s}_i}{2\Delta\tilde{\theta}_i},
\end{aligned}$$

with optimization variables $\tilde{s}_i,\ i \in \{0, 1, \ldots, N-2\}$.

This problem has linear constraints only. As for the objective function, it is convex, as can be seen as follows: $\sqrt{\tilde{s}_i},\ i \in \{0, 1, \ldots N-1\}$ are concave functions in $\tilde{s}_i$. Since summing preserves concavity, $\sqrt{\tilde{s}_{i+1}} + \sqrt{\tilde{s}_i},\ i \in \{0, 1, \ldots N-2\}$ are also concave functions in their variables. Using scalar composition rules and the fact that the function

$$f(x) = \frac{1}{x} \quad x > 0 \tag{2.27}$$

is convex on its domain and that the extended-value extension function of $f(x)$,

$$\tilde{f}(x) = \begin{cases} \frac{1}{x} & x > 0 \\ \infty & x \leq 0 \end{cases}, \tag{2.28}$$

is nondecreasing on its domain, it follows that the functions $\frac{1}{\sqrt{\tilde{s}_{i+1}}+\sqrt{\tilde{s}_i}},\ i \in \{0, 1, \ldots N-2\}$ are convex. Finally, summation over i preserves convexity, which proves our objective function to be convex. Solving this problem means solving a semi-definite program (SDP).

To simplify things further, we will consider a (heuristic) variation on this formulation. Instead of minimizing lap time, one might opt to maximize the sum of squared speed samples, replacing the objective by

$$\max. \quad \sum_{i=1}^{N} s_i^2 = \sum_{i=1}^{N} \tilde{s}_i, \tag{2.29}$$

Doing so effectively turns problem formulation 4 into a linear program (LP) that can be solved very efficiently.

As a side note, one could also consider a slightly more elaborated dynamics model by introducing a *combined slippage* constraint

$$a_i^2 + \frac{a_{\text{long,max}}^2}{\left(r_i \cdot a_{\text{lat,max}}\right)^2} (\tilde{s}_i)^2 \leq a_{\text{long,max}}^2, \tag{2.30}$$

which would mean solving a quadratically constrained LP that can be formulated as an SOCP. We will however not further pursue this in the course of this project.

2.1.3 Feedback Approach

As pointed out, simply solving for the trajectory and speed profile consecutively is suboptimal since car dynamics are not included in the trajectory finding sub-problem.
We also mentioned that, in determining a good trajectory, there exists a trade-off between minimizing trajectory length and curvature. It is unclear at this point which approach is giving better results under which conditions[6] but it is reasonable to assume that, in most cases, a better solution can be obtained by combining the two objectives.
In order to address these two issues, we will consider the improved[7] problem of minimizing lap time given a trajectory taking into account car dynamics.
One way of doing this is as follows. We include the car dynamics parameters indirectly into the trajectory sub-problem by choosing its objective function to be the *time-optimal* combination between curvature and length minimization. In order words, we insert "feedback“ from the speed profile subproblem to the trajectory subproblem. This extended optimization problem consists in determining the length-curvature trade-off minimizing lap time.
In practice, we will introduce a scalar weighting factor $\epsilon \in [0, 1]$ and choose as the new trajectory objective function a linear combination of the shortest length and least curvature objective functions. Given a fixed value of ϵ, we can determine a trajectory according to the new objective function and compute the optimal speed profile using formulation 4 and consequently the resulting lap time. Unfortunately, lap time as a function of ϵ turns out to be nonconvex (see later).We will therefor resort to gridding ϵ and computing the lap time for each grid point, which allows us to choose the optimal, *i.e.* time-minimizing, ϵ from the results.

[6]Conditions are defined by both car and track characteristics.
[7]Improved compared to the approach without feedback.

2.1.4 Boundary Conditions

Up to this point, we have considered our problem without any boundary conditions on vehicle position or velocity. While this makes sense as long as we look for the globally best time-minimizing solution, a practical scenario does require boundary conditions to be taken into account:

- In an receding horizon application, the solution for any given time sample necessarily has to be based on the state of the car at the end of the preceding time slot. Enforcing constraints at the end of the horizon (*final state constraints*) could furthermore mitigate potential feasibility problems (see section 3.4.3).
- The one-shot case, *i.e.* planning a trajectory from start to finish of a race, requires conditions on the end points in order to account for transient speed-up and slow-down phases.

It is trivial to extend our optimization problems with additional constraints allowing to enforce such conditions:

- Positional constraints can be expressed by linear constraints on the α_i.
- Speed constraints become linear constraints on the $\tilde{s}_i$.
- Directional constraints can be implemented in two ways. One can choose to interpolate using clamped splines instead of natural splines, which allows to set the first derivative at the trajectory start and/or end point. Alternatively, one may assume a piece-wise linear trajectory and approximate first order derivatives by finite differences.

For our analysis, we will, except for the receding horizon application example in section 3.4, consider the case without boundary conditions[8].

2.2 Bi-Convex Approach: An Experiment

There are two aspects of the two-step solution that need further investigating. One is that nothing has been said as to how to efficiently optimize over the scalar linear combination factor ϵ. The other, more crucial aspect is that we explicitly restrict ourselves to solutions that minimize a linear combination of length and curvature objective functions. While it is not clear how restrictive this assumption actually is, it would at least seem plausible that the optimal ϵ should somehow vary along the (portion of) track considered. Of course, one could in principle introduce one linear combination factor ϵ_i per track sample, but using gridding to optimize over all those variables simultaneously quickly becomes computationally untractable.
We now introduce an approach that integrates car dynamics into the trajectory finding

[8]Except for closed loop trajectories on which we enforce closed loop constraints.

process, does not require any nonconvex optimization problems to be solved and does not directly assume any particular trajectory structure.
The main idea here is that, while the global problem formulation 1 is not convex in all of its variables simulateously, it can (under simplifying assumptions) be split up into two convex subproblems that are *each* dependent on car dynamics[9]:

- finding the speed profile given a known trajectory
- finding a trajectory given a known speed profile

The approach is iterative, each iteration consisting in solving the two problems sequentially, in the above order. Consequently, we need to initialize the procedure with a trajectory estimate. We will investigate the importance of this initializer and see that the behavior of the algorithm sensibly depends on it. More on this will be said in section 3.3.

2.2.1 Speed Profile

The speed profile problem will be exactly the same as in the two-step approach, *i.e.* formulation 4.

2.2.2 Trajectory

Ideally, our objective function will be the same as in the speed profile problem, *i.e.* total travel time. The constraints to be fulfilled will be the geometric constraints from the track and the dynamics constraints on the acceleration (note that since we assume speeds to be known, the speed constraints can be omitted).
Skipping the continuous-space formulation as well as other previously seen details, the discretized problem, where $\tilde{\theta}$ is the arc length parameterizations of our trajectory, can be written as

$$\begin{array}{ll} \text{min.} & T = \sum_{i=0}^{N-2} \Delta t_i = \sum_{i=0}^{N-2} \frac{\Delta\tilde{\theta}_i}{\frac{s_{i+1}+s_i}{2}} \\ \text{s.t.} & 0 \leq \alpha_i \leq 1 \quad \forall\, i \in \{0,1,\ldots,N-1\} \\ & r_i = r_{r,i} + \alpha_i \Delta_i \\ & a_{\text{long,min},i} \leq a_{\text{long},i} \leq a_{\text{long,max},i} \quad \forall\, i \in \{0,1,\ldots,N-2\} \\ & |a_{\text{lat},i}| \leq a_{\text{lat,max},i} \end{array} \tag{2.31}$$

where

$$\begin{array}{l} |a_{\text{lat},i}| = s_i^2 \kappa(\tilde{\theta}_i) \\ a_{\text{long},i} = \frac{(s_{i+1}^2 - s_i^2)}{2\Delta\tilde{\theta}_i}, \end{array} \tag{2.32}$$

with optimization variables $\alpha_i,\ i \in \{0,1,\ldots,N-1\}$. Note that the speeds s_i are assumed to be known.
In order to find expressions for the trajectory segment lengths $\Delta\tilde{\theta}_i$ and curvature samples κ_i, we observe that the results of the shortest length and least curvature problems come in handily:

[9]As opposed to the two-step solution that only had them enter one of the problems.

- if we approximate each trajectory segment length $\Delta\tilde{\theta}_i$ by the euclidean distance l_i between trajectory samples, we know from the shortest length problem that the squares of these distances can be written as quadratic functions of sets of α_i
- if we assume cubic spline interpolation in between trajectory points, the squared second derivative samples that determine curvature[10] can also be written as quadratic functions of sets of α_i

To obtain a simple convex objective function, we should thus minimize the sum of *squared* travel times, using euclidean distances, *i.e.*

$$\text{min.} \quad \tilde{T} = \textstyle\sum_{i=0}^{N-2} (\Delta t_i)^2 = \sum_{i=0}^{N-2} \frac{l_i^2}{\left(\frac{s_{i+1}+s_i}{2}\right)^2}. \tag{2.33}$$

The lateral acceleration constraint is easily made quadratic by squaring all terms:

$$s_i^2 \kappa_i \leq a_{\text{lat,max},i} \iff \kappa_i^2 \leq \frac{a_{\text{lat,max},i}^2}{s_i^4} \tag{2.34}$$

which holds because the left hand side is non-negative.
The longitudinal acceleration constraint on the other hand is more problematic. First, using the euclidean distance approximation, we rewrite it as

$$\begin{array}{ll} & a_{\text{long,min},i} \leq \frac{s_{i+1}^2 - s_i^2}{2l_i} \leq a_{\text{long,max},i} \\ \iff & \begin{cases} \frac{s_{i+1}^2 - s_i^2}{2l_i} \leq a_{\text{long,max},i}, & \text{if } s_{i+1} \geq s_i \\ \frac{s_i^2 - s_{i+1}^2}{2l_i} \leq -a_{\text{long,min},i}, & \text{if } s_{i+1} < s_i \end{cases} \\ \iff & \begin{cases} l_i^2 \geq \frac{(s_{i+1}^2 - s_i^2)^2}{4a_{\text{long,max},i}^2} & \text{if } s_{i+1} \geq s_i \\ l_i^2 \geq \frac{(s_{i+1}^2 - s_i^2)^2}{4a_{\text{long,min},i}^2} & \text{if } s_{i+1} < s_i \end{cases} \end{array} \tag{2.35}$$

This constraint is not convex, since a lower bound on a quadratic expression yields a non-convex set. On the other hand, dropping it is not an option whenever the longitudinal acceleration limit is relevant. In order to circumvent this problem, we linearize the constraint by linearizing the left hand side, *i.e.* the quadratic function l_i^2, as $l_{i,\text{lin}}^2$. We know that a linearized convex expression underestimates the original expression, which means in our case that the new feasible set will be smaller than the original one, *i.e.* we are being conservative and the linearized problem definitely is feasible as long as the original one is.
Summarizing, we obtain the following convex problem:

[10]In the case of arc length parameterization.

