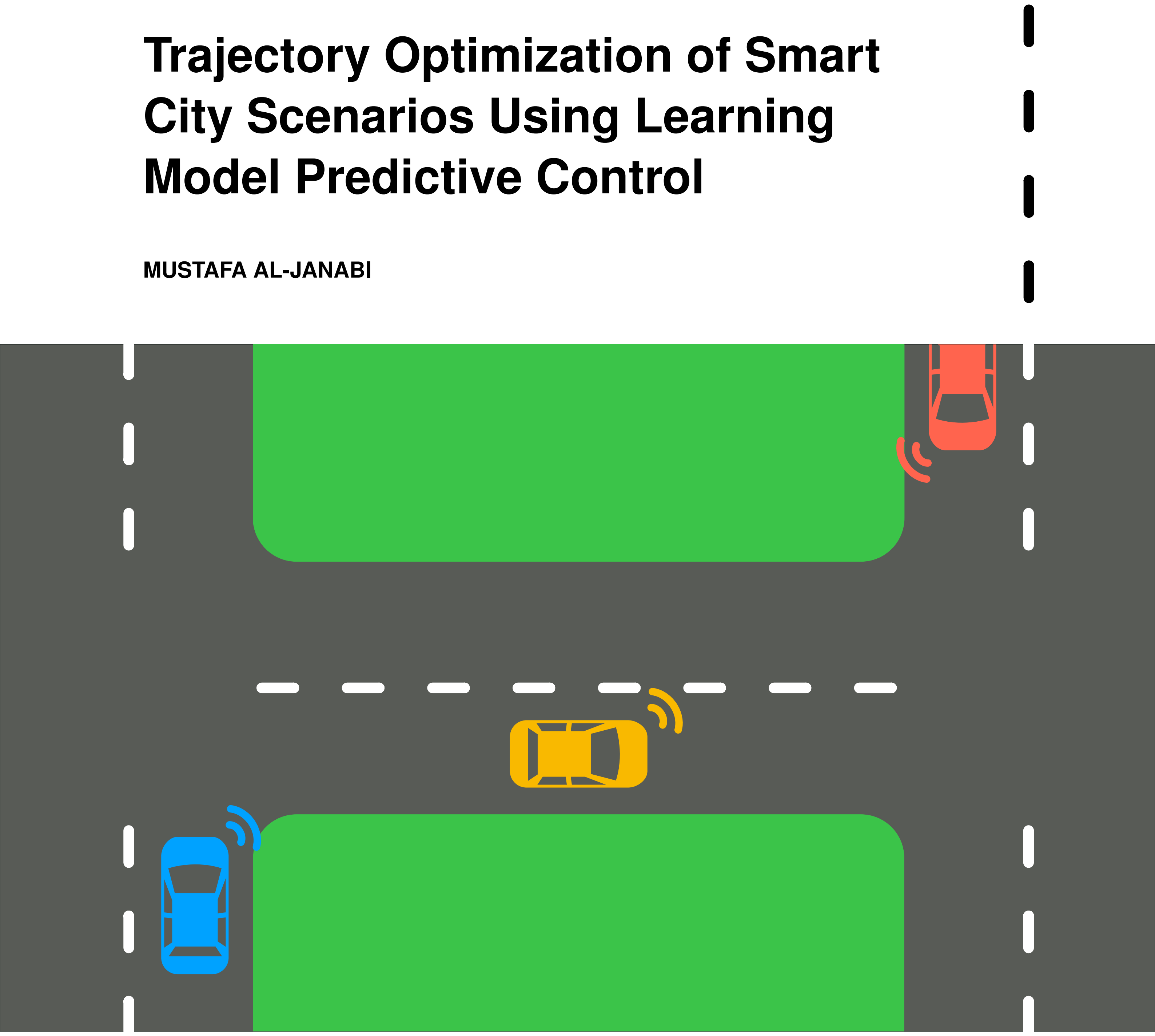
Degree Project in Engineering Physics and the main field of study Electrical Engineering, specializing in Systems, Control, and Robotics

Second cycle, 30 credits

Trajectory Optimization of Smart City Scenarios Using Learning Model Predictive Control

MUSTAFA AL-JANABI



Stockholm, Sweden, 2023

Trajectory Optimization of Smart City Scenarios Using Learning Model Predictive Control

MUSTAFA AL-JANABI

Master’s Programme, Systems, Control and Robotics, 120 credits
Date: May 5, 2023

Supervisor: Frank Jiang
Examiner: Jonas Mårtensson
School of Electrical Engineering and Computer Science

Abstract

Smart cities embrace cutting-edge technologies to improve transportation efficiency and safety. With the rollout of 5G and an ever-growing network of connected infrastructure sensors, real-time road condition awareness is becoming a reality. However, this progress brings new challenges. The communication and vast amounts of data generated by autonomous vehicles and the connected infrastructure must be navigated. Furthermore, different levels of autonomous driving (ranging from 0 to 5) are rolled out gradually and human-driven vehicles will continue to share the roads with autonomous vehicles for some time. In this work, we apply a data-driven control scheme called Learning Model Predictive Control (LMPC) to three different smart city scenarios of increasing complexity. Given a successful execution of a scenario, LMPC uses the trajectory data from previous executions to improve the performance of subsequent executions while guaranteeing safety and recursive feasibility. Furthermore, the performance from one execution to another is guaranteed to be non-decreasing. For our three smart-city scenarios, we apply a minimum time objective and start with a single vehicle in a two-lane intersection. Then, we add an obstacle on the lane of the ego vehicle, and lastly, we add oncoming traffic. We find that LMPC gives us improved traffic efficiency with shorter travel. However, we find that LMPC may not be suitable for real-time training in smart city scenarios. Thus, we conclude that this approach is suitable for simulator-driven, offline, training on any trajectory data that might be generated from autonomous vehicles and the infrastructure sensors in future smart cities. Over time, this can be used to construct large data sets of optimal trajectories which are available for the connected vehicles in most urban scenarios.