Formulation 5 *Biconvex Trajectory Generation*

$$\begin{aligned}
\text{min.}\quad & \tilde{T} = \textstyle\sum_{i=0}^{N-2} \frac{l_i^2(\alpha)}{\left(\frac{s_{i+1}+s_i}{2}\right)^2} \\
\text{s.t.}\quad & 0 \leq \alpha_i \leq 1 \quad \forall\, i \in \{0, 1, \dots, N-1\} \\
& \kappa_i^2(\alpha) \leq \frac{a^2_{\text{lat,max},i}}{s_i^4} \quad \forall\, i \in \{0, 1, \dots, N-2\} \\
& l^2_{i,\text{lin}}(\alpha) \geq \frac{(s^2_{i+1}-s_i^2)^2}{4a^2_{\text{long,max},i}} \quad \text{if } s_{i+1} \geq s_i \\
& l^2_{i,\text{lin}}(\alpha) \geq \frac{(s^2_{i+1}-s_i^2)^2}{4a^2_{\text{long,min},i}} \quad \text{if } s_{i+1} < s_i
\end{aligned}$$

with optimization variables $\alpha_i,\ i \in \{1, 2, \dots, N-1\}$.

The problem in this form is a *quadratically constrained quadratic program* (QCQP), the lateral acceleration constraint being quadratic.
Combining the two sub-problems, we tentatively propose the following solution procedure:

- Initialize: (somehow) determine a reasonable trajectory to improve upon
- Outer loop
  1. Solve the speed profile problem according to formulation 4
  2. Inner loop
     - Solve the linearized trajectory problem according to formulation 5
     - Break if $||\Delta\alpha|| <$ thresh.
  3. Break if $||\Delta\alpha|| <$ thresh.

Note that we iterate on the linearization in order to reduce the effect of underestimating l_i^2.

Chapter 3

Simulation & Analysis

This section uses numerical simulation to investigate the performance of the two approaches presented in the previous section, pointing out their respective advantages as well as flaws. We conclude by considering the challenges of using the two-step approach in a receding horizon application example.

All code has been implemented in MATLAB R2009a. Optimization problems were solved using MATLAB's *linprog* and *quadprog* routines where possible, otherwise using cvx REF, with SDPT3 REF as a solver.

3.1 Approach

Whether any of the proposed (suboptimal) methods works well or not essentially depends on two elements:

- track geometry: track width (as compared to the radius of curvature), track curviness[1]
- car characteristics: constraints on speed and acceleration

In what follows, we will thus try to discern those conditions that are most crucial to each approach and cause it to break down, *i.e.* to be excessively suboptimal.
Testing was done on a wide range of tracks and assuming different car profiles. We have considered entire closed loop tracks but also open loop portions of tracks, given that we are ultimately looking for an approach that can be applied on-line in a finite horizon setup. In what follows, we will only be presenting a subset of the tests run during this project, examples chosen to demonstrate specific aspects of the different approaches. Since it would go beyond the scope of this project to comprehensively analyze all relevant track, car and method parameter configurations, we will abstain from giving specific instructions on how to set these parameters and rather give general guidelines when approriate. Furthermore,

[1]By curviness, we somewhat hand-wavingly mean the proportion of the track consisting of curves as opposed to straight track parts.

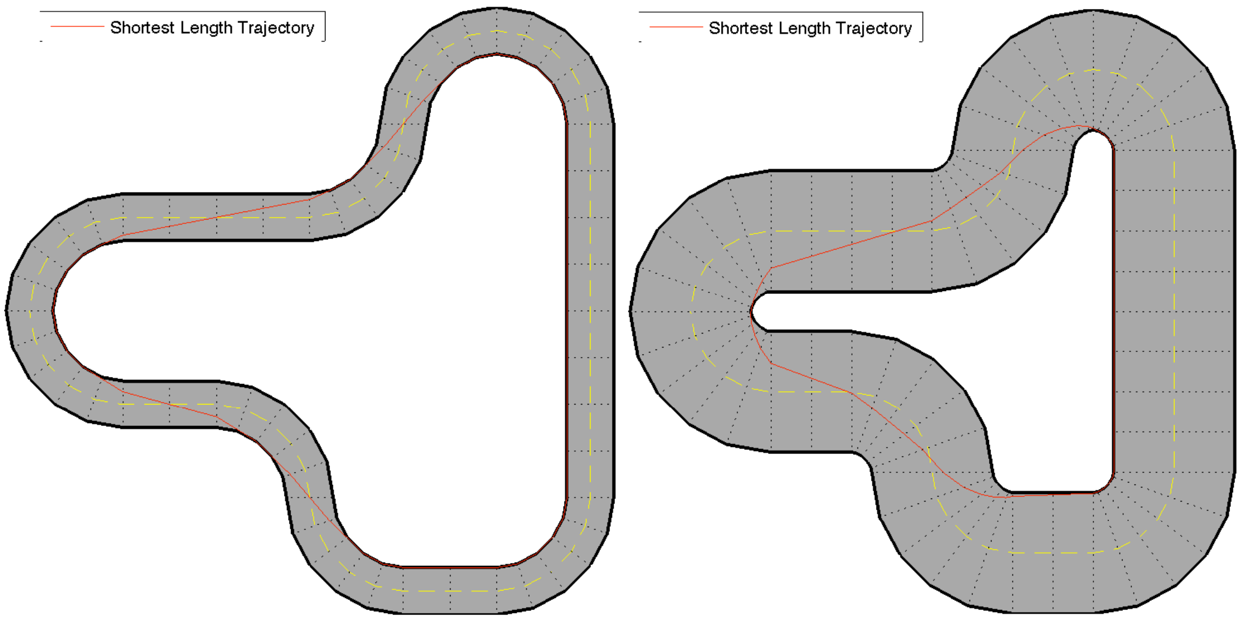
we hope that *understanding* the underlying *effects* helps in tuning and implementing the proposed methods in practice.

3.2 Two-Step Method

3.2.1 Trajectory

Shortest Length

We expect the solution to the modified problem formulation 2 to differ from that of the original problem (2.2), given that we are minimizing the sum of *squared* trajectory segment lengths.
While the effect can hardly be seen on tracks of small width, increasing the track width makes it quite apparent. Consider the example shown in figures 3.1a and 3.1b. The second

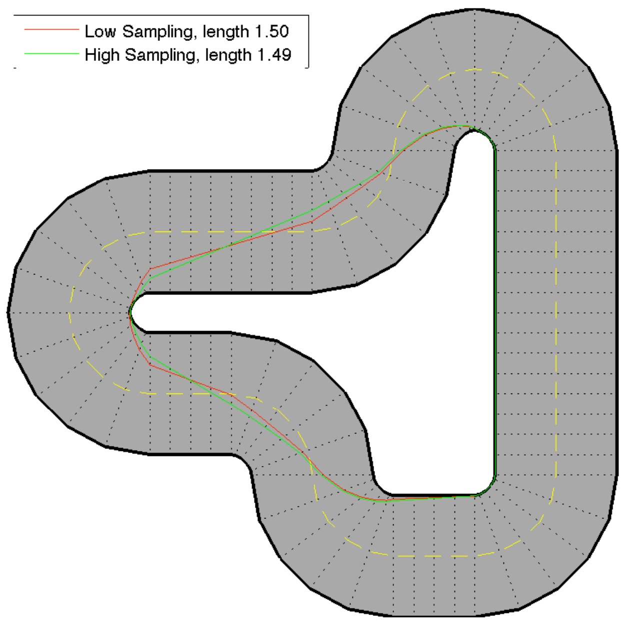


(a) Narrow track (b) Wide track

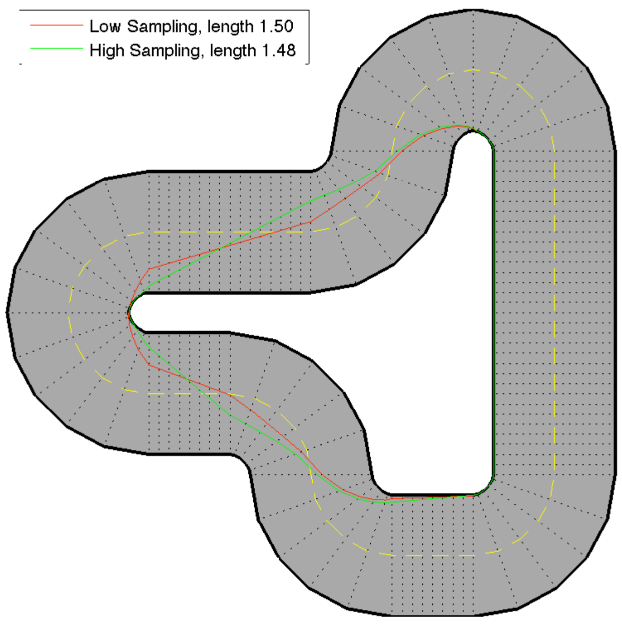
Figure 3.1: Shortest length solution for different track widths

figure shows a very obvious discrepancy between our trajectory and what we would expect the minimum length trajectory to be. This can be explained by the fact that squaring lengths penalizes longer trajectory segments disproportionately. Considering our example track, the slopes (with respect to the track center line) of the (ideally) straight trajectory parts end up being smaller (in magnitude) than they would be in an optimal solution. Compared to the actual shortest length solution, our trajectory covers more ground than it should on the curvy track parts and less ground on the straight track parts. In general, one would expect the effect to be larger the wider a track is. First, because the track would in principle allow

trajectory segments with ever larger slopes but they would most likely be avoided by our solution method. Second, because the difference between trajectory lengths on the inside and on the outside of curves grows.
We expect a finer segmentation of the straight track segments to get us closer to the solution of our original problem because the cost incurred by larger slopes is smaller. Indeed, as can be see on figures 3.2a and 3.2b, the wide track solution improves if we double and quadruple the sampling of the straight track parts. It can in fact be verified that the total trajectory



(a) 4 samples per segment



(b) 8 samples per segment

Figure 3.2: Effect of increasing straight building block sampling

length is shortened, in this case, by 0.91 and 0.94%, respectively.
Varying the curve sampling rate (as expected) also has an effect on the solution. However, as soon as both sampling rates are varied independently[2], it becomes much more difficult to discuss their respective effect on solution optimality. In the end, we have found a good choice for sampling rates to be heavily dependent on the particular track geometry considered. Additionally, the beneficial effect of increasing the sampling rate clearly follows a law of diminishing returns in the sense that the computational cost increases considerably for small gains in trajectory length. We will therefore in all that follows assume the sampling as used in figure 3.1, *i.e.* 2 samples per segment on straight blocks and 5 per segment on curve blocks, as this sampling proved to deliver reasonable results.

[2]It seems to make sense to look at the ratio of the sampling rates. A sharp increase in one usually tends to improve the behavior on that building block, causing undesired behavior on the other.

Least Curvature

We start by looking at a toy example that clearly demonstrates that our approach, as mentioned earlier, is not actually minimizing curvature as we would ideally want it to.
Consider a track consisting of a perfect circle of radius 1, width 1, centered around the origin $(0,0)$. The actual steady state least curvature solution, *i.e.* the minimizer of (2.10), would be *any* circle inside the track, centered around the origin. This can easily be seen by the fact that, assuming a circular trajectory[3] of radius of curvature R, the integrated curvature is given by $C = \sum_{i=0}^{N-2} \kappa_i \Delta\theta_i = \sum_{i=0}^{N-2} \frac{1}{R}\frac{2\pi R}{N-1} = 2\pi$, which is independent of R. In section 2.1.2, we argued that it seems reasonable to remove any length information from the curvature minimization problem. If we do so, minimizing $\hat{C} := \sum_{i=0}^{N-2} \kappa_i = \frac{N-1}{R}$, the optimal solution becomes unique, lying on the outer border of the track. If one were to exclusively consider the curvature minimization problem, it is not clear which of the two solutions leads to better lap times. The weighted solution typically is somewhat shorter, as expected. Consequently, purely speed-limited cars will in general show better performance on this solution. However, since we would in the long run like to consider linear combinations of least curvature and shortest length, this reinforces our idea that it makes more sense to only consider the unweighted sum of curvature samples. For that reason, in what follows, we restrict ourselves to the *unweighted* formulation.
Our least curvature solution, based on formulation 3, but *without* weighting, is shown in 3.3. We immediately see that this is not the solution we were looking for, *i.e.* the outer border of the track.
The seemingly erratic behavior can easily be explained looking at the two main simplifying assumptions made in deriving formulation 3:

1. in order to estimate curvature, we assumed our trajectory to be arc length parameterized by the track center line arc length parameterization θ: $\kappa_i \approx \sqrt{(x''(\theta_i))^2 + (y''(\theta_i))^2}$
2. we minimized by the sum of *squared* (weighted) curvature samples, *i.e.* squared second derivative samples: min. $\sum_{i=0}^{N-2} \kappa_i^2 = \sum_{i=0}^{N-2}((x''(\theta_i))^2 + (y''(\theta_i))^2$

In what follows, we denote by κ_i the actual trajectory curvature and by $\hat{\kappa}_i$ our estimation of it. We can now distinguish two cases for choosing the trajectory:

- Choosing the trajectory to lie precisely on the center line, *i.e.* $R = 1$, would yield $\hat{\kappa}_i = \kappa_i = \frac{1}{R} = 1$. The curvature would be correctly estimated since the arc length parameterization assumption perfectly holds on the center line. For future reference, we denote this trajectory by $r_c(\theta)$.
- Now consider what happens if we choose the trajectory to not lie on the center line, *i.e.* $R \in [0.5, 1.5] \setminus 1$, still assuming parameterization by the center line arc length parameter. The new trajectory $r(\theta)$ will, due to our choice of centering the track

[3]The fact that the optimal steady state trajectory has to be a circle can for instance be shown using symmetry arguments.

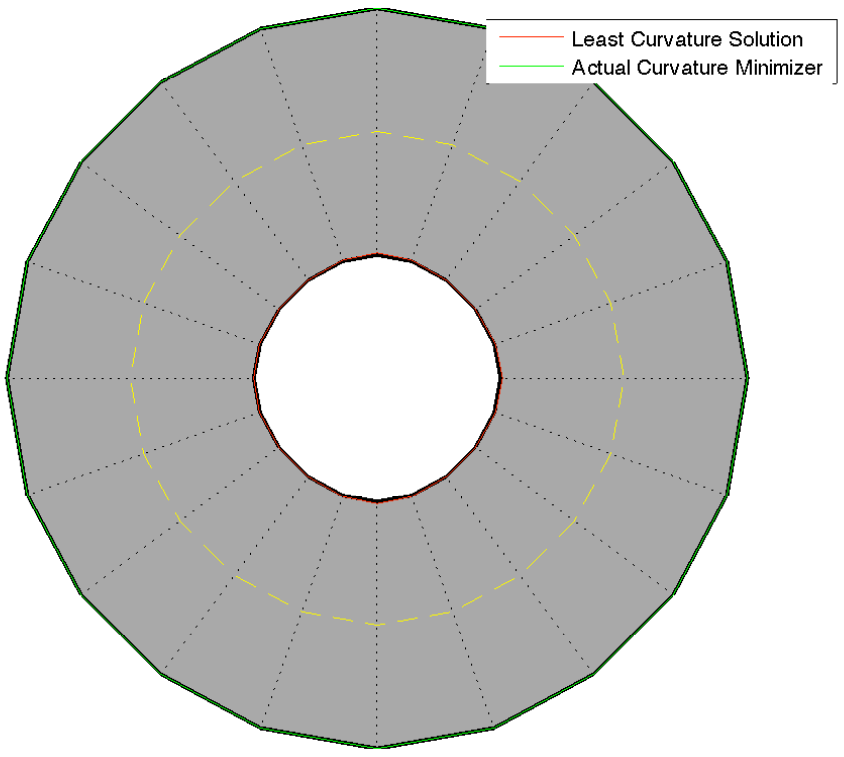


Figure 3.3: Are we really minimizing curvature?

around the origin[4], simply be a scaled version of $r_c(\theta)$, with scaling factor $\alpha = R$. Accordingly, we estimate curvature by

$$\hat{\kappa}_i = ||r''(\theta_i)|| = \alpha||r_c''(\theta_i)|| = R\kappa_{c,i} = R. \tag{3.1}$$

We conclude that, instead of scaling like $\frac{1}{R}$, our estimate of curvature scales as R. Considering squared curvature (assumption 2) further amplifies the discrepancy, causing scaling as R^2 instead of as R^{-2}. The real reason for the problem lies in assumption 1: let us assume we were to update the trajectory parameterization to $\tilde{\theta}(R)$, so that it actually reflects arc length as we vary R. In our simple example, we would obtain $\tilde{\theta} = R\theta$. Again starting from the center line trajectory $r_c(\theta)$, we would obtain the new trajectory, in the new parameterization variable, as

$$r(\tilde{\theta}) = Rr_c(\tilde{\theta}/R). \tag{3.2}$$

Accordingly, our estimate of curvature would become

$$\hat{\kappa}_i = ||r''(\tilde{\theta}_i)|| = R\left|\left|\frac{\partial^2}{\partial\tilde{\theta}^2}r_c\left(\frac{\tilde{\theta}_i}{R}\right)\right|\right| = \frac{1}{R}\left|\left|r_c''\left(\frac{\tilde{\theta}_i}{R}\right)\right|\right| = \frac{1}{R}\,||r_c''(\theta_i)|| = \frac{\kappa_{c,i}}{R}, \tag{3.3}$$

[4]This choice of coordinate system makes it easier to visualize the results stated; they are of course translation invariant.

which is the desired scaling. Unfortunately, introducing the parameterization as a variable in our optimization problem would cause it to be nonconvex, which is why we kept the parameterization constant in the first place.
It remains to be seen what the implications of this effect are. We can make a few tentative observations:

- We expect our approach to favor trajectories that run on the inside of curves. It therefore implicitly tends towards the shortest length solution that also in general favors insides of curves.

- This result implies a fundamental *limitation* of our approach based on minimizing a linear combination of least curvature and shortest length objectives, simply because we are not able to construct a pure curvature minimizing solution to begin with, which limits the set of candidates the optimization routine sweeps through[5]. At this point however, it is not clear how restrictive this limitation will be.

- On the other hand, one could try to argue that the effect is *beneficial.* Still considering the perfect circle example, one might wonder what the time-optimal trajectory and speed profile combination is. In this case, it is intuitively seen that the optimal trajectory (as discussed before) is a circle and that the optimal speed will be a constant[6], the only choice being the radius R and the speed s. For a given radius, the trajectory length is simply $2\pi R$ and the total travel time T can be bounded by

$$T \geq \frac{2\pi R}{s_{\max}} = \frac{2\pi R}{a_{\text{lat,max}}\sqrt{R}} = \frac{2\pi\sqrt{R}}{a_{\text{lat,max}}}. \tag{3.4}$$

  Equality is achieved when the lateral acceleration constraint is active. This implies that, for a given car, in order to minimize travel time, one will always choose R as small as possible.

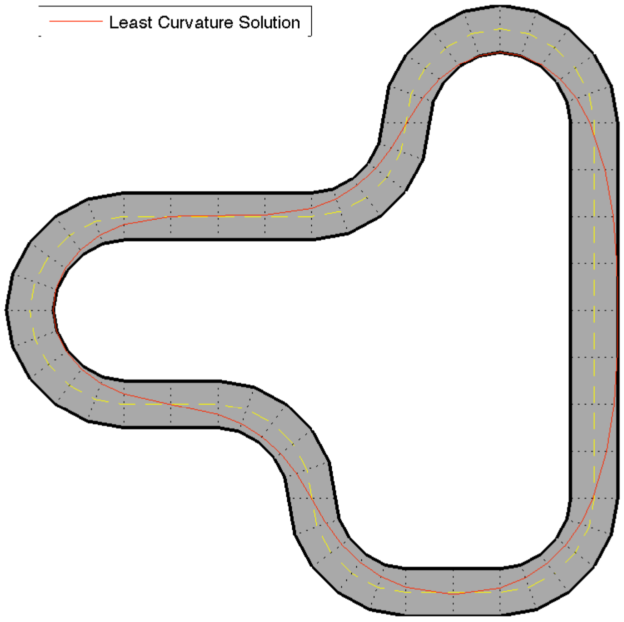
In general, while the perfect circle example gives us some valuable insight into our approach (and has nicely illustrated its main flaw), we should refrain from inferring too much about performance on actual tracks, given that it is rather artificial and stilized compared to typical (race) tracks.

For that reason, consider fig. 3.4a, which shows the least curvature solution, for a narrow track[7]. While it is not clear what the global curvature minimizer would be, our solution looks plausible. Notice how the trajectory is clinging to the inside of curves. The same track

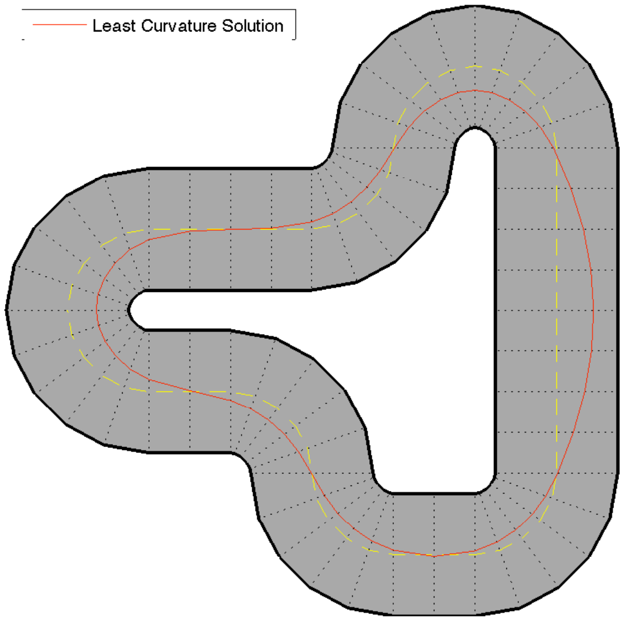
[5]Strictly speaking, our shortest length solution also is suboptimal, as was seen earlier; in that case however, the suboptimality is much less evident and can more easily be ignored.

[6]We are only looking at the steady state solution, *i.e.* consider a car driving at its final speed, ignoring any transient behavior.

[7]For closed loop tracks we require $\alpha_0 = \alpha_{N-1}$ to hold. This makes sense if we are looking for a steady-state closed loop trajectory. Interpolation for curvature estimation is in that case done using closed cubic splines, to enforce second order continuity also at the starting/end point.