Keywords

LMPC, multi-agent control, Smart City, Data-driven Control, multi-level autonomy.

Sammanfattning

Smarta städer använder modern teknik för att förbättra transporteffektiviteten och säkerheten. Med införandet av 5G och ett allt större nätverk av uppkopplade sensorsystem för infrastruktur blir realtidsmedvetenhet om vägförhållandena en verklighet. Denna utveckling medför nya utmaningar. Kommunikationen mellan autonoma fordon och uppkopplade sensorsystem ger upphov till stora mängder data som måste hanteras. Dessutom kommer fordon med olika autocnominivåer (från 0 till 5) att behöva dela gatorna tillsammans med människostyrda fordon samtidigt under en tid. I detta arbete tillämpar vi en datadriven reglermetod som heter Learning Model Predictive Control (LMPC) på tre olika scenarier i en smart stad med ökande komplexitet. LMPC utnyttjar data från en tidigare lyckad körning av ett visst scenario för att förbättra prestandan på efterföljande körningar samtidigt som säkerheten och rekursiv genomförbarhet garanteras. Vidare garanteras att prestandan från en körning till en annan inte minskar. För våra tre scenarier är målet att minimerar restiden och börjar med ett enda fordon i en tvåfilig korsning. Sedan lägger vi till ett hinder på högra filen och till sist lägger vi till mötande trafik. Vi finner att LMPC ger oss förbättrad trafikeffektivitet med kortare restid. Vi finner dock att LMPC må vara mindre lämplig för realtids scenarier. Således drar vi slutsatsen att denna metod är lämplig för optimering i simulatorer, offline, på data som kan genereras från autonoma fordon och sensorsystemet i infrastrukturen. Så småningom kan vår metod användas för att konstruera stora dataset av optimala trajektorier som är tillgängliga för uppkopplade fordon i framtidens smarta städer.

Nyckelord

LMPC, smarta städer, multiagentsystem, datadriven reglerteknik.

Contents

Chapter 1

Introduction

In this chapter, we overview the problem addressed in this thesis and define the scope of the project. Furthermore, the introduction includes an illustration highlighting how we tackle the research question in focus.

1.1 Overview

Our cities are increasingly becoming smart. Smart cities are urban areas that utilize technology to improve quality of life and increase efficiency. This includes the use of connected infrastructure sensors and connected/autonomous vehicles to improve transportation efficiency and safety. With the rollout of 5G, edge computing, and ever more connected infrastructure sensors, we are given an unparalleled situational awareness of road conditions in real time. Work on making road vehicles connected and autonomous provides new opportunities to improve transportation efficiency and safety on our roads. For example, connected infrastructure sensors can monitor the environment in real time and provide detailed information about the road conditions such as traffic flow, road surface quality, and weather conditions. Connected autonomous vehicles can then use this information to adjust their behavior accordingly to improve safety and efficiency. This can yield many benefits such as reducing congestion, reducing fuel consumption, and reducing the number of accidents. Smart cities also provide the opportunity to experiment with new technologies and approaches to urban transportation, such as platooning of autonomous vehicles.

As we move towards a future with autonomous vehicles and connected infrastructure, there are undoubtedly many opportunities for improved safety and efficiency. However, there are also numerous challenges that need to be addressed. One of the primary challenges is the management and communication of the large amounts of data that will be generated. Effective data-driven methodologies, such as the Learning Model Predictive Control (LMPC) approach used in this study, will be crucial in dealing with this challenge. Another challenge is the gradual rollout of autonomous driving, which will require the coexistence of human-driven and autonomous vehicles of varying levels. Finally, with multiple OEMs and service providers providing both infrastructure and vehicles, data generation and utilization will be a challenge that requires scalable solutions such as edge computing and decentralized algorithms.

To address some of these challenges, we use LMPC to manage three different smart city scenarios of increasing complexity. By leveraging trajectory data from past executions, LMPC can optimize future executions while ensuring safety and recursive feasibility. Additionally, LMPC guarantees a non-decreasing performance from one execution to the next. This data-driven approach provides a promising solution to some of the challenges we face in managing and utilizing the vast amounts of data generated by autonomous vehicles and connected infrastructure.

1.2 Problem formulation

This project utilizes the LMPC framework to find time-optimal trajectories for complex, multi-vehicle scenarios in a smart city environment. To achieve this, the project follows a procedure that involves formulating a smart city scenario by defining the dynamics of the vehicles involved and their initial and final states. An initial feasible trajectory is generated using a safe and conservative algorithm, which is then used to synthesize the terminal components for the LMPC problem corresponding to the scenario. The LMPC training is run over multiple iterations until convergence is achieved, *i.e.,* a locally time-optimal trajectory is found for all vehicles. Finally, the time-optimal trajectory is stored in a database of time-optimal trajectories corresponding to each scenario and variation within the scenario. The database can then be queried for an optimal trajectory when an optimized scenario is encountered in the future. The overall procedure is illustrated in fig. 1.1.