(a) Narrow track



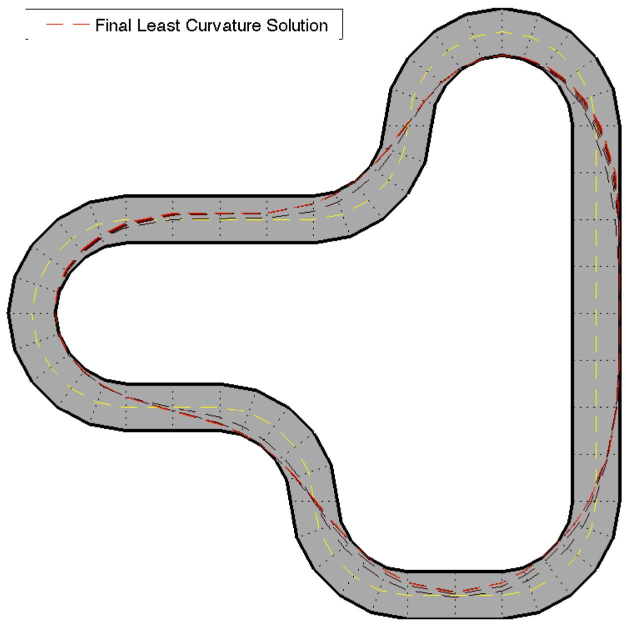
(b) Wide track

Figure 3.4: Least curvature solution for different track widths

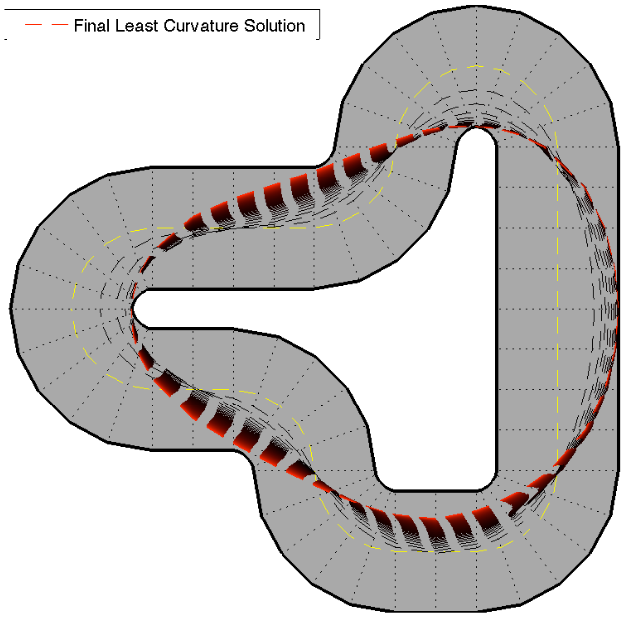
plus solution, for a wider track, can be seen in fig. 3.4b. The witnessed behavior is not as expected, the wider track solutions definitely are not globally minimizing curvature. One can still attempt to (somewhat handwavingly) explain this effect: the narrower the track, the more similar any (reasonable) trajectory will be to the center line in terms of arc length parameterization. Wider tracks allow for reasonable trajectories with arc length parameterizations differing substantially from the center line parameterization. As we have seen, the parameterization is crucial in estimating curvature along a trajectory. Thus, these trajectories will be penalized by the mismatch in parameterizations, *i.e.* their curvature will be overestimated, to the point of not being able to compete with solutions that are effectively worse but more similar to the center line.
In order to mitigate this problem of suboptimal arc length parameterization, we introduced a simple iterating approach. Each iteration consists of solving the least curvature problem and finding a new arc length parameterization[8], to be assumed as given in the next iteration. The results of this procedure for different widths can be seen in fig. 3.5. For these examples, we have not limited the number of iterations, the stopping condition is chosen as a simple threshold on $||\alpha - \alpha^{+}||$. Note that the sum of squared curvature samples can be shown to be non-increasing in the number of iterations. However, we do not know whether we are converging to a local or global minimizer.

We conclude that this effect can be largely countered by our iterating policy. Note however

[8]As usual, we are approximating the arc length parameterization by a chord length parameterization.



(a) Narrow track



(b) Wide track

Figure 3.5: Iterative least curvature solution for different track widths

the sky-rocketing increase in the number of iterations required to reach the same threshold for wider tracks – 12 iterations were required for the narrow track compared to 147 for the wide track. It clearly makes sense to restrict the maximum number of iterations, based on the computation time available. For all that follows, we will assume an upper bound of 20 iterations, which proved to be sufficient for most tracks and track widths considered.

As a side note, we have also considered minizing the *maximum* squared curvature sample instead of the sum. The problem with this approach is that the minmax objective alone does not lead to satisfactory solutions: trajectories will typically be reasonable in curves but can be very wobbly on straight track parts, simply because non-zero curvature values smaller than the overall maximum curvature are not penalized. Put differently, we witness the difference between minimizing the 1-norm and the ∞-norm, the 1-norm typically leading to sparse solutions. This effect can be controlled by introducing a regularizing term penalizing *e.g.* the sum of curvature samples. Still, the reason why we have not further investigated this approach is precisely this inherent sense of “locality”. While it might be interesting to apply this approach to track parts with a single curve, *i.e.* high curvature part, as soon as we consider a sequence of (possibly different) curves, the minmax behavior is less predictable and eventually washes out completely for long tracks. It then becomes questionable whether a minor potential gain justifies the increased computational cost of solving the (regularized) minmax problem.

We would like to say a few more words on the approach’s fundamental limitation men-

tioned earlier. The question is whether there are cases where this limitation, in particular the clinging effect, obviously leads to suboptimality. While it is hard to discuss optimality of curves without any reference material, we can very easily construct such examples by adding to the problem the additional dimension of *boundary conditions*. Assume a car with limited maximum longitudinal deceleration, entering a U-shaped track part at a fixed initial speed. The two-step approach will, independently of the current vehicle speed, construct a trajectory that clings to the inside of the curve, thus reducing the distance over which deceleration is possible. It follows that the speed profile problem can become infeasible even though choosing a longer trajectory (possibly with the same curvature profile) might have preserved feasibility[9]. We will discuss the aspect of boundary conditions and infeasibility in greater detail when discussing the receding horizon scenario in section 3.4. However, one does not have to look that far to find examples where our approach fails. Other straight forward examples can be based on a *symmetry* argument. Our trajectory problem will invariably generate paths that are symmetric w.r.t. the (piece-wise) symmetry axis' of the track. In practice cars however tend to have asymmetric longitudinal acceleration and deceleration behavior and limits, the upper bound on acceleration being lower in general. Intuitively, we would thus expect optimal trajectories not to be symmetric.
These observations confirm the heuristic character of our approach – one should always keep in mind that we are in general working with suboptimal trajectory solutions.

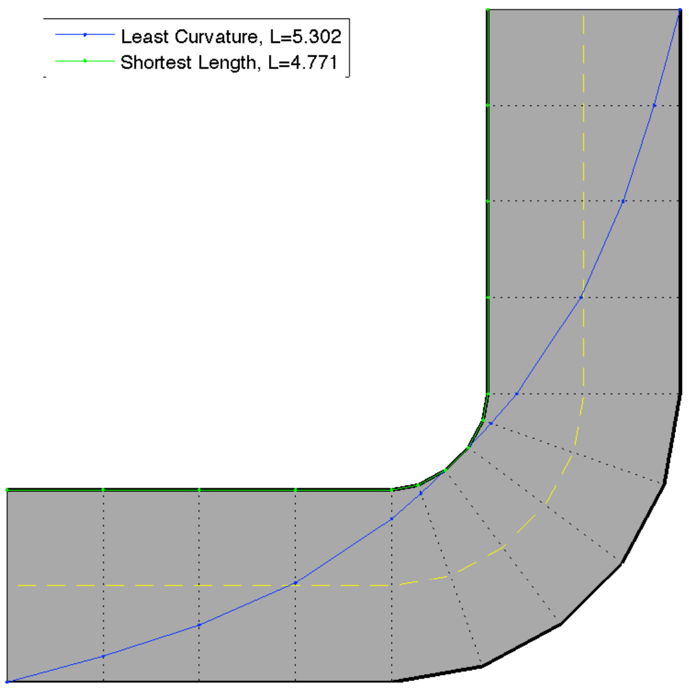
3.2.2 Speed Profile

Fig. 3.6a displays the shortest length and least curvature trajectory solutions for a simple curve track part, after 20 reparameterizing iterations. Fig. 3.6b compares the exact trajectory curvature[10], fig. 3.6c shows the evolution of speed and longitudinal acceleration. Note how the shortest length solution, due to higher curvature, forces the car to slow down while the least curvature speed stays constant all along the trajectory.
Fig. 3.7a shows the same two trajectories for a more complex track. Figs. 3.7b and 3.7c allow to compare the speed profiles for the two solutions. Here we have additionally plotted the longitudinal acceleration constraint, the speed constraint and the lateral acceleration constraint (which is effectively another constraint on speed, hence the same color; the actual speed constraint is given by the point-wise minimum of both constraints). The horizontal axis in both cases is chosen as the trajectory arc length parameter $\tilde{\theta}$.

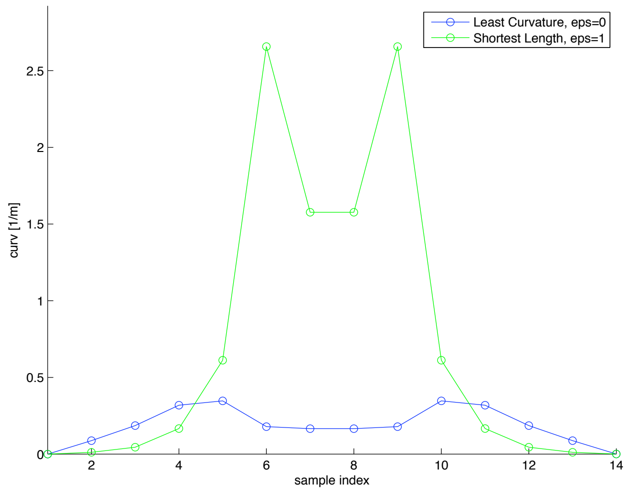
It goes without saying that the so far largely ignored car characteristics are the determining factor in finding the speed profile. In the preceding examples, we have, somewhat randomly, chosen a lateral acceleration bound of $10\frac{m}{s^2}$, longitudinal acceleration bounds of $+/-6\frac{m}{s^2}$ and a speed bound of $5\frac{m}{s}$. In all that follows, those are the default values that can be assumed unless stated otherwise. Note that these values are fictitious and chosen to give insight into

[9]This example is somewhat artificial in the sense that an actual driver would anticipate such a situation and, assuming sufficient track knowledge, attempt to slow down early enough.

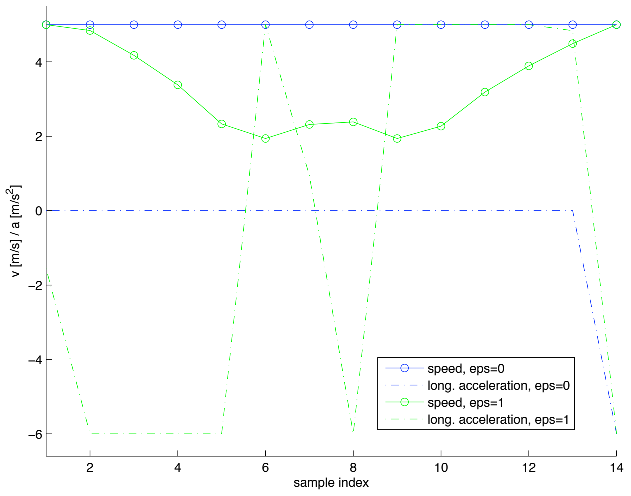
[10]This curvature is found using a cubic spline interpolation and the exact expression for curvature.