The project focuses on three scenarios of increasing complexity, starting with a simple scenario where a single vehicle is driving on a two-lane road segment with a right turn followed by a left turn, followed by a scenario where an obstacle is added on the main lane mid-way through the road segment, and finally, a scenario with oncoming traffic in a multi-vehicle scenario. The

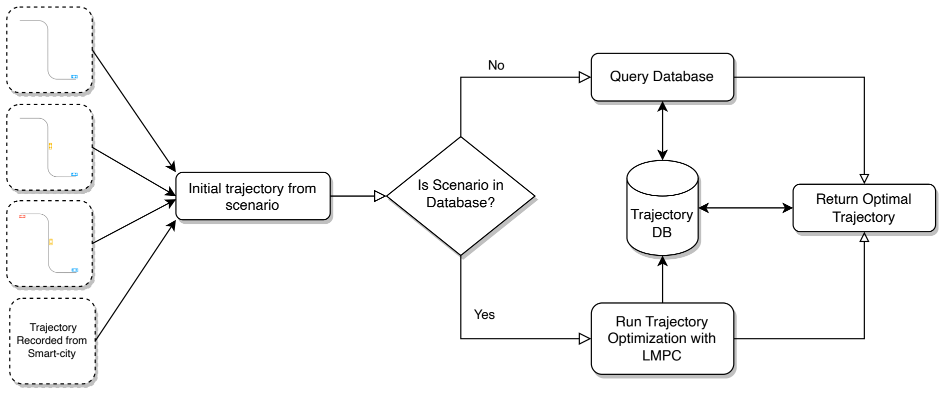

Trajectory Recorded from Smart-city
Initial trajectory from scenario
Is Scenario in Database?
No
Yes
Query Database
Trajectory DB
Run Trajectory Optimization with LMPC
Return Optimal Trajectory

Figure 1.1: Overview of the procedure used to optimize smart-city trajectories.

project also investigates how the converged solution in the LMPC framework depends on the initial feasible trajectory, particularly in the third scenario where the ego vehicle has no option but to meet the oncoming traffic head-on. Ultimately, the project aims to find locally time-optimal trajectories for all vehicles in complex, multi-vehicle scenarios and store them in a database for future use.

1.3 Objective

The goal of this project is to evaluate the performance of Learning Model Predictive Control (LMPC) in three smart city scenarios of increasing complexity. The focus is on improving traffic efficiency by minimizing travel time, as well as assessing LMPC's safety and recursive feasibility. To do this, simulation software and mathematical analysis will be used to study the scenarios and analyze the data. The project will then provide a comprehensive report on LMPC's performance in these three scenarios, which will guide further research into its application in smart cities. Ultimately, our objective is to answer the research question: How do we utilize vehicle and infrastructure data to improve traffic efficiency while guaranteeing safety in smart city scenarios?

1.4 Structure of the thesis

Chapter 2 presents relevant background information about theoretical frameworks used in this work. Chapter 3 presents the methodology and method used to apply the theoretical frameworks on the three smart city scenarios described above. In Chapter 4 the results of the work are presented. In Chapter 5 the results are analysed and discussed. Finally, Chapter 6 presents the concluding remarks and highlights possible future work.

Chapter 2

Background

2.1 Model Predictive Control

In Model Predictive Control (MPC), we solve an open loop optimization problem for a system and apply the solution as an optimal control law. A feedback loop is formed by repeating this procedure every time step. This is known as a receding horizon control strategy. We consider a discrete-time, nonlinear system

$$x_{t+1} = f(x_t, u_t). \tag{2.1}$$

The discrete-time and infinite-horizon optimization problem of the MPC scheme is formulated according to the following

$$J^*_{t\to\infty}(x_t) = \min_{u_{t|t}, u_{t+1|t}, \cdots} \sum_{k=t}^{\infty} h(x_{k|t}, u_{k|t}) \tag{2.2a}$$

$$\text{s.t.} \quad x_{k+1|t} = f(x_{k|t}, u_{k|t}) \quad \forall k \geq t \tag{2.2b}$$

$$x_{k|t} \in \mathcal{X} \quad u_{k|t} \in \mathcal{U} \quad \forall k \geq t \tag{2.2c}$$

$$x_{t|t} = x_t, \tag{2.2d}$$

where (2.2a) represents the optimization objective, (2.2b) the system dynamics constraints, and (2.2c) the state constraints (*e.g.,* collision avoidance) and control constraints (*e.g.,* actuation limits). Equation (2.2d) sets the first step in the optimization horizon to the current state of the system. We assume that the stage cost $h(\cdot, \cdot)$ in (2.2a) is a continuous function which satisfies

$$h(x_F, 0) = 0 \text{ and } h(x_t, u_t) \succ 0 \quad \forall x_t \in \mathbb{R}^n \setminus \{x_F\}, u_t \in \mathbb{R}^m \setminus \{0\}, \tag{2.3}$$

i.e., it is strictly postive and zero at the state x_F which is assumed to be a stable equilibrium for the unforced system (2.1) [1].

Note that in this project, we use nonlinear system dynamics, so our MPC problem becomes a nonlinear MPC, commonly referred to as NMPC in the literature. In the remainder of the report we use MPC and NMPC interchangeably.

In practice, the infinite horizon formulation suffers from two limitations. Firstly, implementing eq. (2.2) in a computer becomes infeasible because it requires infinite computation time. Even for large optimization horizons the problem is hard to solve [1]. Secondly, there might be disturbances to the system or model inaccuracies in the system dynamics $f(x, u)$ which over a long horizon could accumulate to large errors and render the solution ineffective in closed loop. To overcome both limitations, we use a shorter optimization horizon of N steps and solve the following discrete-time problem:

$$J^{MPC}_{t\to t+N}(x_t) = \min_{u_{t|t},\cdots,u_{t+N-1|t}} \sum_{k=t}^{N-1} h(x_{k|t}, u_{k|t}) + V(x_{t+N|t}) \tag{2.4a}$$
$$\text{s.t.} \quad x_{k+1|t} = f(x_{k|t}, u_{k|t}) \tag{2.4b}$$
$$x_{k|t} \in \mathcal{X} \quad u_{k|t} \in \mathcal{U} \tag{2.4c}$$
$$x_{t+N|t} \in \mathcal{X}_N \tag{2.4d}$$
$$x_{t|t} = x_t \tag{2.4e}$$
$$\forall k \in \{t, \dots, t+N-1\}. \tag{2.4f}$$

The predicted state trajectory and optimal control sequence which solve (2.4), at time t, are expressed as

$$\begin{aligned}\mathbf{x}^* &= [x^*_{t|t}, x^*_{t+1|t}, \dots, x^*_{t+N|t}]\\ \mathbf{u}^* &= [u^*_{t|t}, u^*_{t+1|t}, \dots, u^*_{t+N-1|t}].\end{aligned}$$

We define the closed loop feedback policy for MPC as

$$u_t = \pi_t^{MPC}(x_t) = u^*_{t|t}. \tag{2.5}$$

The feedback policy (2.23) is applied at time t and in the next time step, $t+1$, we repeat the procedure and solve (2.4) with the new state of the system in a receding horizon fashion.

2.1.1 Terminal Components

Compared to the infinite horizon problem in (2.2), we notice that the finite time formulation in (2.4) includes the term $V(x_{t+N|t})$ in the cost (2.4a) and an additional state constraint (2.4d). The additional term in the cost function (2.4a) is known as the *terminal cost*, and the additional state constraint (2.4d) is known as the *terminal state constraint*. Together, they are known as the *terminal components* and play a vital role in the MPC scheme.

A key concept in the MPC scheme is known as recursive feasibility, which means that solving the MPC problem successfully at time t guarantees that we can solve the problem for all future times without taking short-sighted action that break the constraints [2]. The terminal state constraint (2.4d) forces the optimization problem to bring the system into a set of states $\mathcal{X}_N$ at the end of the horizon. For a nonlinear system, recursive feasibility is achieved if the set $\mathcal{X}_N$ is a control invariant set [2]. A control invariant set contains all states which have a control input that keeps the system within the set while satisfying all constraints. A formal definition, adopted from [3], is given in 2.1.1.

Definition 2.1.1 (Control Invariant Set) *A control invariant set $\mathcal{C} \subseteq \mathcal{X}$ for the system* (2.4b) *subject to constraints* (2.4c)*, is defined as:*

$$\mathcal{C} = \{\, x_t \in \mathcal{X}_N \mid \exists u_t \in \mathcal{U} \ \textit{such that} \ f(x_t, u_t) \in \mathcal{X}_N \,, \forall t \geq 0 \,\}.$$

The terminal cost, $V(x_{t+N|t})$, contains the cost-to-go for all subsequent steps starting from the terminal state $x_{t+N|t}$. For closed-loop stability of the finite horizon MPC problem (2.4), we want the terminal cost function in (2.4a) to be a control Lyapunov function (defined in 2.1.2 [4], which means that every state, x_t, has a control input, u_t, that takes the system to a state, x_{t+1} with a lower cost along the closed-loop trajectory.

Definition 2.1.2 (Control Lyapunov function) *A control Lyapunov function is a function $V : \mathbb{R}^{\ltimes} \to \mathbb{R}$ for the system* (2.4b) *subject to constraints* (2.4c)*, such that for all $x \in \mathbb{X}$ there exists $u \in \mathcal{U}$ such that*

$$V(f(x,u)) - V(x) \leq -h(x,u)$$

for some positive semi-definite function $h(x,u)$.

2.1.2 MPC Optimality

While the above conditions on the terminal components guarantee recursive feasibility and asymptotic stability, they do not guarantee optimality for the cost $J^{MPC}_{t\rightarrow t+N}(x_t)$ in (2.4a). This becomes evident if we rewrite the cost function of the infinite-horizon MPC problem (2.2) as

$$J^*_{t\rightarrow\infty}(x_t) = \min_{u_{t|t},\ldots,u_{N-1|t}} \sum_{k=t}^{N-1} h(x_{k|t}, u_{k|t}) + J^*_{t+N\rightarrow\infty}(x_{t+N|t}),$$

and introduce the terminal state constraint

$$x_{t+N|t} \in \mathcal{X}_\infty,$$

where $\mathcal{X}_\infty$ is the maximal stabilizable set. In this infinite-horizon formulation we see that the terminal cost is the optimal cost-to-go and the terminal state set is the corresponding maximal stabilizable set, loosely defined as largest set of states which satisfy the control task. Note that $\mathcal{X}_N \subseteq \mathcal{X}_\infty$ *i.e.,* the control invariant, terminal state set in (2.4d) is a subset of the maximal stabilizable set.

In practice, computing the optimal terminal cost $J^*(\cdot)$ and the maximal stabilizable set $\mathcal{X}_\infty$ can be as complex as solving the problem itself. Therefore, we resort to using approximation of the terminal components. Thus, the closer the terminal components in (2.4) approximate $J^*(\cdot)$ and $\mathcal{X}_\infty$, the closer we get to the optimal solution [1].

2.2 Learning Model Predictive Control

We have previously discussed that the finite-time formulation of the MPC problem (2.4) requires the introduction of a terminal cost and terminal state constraint. The Learning Model Predicitve Control (LMPC) scheme provides a data-driven approach to synthesising the terminal components. This is achieved by utilizing historic data from the system to construct control invariant sets and control Lyapunov functions [1].