(a) Trajectory

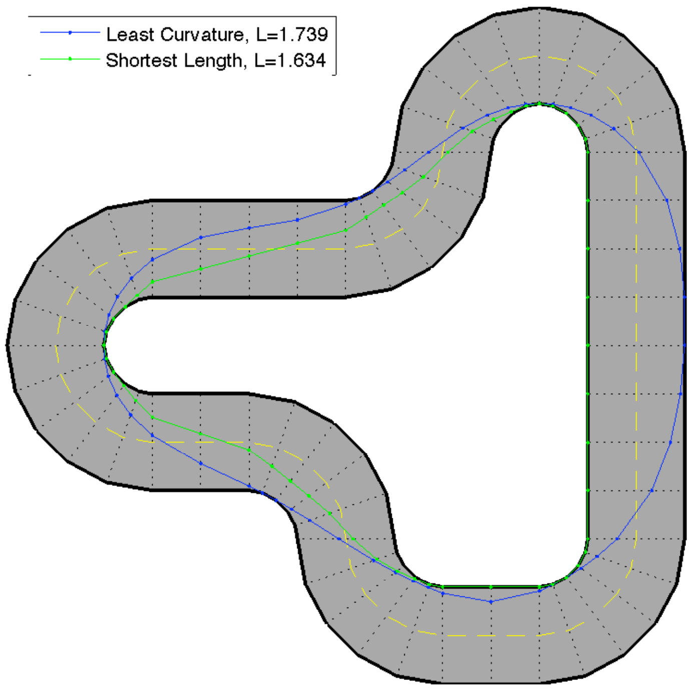


(b) Curvature

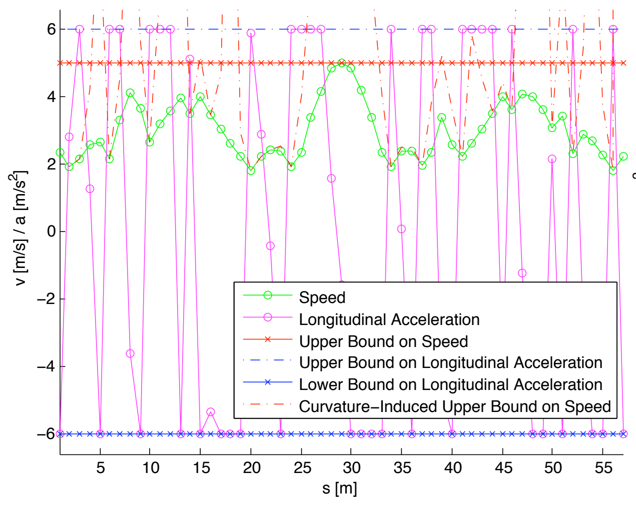


(c) Speed profile

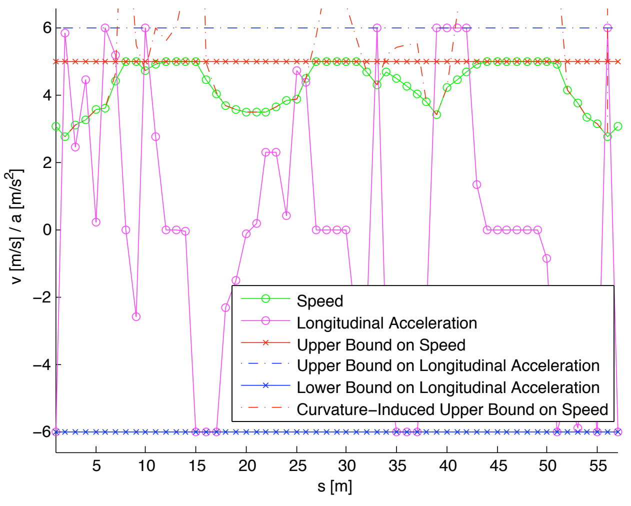
Figure 3.6: Shortest length and least curvature solutions



(a) Trajectories



(b) Shortest length speed profile



(c) Least curvature speed profile

Figure 3.7: Shortest length and least curvature solutions

the problem rather than to coincide with actual cars[11].

Surprisingly, it turns out that using the simplified objective function (2.29) instead of the actual travel time used in formulation 4 causes virtually no difference in the resulting speeds and travel time[12]. This result is very convenient in that it allows us to use the computationally appealing LP approach without worrying about optimality.

3.2.3 Feedback Solution

Finally, we consider linear combinations of the shortest length and least curvature solutions and how to choose the linear combination factor ϵ. Fig. 3.8 displays, for a given track, the pure shortest length and pure least curvature solutions as well as a number of linear combinations (in red). Note how the clinging effect causes solutions to be largely undiscernable in

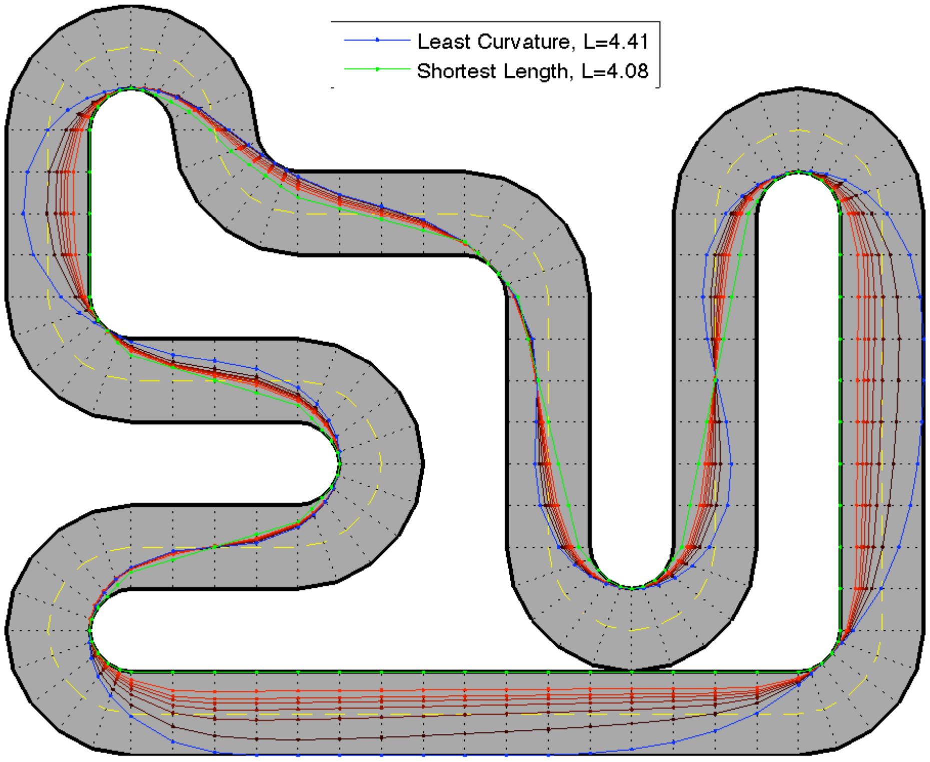


Figure 3.8: Combining length and curvature objectives

curves.

[11]Just to give an order of magnitude idea of real-world values, consider that a full-blown race car achieves an average longitudinal acceleration of $6 - 7\frac{m}{s^2}$ when going from 0 to 100 $\frac{km}{h}$; its lateral acceleration tops at around $1g - 1.5g$.

[12]At least no differences to speak of. A wide range of tracks, char characteristics and different path-finding methods all resulted in relative changes of $10^{-7} - 10^{-9}\%$.

We expect the optimal choice of ϵ to heavily depend on the type of car considered. This is exemplified in fig. 3.9. Here we have plotted, for different types of car profiles, lap times as we vary ϵ. Each car profile corresponds to heavily decreasing, as compared to the default

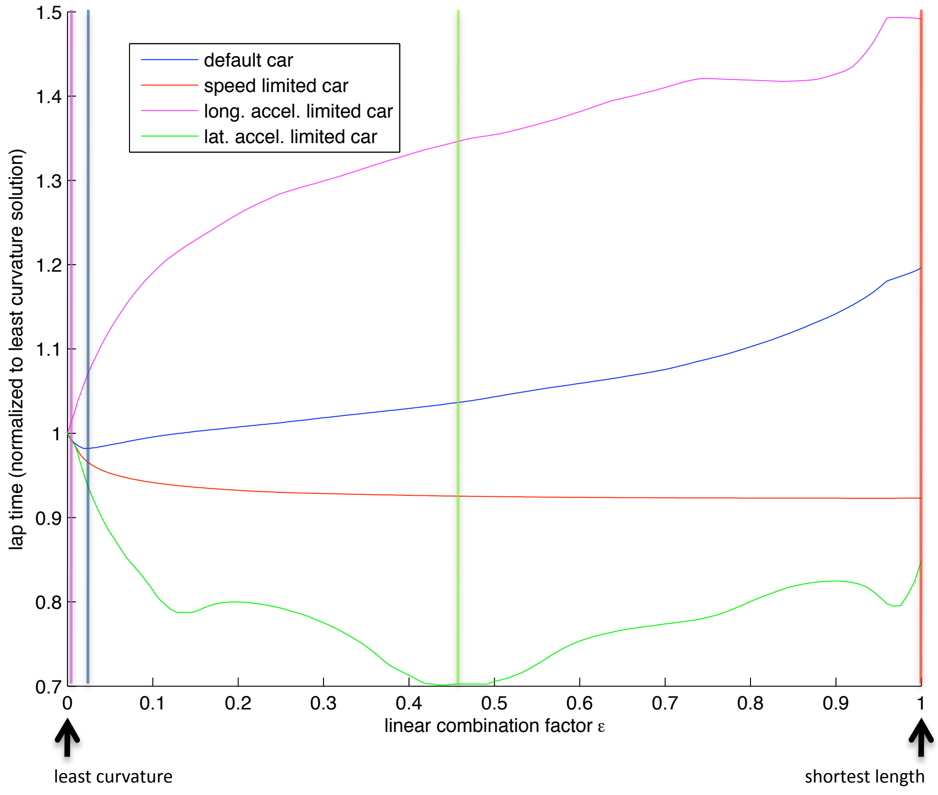


Figure 3.9: Lap times for varying length/curvature trade-off

car, one of the three dynamics constraints upper bounds. Note that lap times are normalized to the respective least curvature solutions since we are interested in how lap time evolves relatively rather than absolutely. We can interpret these results as follows:

- a car whose speed profile is largely limited by the upper bound on speed will tend to choose a shorter trajectory, given that, even on a high curvature trajectory, this constraint will be dominant
- a car with low longitudinal acceleration and deceleration bounds will prefer a long but smoother trajectory since slowing down and speeding up is costly
- a car limited by lateral acceleration chooses a solution somewhere in between

Note that this plot will look strikingly different for other tracks[13]. The reason for showing this particular example is to stress the strong dependence of the optimal ϵ on the car characteristics.