LMPC is an extension of the MPC scheme applied to systems that perform an iterative task and have no particular reference trajectory. In other words, it is a reference free control scheme. Through repeated iterations the LMPC scheme finds a trajectory for the system such that the performance of the current iteration is at least as good as the performance of previous iterations. Furthermore, the LMPC framework guarantees convergences of the performance [5]. The infinite horizon LMPC problem is formulated as

$$J^{*,j}_{t\to\infty}(x_t) = \min_{u^j_{t|t}, u^j_{t+1|t}, \dots} \sum_{k=t}^{\infty} h(x^j_{k|t}, u^j_{k|t}) \tag{2.6a}$$

$$\text{s.t.} \quad x^j_{k+1|t} = f(x^j_{k|t}, u^j_{k|t}) \quad \forall k \geq t \tag{2.6b}$$

$$x^j_{k|t} \in \mathcal{X} \quad u^j_{k|t} \in \mathcal{U} \quad \forall k \geq t \tag{2.6c}$$

$$x^j_{t|t} = x_t, \tag{2.6d}$$

where j indicates the iteration number, and the stage cost $h(\cdot,\cdot)$ in (2.6a) satisfies the condition in (2.3).

The cost-to-go at time t of the j-th iteration is given by

$$J^j_{t\to\infty}(x^j_t) = \sum_{k=t}^{\infty} h(x^j_k, u^j_k), \tag{2.7}$$

and the overall iteration cost which quantifies the performance of the controller performance is given by

$$J^j_{0\to\infty}(x^j_0) = \sum_{k=0}^{\infty} h(x^j_k, u^j_k), \tag{2.8}$$

where $x^j_0 = x_S$ for all iterations [5].

2.2.1 LMPC Assumptions

The LMPC scheme builds on the assumption that we have an iterative task which starts in a known state x_S and ends in another known state x_F. Successfully taking the system from x_S to x_F counts as one iteration. Furthermore, we assume that every iteration starts and finishes in the same state, in other words

$$x^j_0 = x_S \quad \forall j \in \{0, \dots, J\}$$
$$x^j_{T^j} = x_F \quad \forall j \in \{0, \dots, J\}$$

where j denotes the iteration index and T^j denotes the total number of time-steps in the iteration.

Finally, the LMPC scheme assumes that there exists an initial feasible trajectory, at $j = 0$,

$$\begin{aligned}\mathbf{x}^0 &= [x_0^0, \dots, x_{T^0}^0] \\ \mathbf{u}^0 &= [u_0^0, \dots, u_{T^0-1}^0],\end{aligned}$$

which successfully takes the system from $x_0^0 = x_S$ to $x_{T^0}^0 = x_F$ while satisfying all constraints.

In summary, the LMPC assumptions are:

1. We are solving an iterative task
2. Every iteration has the same starting and finishing states
3. There exists an initial trajectory from a successful iteration

2.2.2 Sampled Safe Set and Value Function

Given the iterative nature of the control problems in focus, we can collect historic data from previous iterations into a set of states which are known to be safe. We know these states to be safe because we assume that they were generated through a successful previous iteration of a deterministic system. Formally, we define the safe set at iteration j as

$$\mathcal{SS}^j = \bigcup_{i \in M^j} \bigcup_{i=t}^{\infty} x_t^i, \tag{2.9}$$

where the set

$$M^j = \{i \in [0, j] : \lim_{t \to \infty} x_t^i = x_F\}, \tag{2.10}$$

contains all indices which correspond to a successful iteration [5], see fig. 2.1. We see that by definition $M^i \subseteq M^j,\ \forall i \leq j$, and thus, $\mathcal{SS}^i \subseteq \mathcal{SS}^j,\ \forall i \leq j$. Furthermore, we see that $\mathcal{SS}^j \subseteq \mathcal{X}_\infty$, *i.e.,* the sampled safe set is a subset of the maximal stabilizable set since for every state in $\mathcal{SS}^j$ there exists a control input which satisfies the constraints and takes the system to the equilibrium state x_F [1].

Since the sampled safe set $\mathcal{SS}^j$ collects the states of all successful past trajectory, we define a value function which assigns the cost-to-go for each point in $\mathcal{SS}^j$

$$V^j(x) = \begin{cases} \min_{(i,t) \in F^j(x)} J_{t \to \infty}^i(x), & \text{if } x \in \mathcal{SS}^j \\ \infty & \text{if } x \notin \mathcal{SS}^j \end{cases}, \tag{2.11}$$

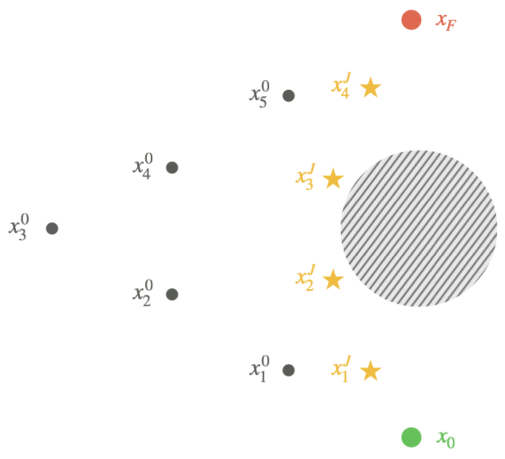


Figure 2.1: An illustration of the sampled safe set consisting of the initial trajectory 0 and the converged trajectory J.

where $J^i_{t\to\infty}(\cdot)$ is the cost-to-go defined in (2.7) and the function

$$F^j(x) = \left\{(i,t) : i \in [0,j], t \geq 0 \text{ with } x^i_t = x \text{ for } x^i_t \in \mathcal{SS}^j\right\},$$

collects all index pairs corresponding to the iteration numbers and time steps of a particular state x inside SS^j. We see that the value function in (2.11) computes the minimum cost-to-go for every point in $\mathcal{SS}^j$, *i.e.*, it gives us the best possible performance for every state we have visited in past trajectories [5].