[13]Roughly, tracks that are different in their ratio of curved to straight parts.

In practice, the optimal ϵ can be determined by gridding. A possible strategy would be to start with a coarse uniform gridding (depending on width of the track, say 10 values to start with) and to iteratively apply a refined sampling granularity on a shrinked ϵ region centered in the most recent optimum. Computational efficiency is in general not crucial in optimizing ϵ, since this procedure is typically run once, off-line, even in a receding horizon scenario (*cf.* section 3.4).
Note that, once ϵ has been set, we expect the two-step approach to be computationally tractable even in real time, at high rates, *e.g.* 50Hz. Conducting a thorough quantitative analysis of computational cost would come down to writing custom solvers that exploit problem structure, employing linear algebra tricks and implementing everything in a fast compiled language, *e.g.* C, which is clearly outside the scope of this project. Handwavingly, solving for the trajectory takes up to 20 QPs, the speed profile problem a single LP. For track parts of sizes typical for a receding horizon scenario, *i.e.* 20 to 30 samples, we would expect these to be solvable in under 10 ms.

3.3 Bi-Convex Method

As mentioned earlier, the bi-convex method, rather than trying to come up with a trajectory from scratch, assumes a good trajectory to be known and attempts to improve that solution in a time-optimal sense. The approach brings up quite a few questions, such as:

- Could any of the steps involved be infeasible?
- How does the initializer matter?
- How much improvement in lap time can we expect?

The first question can be answered easily. Given the initializing trajectory, the first problem solved is a speed profile problem. That problem is (assuming no boundary constraints) always feasible: the all zero speed (and consequently all zero acceleration) is always a solution to the problem. As for the second half of the bi-convex iteration, the trajectory finding problem, it is also feasible: since the most recently chosen speeds are precisely fulfilling the acceleration constraints imposed by the latest trajectory, this same trajectory will always be a solution to the problem[14]. Note that the fact that we are not using the original longitudinal acceleration constraint, but were forced to linearize it, causes no problem. As mentioned in 2.2.2, we are only being conservative by linearizing.
On the other hand, it is not clear whether the laptime will certainly decrease (or at least stay constant) in each bi-convex iteration. It turns out that it does not, the reason for which is twofold and lies in the trajectory generation sub-problem:

[14]It turns out that, due to numerical issues, it is possible that the trajectory problem becomes infeasible when the iterations cause only minor updates. We will interpret this infeasibility as the algorithm having converged.

- We are not able to perfectly estimate the lateral acceleration for trajectories other than the previous iteration estimate. This is precisely the same effect that affected the two-step solution, *i.e.* that we assume a fixed parameterization for our spline interpolation.
- We are not optimizing lap time, strictly speaking, but the sum of *squared* segment traversal times. Note that the first condition is dominant.

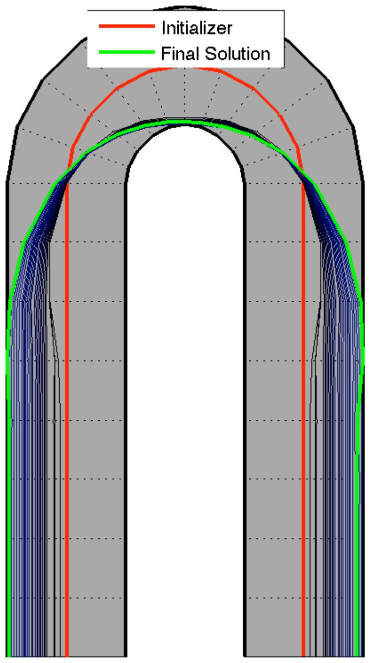
Effectively, this means that the trajectory optimization might find a solution that fulfills the acceleration constraints but does so only under the assumption of *incorrect* lateral acceleration estimates. Its estimate of the lap time will in that case be off[15], which allows the trajectory to come out on top in the optimization. Consequently, it is possible that the new speed profile found at the beginning of the next iteration will achieve a worse lap time compared to the previous iteration.
Ideally, without the two above-mentioned conditions, lap times would certainly be non-increasing from one iteration to the next. However, there is not much one can do about this problem that is inherently linked to the way we estimate curvature. Furthermore, results have shown that, in most cases, lap time is also not convex in the iteration index, meaning that, even after observing an increase in lap time, additional iterations can lead to further improvements. Both for the sake of computational feasibility and because of the unpredictability of this behavior, we will, in general, set an upper bound on the number of bi-convex iterations.

The next question to ask is under which conditions we can expect to see improvements in lap time. We start by making a more general observation about the problem. Looking at the trajectory problem formulation 5 objective function, we notice that, assuming a fixed speed profile, improvements can be produced, roughly speaking, by one of two effects:

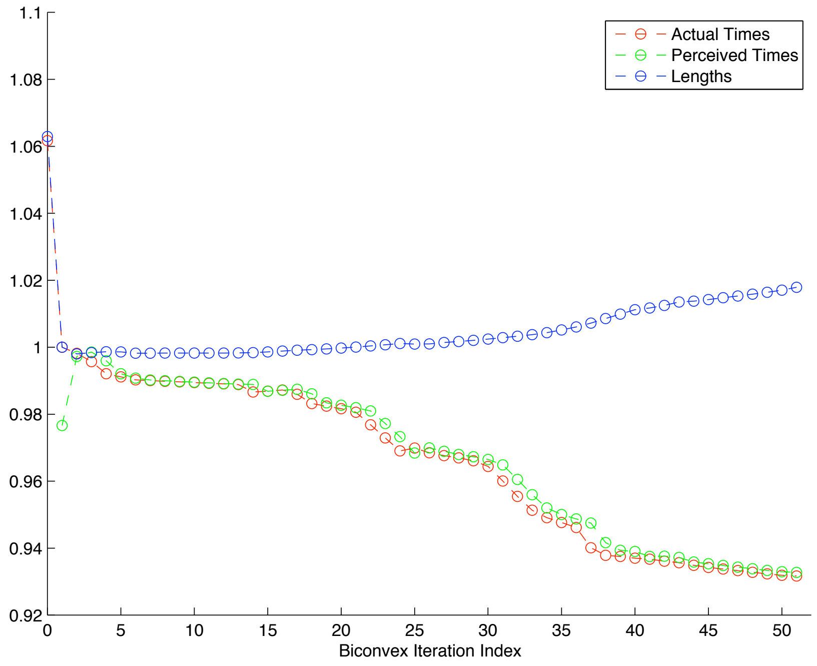
1. by "rearranging" the trajectory segment lengths that act as weighting factors for the speeds; in this case, increasing one segment length has to be balanced out by decreasing another
2. by simple shortening of segment lengths

Intuitively, we expect the second effect to lead to more radical improvements, since reducing segment lengths univocally translates into lap time gains. Overall then, we would expect the biconvex method to have an inherent tendency to "contract" trajectories whenever effect 2 can become active. The question remaining is *under which circumstances* each of the two effects comes into play.
Let us look at an example. We consider a U-shaped track, initialize the procedure with the center line and choose the default car profile. Fig. 3.10a shows the biconvex iterations, fig. 3.10b displays the evolution of normalized trajectory length and lap times against the iteration index. Note that by "perceived times" we mean the lap times as (incorrectly)

[15]Note however that mis-estimating time does in general not preclude the *actual* lap time from improving, provided we are lucky enough that the trajectory found allows for such improvement.



(a) Biconvex iterations



(b) Evolution of lap times

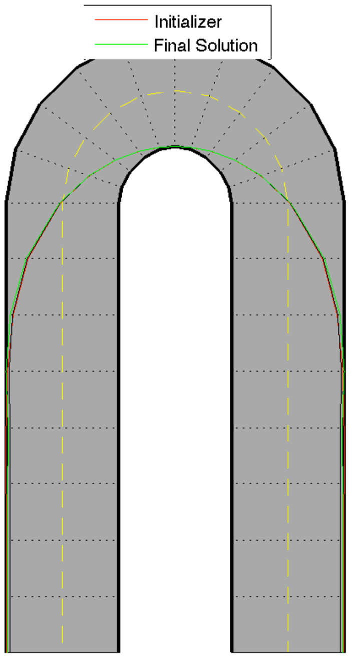
Figure 3.10: Initializing with center line

estimated by the trajectory optimization routine, while "actual times" denotes lap times based on the corrected speed profile. The procedure converged after 51 biconvex iterations. We make the following observations:

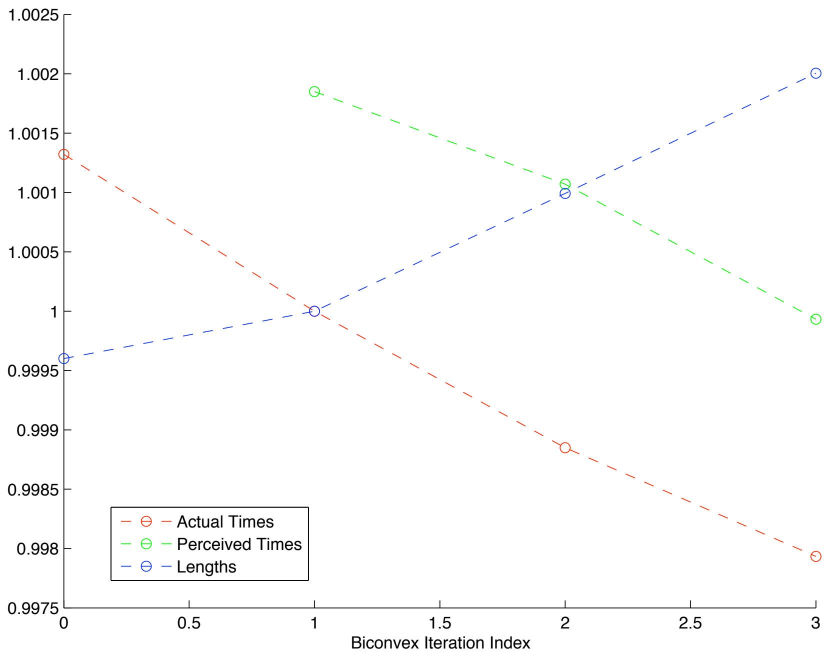
- The first iteration produces a massive trajectory shortening that is, as expected, accompanied by a large improvement in lap time. This confirms that, whenever possible, effect 2 kicks in and the trajectory is shortened.

- Later iterations produce only minor improvements that mostly stem from reorganizing segment lengths. This can be interpreted as effect 1 being active.