2.2.3 Minimum Time Safe Set and Value Function

If we have a control problem with the objective of steering the system from x_S to x_F in minimum time, we can express the stage cost as

$$h(x^j_t, u^j_t) = \mathbb{1}_{x_F}(x^j_t) = \begin{cases} 1 & \text{if } x^j_t \neq x_F \\ 0 & \text{else} \end{cases}. \tag{2.12}$$

Taking this into account, we define the optimal trajectory time as

$$T^{j,*} = \min_{i\in[0,\dots,j]} T^i,$$

and construct the time-varying sampled safe-sets

$$\mathcal{SS}_t^j = \bigcup_{i=0}^{j} \bigcup_{k=\delta_t^{j,i}}^{T^i} x_k^i, \tag{2.13}$$

where

$$\delta_t^{j,i} = \min(T^i - T^{j,*} + t, T^i). \tag{2.14}$$

The value function which approximates the cost-to-go for each state $\mathcal{SS}_t^j$ becomes

$$V_t^j(x) = \min_{\substack{i\in[0,\dots,j] \\ k\in[\delta_t^{j,i},\dots,T^i]}} J_{t\to T^i}^i(x_k^i), \tag{2.15}$$
$$\text{s.t.} \quad x = x_k^i \in \mathcal{SS}_t^j$$

where $J_{t\to T^j}^j(\cdot)$ is the cost-to-go given by

$$J_{t\to T^j}^j(x_t^j) = \sum_{k=t}^{T^j} \mathbb{1}_{x_F}(x_k^j), \tag{2.16}$$

with the indicator function $\mathbb{1}_{x_F}(\cdot)$ defined in (2.12).

At time t, the time varying safe set $\mathcal{SS}_t^j$ collects all states from which the system can reach the target in at most $(T^{j,*} - t)$ time steps. By construction, the time-varying sampled safe set $\mathcal{SS}_t^j$ is a control invariant set since we know that at time t for each $x_t^j \in \mathcal{SS}_t^j$ there exists a control input $u_t^j \in \mathcal{U}$ which keeps the trajectory of the system (2.1) within the time-varying safe set at time $t+1$, *i.e.*, $f(x_t^j, u_t^j) \in \mathcal{SS}_{t+1}^j$ [6].

2.2.4 Convexified Safe Set and Value Function

The general sampled safe set $\mathcal{SS}^j$ and the time-varying sampled safe $\mathcal{SS}_t^j$ are both sets of discrete points. Consequently, the value function approximations $V^j(x)$ and $V_t^j(x)$ are also discrete functions over the safe sets $\mathcal{SS}^j$ and $\mathcal{SS}_t^j$, respectively. In some cases, it is desirable to express the sampled safe set as a convex set and this can be achieved by constructing the convex hull of the set.

For the general sampled safe set $\mathcal{SS}^j$, the convex hull is given by

$$\mathcal{CS}^j = \left\{ \sum_{t=1}^{|\mathcal{SS}^j|} \lambda_t x_t : \lambda_t \geq 0, \sum_{t=1}^{|\mathcal{SS}^j|} \lambda_t = 1, x_t \in \mathcal{SS}^j \right\}, \tag{2.17}$$

where $|\mathcal{SS}^j|$ is the cardinality of $\mathcal{SS}^j$. With the assumption that the state and input constraints sets, $\mathcal{X}$ and $\mathcal{U}$ respectively, are also convex, we see that $\mathcal{CS}^j$ is a control invariant set [7]. Furthermore, we define the convex value function

$$\begin{aligned} V(x)^j = \min_{\lambda_t^j \geq 0, \forall t \in [0,\infty)} & \sum_{i=0}^{j} \sum_{t=0}^{\infty} \lambda_t^i J_{t\to\infty}^i(x_t^i) \\ \text{s.t.} & \sum_{i=0}^{j} \sum_{t=0}^{\infty} \lambda_t^i = 1 \\ & \sum_{i=0}^{j} \sum_{t=0}^{\infty} \lambda_t^i x_t^i = x, \end{aligned} \tag{2.18}$$

where $x_t^i \in \mathcal{SS}^j$ and $J_{0\to\infty}^j(\cdot)$ is the cost-to-go defined in (2.7). The convex value function $V(\cdot)^j$ in (2.18) constitutes a piecewise-affine interpolation of (2.11) over the convex safe set $\mathcal{CS}^j$ [7].

For the minimum time case, we construct a time-varying convex hull of $\mathcal{SS}_t^j$

$$\mathcal{CS}_t^j = \left\{ \sum_{i=0}^{j} \sum_{k=\delta_t^{j,i}}^{T^i} \lambda_k^i x_k^i : [\lambda_{\delta_t^{j,i}}^0, \dots, \lambda_{T^j}^j] \geq 0, \sum_{i=0}^{j} \sum_{k=\delta_t^{j,i}}^{T^i} \lambda_k^i = 1, x_k^i \in \mathcal{SS}_t^j \right\}, \tag{2.19}$$

where $\delta_t^{j,i}$ is defined in (2.14), and the piecewise-value function interpolation of (2.15) is given by

$$\begin{aligned}
\overline{V}_t(x)^j = \min_{[\lambda^0_{\delta^{j,0}_t},\ldots,\lambda^j_{T^j}]\geq 0} & \sum_{i=0}^{j}\sum_{k=\delta^{j,i}_t}^{T^i} \lambda^i_k J^i_{k\to T^i}(x^i_t) \\
\text{s.t.} & \sum_{i=0}^{j}\sum_{k=\delta^{j,i}_t}^{T^i} \lambda^i_k = 1 \\
& \sum_{i=0}^{j}\sum_{k=\delta^{j,i}_t}^{T^i} \lambda^i_k x^i_k = x,
\end{aligned} \tag{2.20}$$

where $x^j_t \in \mathcal{SS}^j_t$ and $J^i_{t\to T^i}(\cdot)$ is the cost-to-go defined in (2.16) [6].

2.2.5 LMPC Mixed-Integer Formulation

The core motivation behind the LMPC framework is that we use historic data from previous trajectories to approximate the terminal components in the finite time LMPC for discrete systems. Put together, we get the following optimization problem

$$J^{\text{LMPC},j}_{t\to t+N}(x^j_t) = \min_{u^j_{t|t},\ldots,u^j_{t+N-1|t}} \sum_{k=t}^{N-1} h(x^j_{k|t}, u^j_{k|t}) + V^{j-1}(x^j_{t+N|t}) \tag{2.21a}$$

$$\text{s.t.} \quad x^j_{k+1|t} = f(x^j_{k|t}, u^j_{k|t}) \tag{2.21b}$$

$$x^j_{k|t} \in \mathcal{X} \quad u^j_{k|t} \in \mathcal{U} \tag{2.21c}$$

$$x^j_{t+N|t} \in \mathcal{SS}^j \tag{2.21d}$$

$$x^j_{t|t} = x^j_t \tag{2.21e}$$

$$\forall k \in \{t, \ldots, t+N-1\}, \tag{2.21f}$$

where j indicates the iteration number, $\mathcal{SS}^j$ is defined in (2.9) and $V^{j-1}(\cdot)$ is the cost-to-go presented in (2.11). When comparing the LMPC formulation (2.21) with the finite time MPC problem in (2.4), we see that the terminal state constraint $\mathcal{X}_N$ is replaced with the sampled safe $\mathcal{SS}^j$ and the terminal cost is replaced with value function $V^j(\cdot)$ which assigns the minimum cost-to-go for each state in the sampled safe set.

For the minimum time case where the stage cost is the indicator function, defined in (2.12), we replace the sampled safe set $\mathcal{SS}^j$ and the value function approximation $V(\cdot)^j$ with their time-varying counterparts $\mathcal{SS}^j_t$ in (2.13) and

$V(\cdot)_t^j$ in (2.15), respectively.

2.2.6 LMPC Relaxed Formulation

Due to the nature of the sampled safe set (2.21d), the LMPC problem (2.21) becomes a mixed-integer problem, since we need to perform the optimization for each discrete point in $\mathcal{SS}^j$ and the discrete value function $V^j(\cdot)$. To avoid this, it is possible to relax the terminal state constraint of the LMPC formulation by replacing the sampled safe set $\mathcal{SS}^j$ with its convex hull $\mathcal{CS}^j$ in (2.17) and replacing the terminal cost with its affine interpolation $\overline{V}^j(\cdot)$ in (2.18) [7]. Put together, we get the following relaxed optimization problem over a convex set of points

$$\begin{aligned}
J^{\text{LMPC},j}_{t\to t+N}(x_t^j) = \min_{\boldsymbol{u}^j,\boldsymbol{\lambda}^j} \sum_{k=t}^{N-1} h(x^j_{k|t}, u^j_{k|t}) + \sum_{i=0}^{j-1}\sum_{k=0}^{T^i} \lambda_k^i J^i_{k\to T^i} x_k^i \qquad &(2.22a)\\
\text{s.t.} \quad x^j_{k+1|t} = f(x^j_{k|t}, u^j_{k|t}) \qquad &(2.22b)\\
x^j_{k|t} \in \mathcal{X} \quad u^j_{k|t} \in \mathcal{U} \qquad &(2.22c)\\
\sum_{i=0}^{j-1}\sum_{k=0}^{T^i} \lambda_k^i x_k^i = x^j_{t+N|t} \qquad &(2.22d)\\
\sum_{i=0}^{j-1}\sum_{k=0}^{T^i} \lambda_k^i \lambda_k^i = 1 \qquad &(2.22e)\\
x^j_{t|t} = x^j_t \qquad &(2.22f)\\
\forall k \in \{t, \dots, t+N-1\}, \qquad &(2.22g)
\end{aligned}$$

For the minimum time case, where the stage cost $h(\cdot,\cdot)$ is the indicator function (2.12), we use the time-varying convex hull $\mathcal{CS}_t^j$ in (2.19) and the time-varying value function approximation $\overline{V}_t^j(\cdot)$ in (2.20) [6].

2.2.7 Properties of LMPC

At iteration j and time t, the the optimal solution to the LMPC problems presented in (2.21), (2.22) are expressed as

$$\begin{aligned}
\mathbf{x}^{j,*} &= [x^{j,*}_{t|t}, x^{j,*}_{t+1|t}, \dots, x^{j,*}_{t+N|t}]\\
\mathbf{u}^{j,*} &= [u^{j,*}_{t|t}, u^{j,*}_{t+1|t}, \dots, u^{j,*}_{t+N-1|t}],
\end{aligned}$$

and we define the closed loop feedback policy for LMPC as

$$u_t^j = \pi_t^{\text{LMPC},j}(x_t) = u_{t|t}^{j,*}. \tag{2.23}$$

We apply this policy to the system (2.1) at time t and get the next state $x_{t+1}^j = f(x_t^j, u_t^j)$ from which we repeat the optimization step, and so on. At the end of the iteration, when the system reaches the goal state x_F, we collect all the states that were actualized by the system into the sampled safe set and compute the corresponding minimum cost-to-go. The overall performance of the iteration is computed using (2.8).