We conclude that only the first iteration really is meaningful in its effect or, put differently, efficient from a computational point of view. Fig. 3.11 shows another example. This time, we have initialized with the least curvature two-step solution. After merely three iterations, the $\Delta\alpha$ threshold condition already kicks in. The "contraction phase" is missing completely, the procedure directly switches to rearranging segment lengths, with hardly noticeable improvements. Overall, the lap time improves by 0.34%, as compared to 13.02% in the previous example. From a practical point of view, it is questionable whether this can be considered a real improvement.
Comparing these two examples again brings up the question under which conditions we actually observe effect 2, *i.e.* that of a major trajectory contraction? The answer to this question can be found looking at the problem constraints. Large scale contraction should be possible



(a) Biconvex iterations



(b) Evolution of lap times

Figure 3.11: Initializing with least curvature solution

whenever the combination of initializing trajectory and speed profile allow for it, *i.e.* neither the lateral nor de longitudinal acceleration constraints are active to begin with.
Crucially, this expected behavior is only idealized. In reality, the curvature estimation problem thwarts us once more. The longitudinal acceleration constraint is "hard" in the sense that it uses accurate estimations of segment lengths and, once active, definitely does not allow for (major) shortening of the trajectory. On the other hand, the lateral acceleration constraint, as pointed out earlier, takes into account suboptimal curvature estimations. We saw, when discussing the two-step approach, that contracting a trajectory in curves leads to an *under*estimation of curvature and thus, given fixed speeds, of lateral acceleration. This means the optimization routine could move the trajectory inwards more than the lateral acceleration constraints should allow for. Consequently, even if the initializer had active lateral acceleration constraints, the procedure might update the trajectory to one that is tighter in curves. As a side note, observe that the curvature misestimation effect also provides an additional explanation for why we would overall expect a predominantly contractive behavior: "dilating" a trajectory leads to over-estimation of curvature – it is thus possible that the estimated curvature wrongly violates the lateral acceleration constraints.
This is exactly what we could see in our first example: the center line trajectory and speed profile solution did activate lateral acceleration constraints inside the curve, but the first biconvex iteration still led to a substantial contraction of the trajectory, effectively ignoring the constraints. This effect can be seen looking at 3.10b. The estimated lap time after iteration 1 was too optimistic and speeds had to be down-corrected at the beginning of iteration

2, leading to a significantly higher actual lap time. The fact that we nevertheless observed some decrease in actual lap time can be attributed to the beneficial effect discussed in section 3.2.1. Looking at the following iterations, we see that virtually all of them elongate the trajectory, *i.e.* they try to correct the overly aggressive behavior of the first iteration. The comparably small step size can be explained by a combination of overestimating curvature and the necessity that elongations and shortenings of segment lengths balance each other out[16].
The example of fig. 3.11 showed none of this behavior. This can be explained by noting that the initial contraction we witnessed in example 1 was already included, for example 2, in the initializing least curvature solution. More generally, we expect major shortenings of trajectories primarily when the flawed curvature estimation allows for them[17].

All in all, these results are rather deceiving. Recapitulating, we have made the following observations:

- Effect 1 is relatively predictable but slow, only providing small scale improvements, and computationally inefficient.
- Effect 2 typically relies on under-estimating curvature, is too aggressive and its outcome is hard to predict. While it does lead to large scale improvements, it tends to heavily overshoot.
- There is an inherent *asymmetry* between contracting and expanding trajectories. Contractions are favoured by the objective function and by the induced under-estimation of curvature. Similarly, expansions are impeded by the objective function and by the induced over-estimation of curvature. Consequently, effect 1 thus takes many iterations to correct the overshooting produced by effect 2.

We conclude that, in this form, the only application of the biconvex approach would be to achieve small scale improvements produced by effect 1, *i.e.* to "fine-tune" an existing and close-to-optimal trajectory.
A potential remedy that we have not investigated due to time constraints but that might make effect 2 practically interesting would be to introduce maximum iteration step sizes, *e.g.* by upper bounding $\Delta\alpha$. We would expect this to reduce the overshoot effect and make the biconvex approach more "symmetric".
Another aspect that we have not had the time to analyze in depth is the effect of different car profiles on the final biconvex solution. For instance, we mentioned earlier that one advantage of the biconvex method was the absence of assumptions concerning the form of the optimal trajectory. We have observed that asymmetric bounds on longitudinal acceleration lead to

[16]Note that in general, decreases in trajectory length lead to underestimating lap time while elongations lead to overestimating lap time.

[17]There are cases, *e.g.* with a speed limited car, where the lat acceleration constraint indeed is not active for the initializing solution. In this case, a considerable shortening might be possible without violating constraints.

asymmetric biconvex solutions. One would however need to investigate the optimality of such solutions in more detail.

We conclude this section with two examples on a more complex track. We choose the initializers so that primarily effect 1 is active. Fig. 3.12a shows the biconvex iterations when initializing with the two-step least curvature solution ($\epsilon = 0$), fig. 3.12b when initializing with the two-step optimal solution ($\epsilon = 0.018$). For both examples, we assume the default car profile. Convergence is achieved after 3 iterations, in both cases, with a total improvement of 0.35% and 0.38% in lap time. These improvements are similar to what we observed for example 2 – again, it is questionable whether the minor gains are worth the computational strain. Note that the final solution for the least curvature initialized version still lies 1.2% in lap time below the original two-step optimal solution[18].

3.4 Receding Horizon Example Application

3.4.1 Problem Formulation

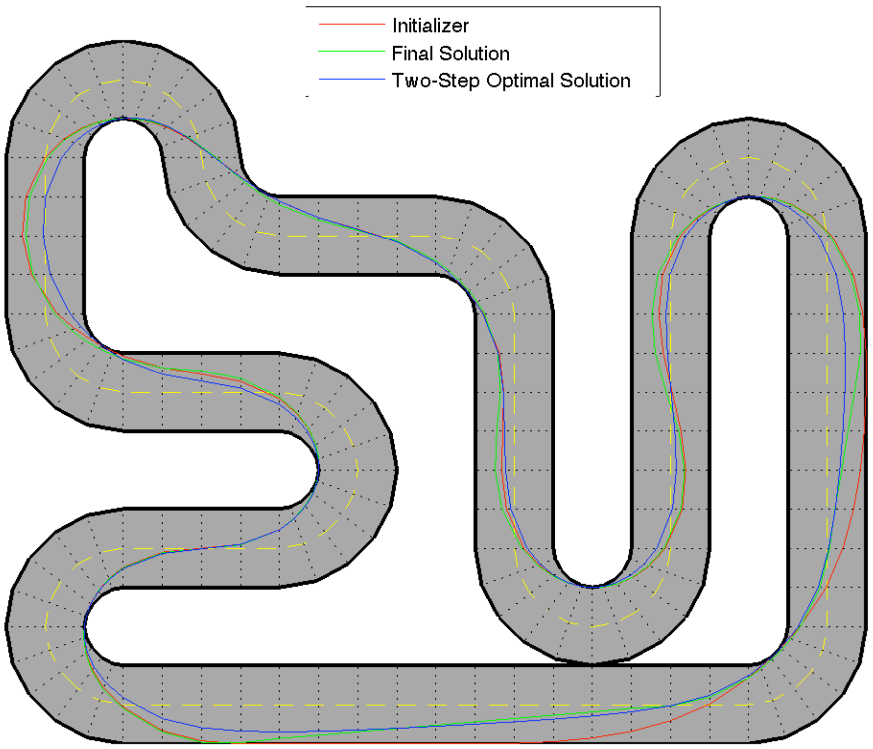
We consider a known closed loop track and would like to determine the lap time-minimizing car trajectory and speed profile to go around the track exactly once. While one might consider using a "one-shot" solution calculated off-line, for the entire track, this approach fails as soon as there are unknown[19] disturbances such as, *e.g.*,

- internal disturbances like faulty car control or
- external disturbances like other road users.

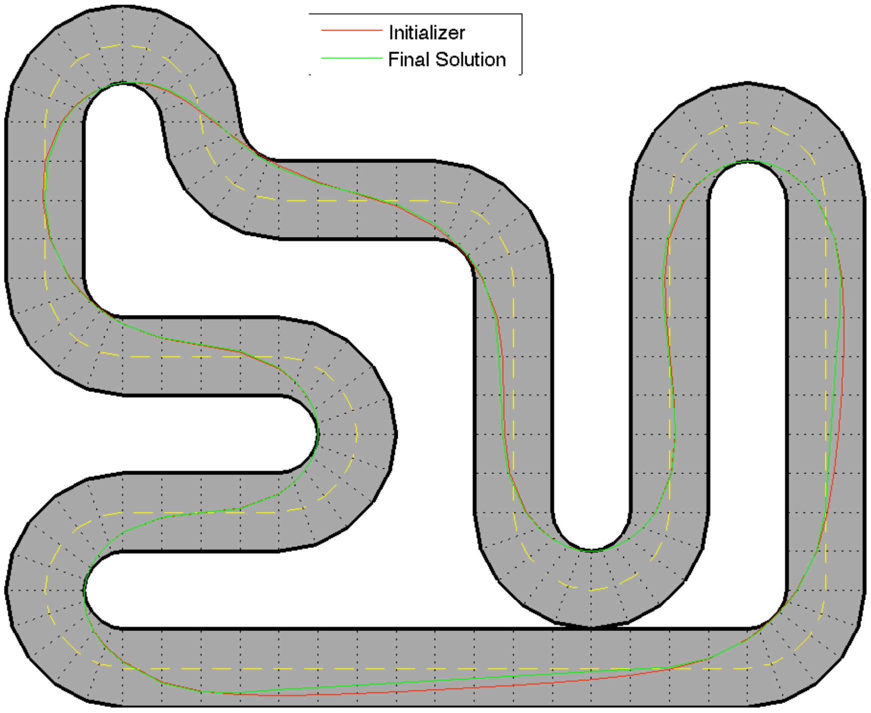
A *receding horizon* (RH) approach lends itself to coping with any kind of disturbances: instead of solving the problem once for the entire track, one solves it repeatedly, once at the beginning of every track segment, for a (usually) smaller part of the track. The newest trajectory and speed profile solutions are then used for one track segment, until the problem is solved again. The main parameter here is the so-called *horizon* $N_{\text{horizon}} \in \{1, 2 \ldots, N\}$ that dictates for how many track sample points, starting from the current sample, this "moving window" solution is computed in every step. The horizon would in practice be set based on computational resources and on the amount of track data available. This procedure introduces *feedback* that allows to take into account the aforementioned disturbances.
Based on the results from the previous section, we will be using the two-step solution, as it offers a good trade-off between optimality and computational efficiency. In all that follows, we invariably assume ϵ to be set to the value that minimizes lap time for the one-shot solution of the given track and car. We will in all examples consider the default car configuration.

[18]This suggests that the biconvex method, at least in its current form, is not suitable for replacing the two-step ϵ-optimization sub-problem.

[19]Unknown at the time of solving the optimization problem.



(a) Least curvature initializer



(b) Optimal initializer

Figure 3.12: Different initializers

3.4.2 Implementation

The receding horizon scenario brings in two aspects that we have so far largely ignored: *boundary conditions* and *boundary effects.*

We now have to consider boundary conditions, more precisely starting conditions, that reflect the current vehicle state. These conditions enter the optimization problem as additional constraints on the initial car position α_0 and car speed s_0. Since these constraints are simple linear equality constraints, the problem formulation remains essentially the same. Note that the starting conditions introduce feasibility issues, as we will discuss shortly.

The second new aspect is what we call boundary effects. These are intrinsically linked to the way the two-step procedure estimates curvature. As explained earlier, we use natural cubic splines to interpolate in between sample points and sample these splines to estimate curvature, both for the trajectory and speed profile generation. By definition, our first and last curvature sample will thus be zero.
For the one-shot problem case, this is a reasonable assumption: cars tend to start and end their laps driving relatively straight. Even if this were not so, boundary effects would tend to be negligeable for sufficiently long tracks. In case of a receding horizon implementation however, the situation is fundamentally different: the car is entering a new track segment at a certain angle. Using natural splines, as described in section 2.1.2, for interpolation amounts to *ignoring* the current car orientation. In practice, this would allow for erratic changes in car direction that are not accounted for by our approach. Even if a (violent) change in orientation should violate the lateral acceleration constraint, it will not, because the starting curvature is invariably estimated as zero, which effectively neutralizes the starting point lateral acceleration constraint. This can lead to rugged trajectories and does not respect the cars physical limitations.
We propose two ways to mitigate this problem. The first is to use clamped instead of natural cubic splines, which would allow to enforce continuity in the first derivative at the starting point. The second approach, which we have implemented, is to introduce a *lookback horizon* N_{lb}: We include the N_{lb} most recent trajectory samples to do the spline interpolation required for curvature estimation. This allows to take into account the current car direction and leads to a robust, *i.e.* smooth over time, curvature estimation. In general, a longer lookback horizon should lead to a more robust curvature estimation since the effect of setting the first sample curvature to zero diminishes with increasing distance from the starting point.

As an example, consider fig. 3.13, which compares one-shot and receding horizon solutions for a given track, assuming no disturbances but limited view. In this case, we have chosen $N_{\text{horizon}} = 15$, $N_{\text{lb}} = N_{\text{horizon}}$, and have restricted the number of two-step iterations to 20. Markers indicate samples in *time* (sampling interval 0.05s), marker size is chosen to correlate with vehicle speed. Notice how, even with this relatively modest horizon, the RH solution is almost identical to the one-shot solution. It turns out that the one-shot solution achieves a lap time of 10.47s, while the RH lap time lies at 10.51s. Restricting ourselves to a finite

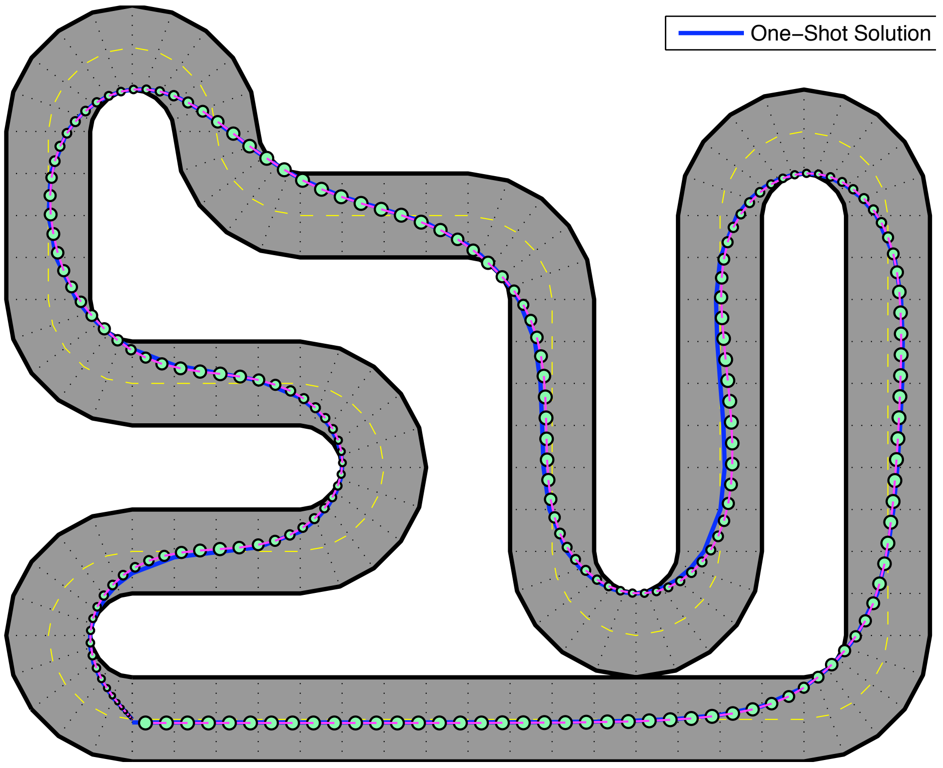


Figure 3.13: One-shot and receding horizon solutions compared

horizon of 15 samples thus costs us a mere 0.4% in lap time.

3.4.3 Feasibility

As mentioned, introducing boundary conditions introduces feasibility issues, which can be seen as follows.
The trajectory problem is always feasible: its only constraints are constant upper and lower bounds on α and the starting condition α_0, by definition, will always allow these constraints to be fulfilled. For the speed profile on the other hand, the situation is less clear. Problems can be caused by a combination of three factors:

- we have a boundary condition on s_0
- the speed profile problem constraints depend on the trajectory that was derived *without* taking into account the car dynamics and the boundary condition
- our curvature estimation can be jumpy due to boundary effects[20]

In other words, if a newly found trajectory happens to be too "aggressive" for the current vehicle speed or if the curvature estimation is not robust enough, the speed profile generation

[20]Even using a lookback horizon and a long prediction horizon, jumps are possible.

problem will be infeasible. Note that, while this problem is strongly accentuated by the two-step procedure's splitting of the global problem, the risk of infeasibility would remain even for an approach solving our problem globally[21]. It is always possible to construct (pathetic) cases that cause the receding horizon implementation to fail. The question is whether there are ways to modify our approach to reduce this risk.

We propose a series of modifications that should improve chances of always being feasible:

1. Choosing an ϵ smaller than the optimal one. This should lead to less aggressive trajectories, closer to the least curvature solution, that are more likely to be feasible. This measure is obviously conservative.
2. Introducing terminal state constraints. By forcing the last sample position and speed to lie on the one-shot solution, we hope to be overall closer to this solution we know is feasible.
3. Enforcing continuity in trajectory slope. By this we mean choosing the first segment slope of the new trajectory to equal the second segment slope of the previous trajectory. This measure too is conservative.

One could also consider reusing an old solution in case the new problem turns out to be infeasible. However, since this effectively means giving up feedback and thus ignoring any disturbances, this approach is not really desirable or practical.

Note that, using any of these measures, there are no absolute guarantees for feasibiliy. We can only hope to reduce the risk of being infeasible. For example, table 3.14 shows the effect of applying counter-infeasibility measures 2 and 3. We compare lap times for the one-shot

		Finite Horizon			
	Horizon	5	10	15	20
With Constraints	Lap time (measured)	-	1.05183	1.04777	1.04532
	Lap time (recalculated)	-	1.05281	1.04981	1.04696
Without Constraints	Lap time (measured)	-	-	1.04815	1.04569
	Lap time (recalculated)	-	-	1.05061	1.04719

Figure 3.14: Lap times with and without additional constraints

[21]Without any terminal state constraints.

solution as well as for different choices of finite prediction horizon. “Lap time measured” was obtained from car behavior simulated based on the RH solution. “Lap time recalculated” results from considering the RH trajectory solution as a whole and recomputing the optimal speed profile for it. Dash entries indicate that the speed profile problem became infeasible at some point around the track, in which case the procedure exited. The one-shot lap time lies at 1.04750, red font indicates worse performance, green font better performance. Note that we have fixed N_{lb} at 20 samples. We make the following observations:

- Lap time benefits from longer prediction horizons. This makes sense given that a longer horizon allows to better cope with the upcoming track geometry.

- Lap time benefits from the feasibility improving measures, which is due to being closer to the one-shot optimizer.

- The recalculated lap times are higher than the measured ones. We conclude that the finite horizon speed profile in general is too optimistic/aggressive. This can be explained by boundary effects causing suboptimal curvature estimation. The same effect can be seen using the one-shot trajectory solution and only computing the speed profile in a receding horizon fashion. We conclude that a practical implementation would have to drive somewhat conservatively.

- Interestingly, the RH solution manages to slightly outperform the one-shot solution, *even* for the (more realistic) recalculated times, if we consider sufficiently long horizons. It is a priori not clear why this is so – we attribute the effect to the intransparency of the two-step splitting of the global problem. We conclude once more that the two-step solution is indeed suboptimal. Note that for even longer horizons, lap times tend to increase again.

- A longer prediction horizon reduces chances of being infeasible. The additional constraints do so too, which means they allow us to choose a shorter horizon.

Note that the results concerning lap times should be interpreted with care. Given the heuristic nature of our approach, potential numerical effects and the marginal differences observed, the results could be car and/or track configuration specific. Also, it is questionable whether such small scall improvements justify the associated substantial increases in computational complexity.
From a practical point of view however, most useful is the fact that we can reduce the horizon without incurring major increases in lap time while remaining feasible thanks to the additional constraints.

Due to time constraints, we have not analyzed how actual disturbances would affect the feasibility of the problem. Intuitively, we would expect feasibility issues to be even more crucial. Indeed, disturbances such as traffic, being more stochastic and possibly closer to our vehicle, should disrupt the smooth evolution of the car predicted trajectory curvature much more than the changes induced by the track segments newly discovered at the end of the horizon. It is a priori not clear how to cope with this issue since we would expect simple measures like the ones taken above to be able to deal with minor disruptions only. One would thus most likely not exclusively rely on a RH implementation of the two-step approach, but combine it with a different, less aggressive method that deals with large scale disturbances such as road traffic. For further information, we refer the reader to REF, a more thorough analysis of feasibility in a receding horizon scenario.

Chapter 4

Outlook & Conclusion

4.1 Outlook

There are a number of aspects that one might be interested in (further) looking into (in no particular order):

- Considering an improved car model. This could mean introducing more elaborate dynamics constraints into the existing model or choosing a more complex model to begin with. For instance, without any substantial changes to the problem formulation, bounds on acceleration can easily be made dependent on car speed, as long as the constraints remain affine in all variables.
- Implementing the two-step approach in a compiled language. The two sub-problems contain a lot of structure which, if exploited, should allow for a blazingly fast implementation.
- Comparing the two-step optimal solution to SCP and DP solutions, both in terms of optimality and computational efficiency.
- Modeling actual disturbances, such as traffic, in the receding horizon scenario.
- Implementing the two-step approach, both one-shot and receding horizon, on an actual track and car.
- Further investigating the biconvex approach. In particular, introducing step sizes to control the extent of curvature estimation problems.
- Looking for more robust ways to estimate curvature.

4.2 Conclusion

In the course of this project, we have looked at two approaches to approximately solving our initial problem formulation 1.

The iterative two-step approach seems to work well for a wide range of cars and tracks and, being easy to configure and a computational light-weight, lends itself to a practical implementation, also in a time-critical setup. It is able to deliver arbitrarily good results, provided we grid ϵ finely enough. On the downside, it makes quite stringent assumptions as to what a trajectory should look like. We have seen that these assumptions in some cases clearly cause results to be suboptimal. Also, in order to be used in a receding horizon scenario, it is crucial that feasibility issues be resolved.

Our verdict on the biconvex approach is much less clear. While interesting in theory, it seems to be of questionable practical use. Its performance is not consistent enough and the required computational effort is hard to estimate but from experience appears to be relatively high. In its current form, the approach could at best be used to marginally improve close-to-optimal trajectory solutions. Using the approach as a replacement for the trade-off optimization part of the two-step approach is definitely over-ambitious.

Interestingly, the performance of both approaches lives and dies with the (in)ability to estimate curvature reliably. While the two-step approach handles the flawed curvature estimation quite well, the biconvex approach considerably suffers from it. This also brings up the question whether convex optimization is the right way to approach this problem – after all, it is the convexity requirements that preclude us from using accurate estimations of curvature.