We have already argued that the safe set $\mathcal{SS}^j$ and $\mathcal{SS}_t^j$ are control invariant. In [5] we find proof that the optimal cost $J_{t\to t+N}^{\text{LMPC,j}}(\cdot)$ is indeed a control Lyapunov function, which makes the sampled safe set and the value function approximations, $V^j(\cdot)$ and $V_t^j(\cdot)$, valid terminal components which yield recursive feasibility and closed loop stability. Furthermore, given the assumptions presented in section 2.2.1 we have that over each iteration, the LMPC framework guarantees:

1. Recursive feasibility
2. Finite-time closed-loop convergence
3. Non-decreasing performance between iterations.

The first and second guarantees imply that the system will always be able to reach the target state x_F without risking any short-sighted actions that would lead to infeasibilities. The third guarantee means that the iterations cost for iteration j is at least as good as the iteration cost of iteration $j-1$, *i.e.,* $J_{0\to\infty}^{\text{LMPC},j} \geq J_{0\to\infty}^{\text{LMPC},j-1}$. We refer the reader to [5] for a rigours proof of the above guarantees in the general case. For proofs taking into account the the minimum time and relaxed constraint cases, we refer the reader to [7], [6], and [1].

We end this section by highlighting that for the relaxed, minimum time-case, *i.e.,* (2.22) with the indicator function (2.12) as stage cost and time-varying terminal components, we have two additional assumptions for the above guarantees to hold:

1. $\mathcal{X}$ and $\mathcal{U}$ in (2.22c) are convex
2. Considering the convex safe set $\mathcal{CS}_t^j$ constructed using the stored closed-loop trajectories for $i \in \{0, \ldots, j\}$. Then, for all $k \in \{1, \ldots, n+1\}$,

$x^{(k)} \in \{\cup_{i=0}^{j} \cup_{t=0}^{T^i} x_t^i\}$ and $x \in \text{Conv}(\cup_{k=1}^{n+1} x^{(k)})$, we have that there exists an input $u \in \mathcal{U}$ such that

$$f(x,u) \in \text{Conv}\left(\bigcup_{k=1}^{n+1} f(x^{(k)}, u^{(k)})\right),$$

where $u^{(k)}$ is the stored input applied at the store state $x^{(k)} \in \{\cup_{i=0}^{j} \cup_{t=0}^{T^i} x_t^i\}$

The second assumption is adopted from Assumption 4 in [6]. This assumption is in general difficult to confirm analytically and requires empirical analysis for each system. We find an analysis for a system similar to the one used in this project in [6].

2.3 Multi-agent LMPC

So far, in the system equation (2.1) we have only considered a a single-agent system with nonlinear dynamics. In this section we extend the LMPC formulation to the multi-agent case. We introduce a new index $i \in \mathcal{M} = \{1, \dots, M\}$ which assigns a unique number to each agent for a system of M agents. Like the single-agent case, we assume that an initial feasible trajectory exists for all agents and that the starting and goal states are the same for all future iteration, *i.e.*, $x_{i,0}^j = x_{i,S}$ and $x_{i,\infty}^j = x_{i,F}$ $\forall i \in \mathcal{M}$. The global state vector of the system is formed by stacking the state vector of each agent. At time t and iteration j we get $x_t^j = [x_{1,t}^{j,T}, \dots, x_{M,t}^{j,T}]^T \in \mathbb{R}^n$, where $x_{i,t}^j \in \mathbb{R}^{n_i}$ $\forall i \in \mathcal{M}$. Similarly, the control input for the global system at time t and iteration j is given by $u_t^j = [u_{1,t}^{j,T}, \dots, u_{M,t}^{j,T}]^T \in \mathbb{R}^m$ where $u_{i,t} \in \mathbb{R}^{m_i}$ $\forall i \in \mathcal{M}$. The global system dynamics are described by (2.1) using the stacked state and control vectors. By assuming that the system dynamics of the agents are decoupled, the system dynamics for the global system (2.1) can also be written as $x_{i,t+1} = f_i(x_{i,t}, u_{i,t})$ for each agent $i \in \mathcal{M}$.

2.3.1 Centralized, Decentralized, and Distributed LMPC

In the field of control systems, there are three main approaches to managing groups of agents: centralized, decentralized, and distributed. The centralized approach relies on a single control system to manage all the agents in the group. In contrast, the decentralized approach delegates control to each individual agent, allowing them to make decisions based on their own local information.

Finally, the distributed approach involves agents communicating with each other to coordinate their actions [8].

Each approach has its own set of advantages and disadvantages. For example, centralized LMPC has the advantage of being able to leverage the global information of all the agents to make optimal control decisions. However, it can quickly become computationally intractable and communication-heavy as the number of agents increases. In contrast, decentralized LMPC allows each agent to independently generate control decisions based on its own local information. This reduces the communication load and allows agents to work together in parallel, leading to improved efficiency. However, decentralized LMPC may not always guarantee optimal performance, especially in complex systems with significant interactions between agents. Distributed LMPC strikes a balance between the two by allowing agents to communicate and coordinate with each other while still maintaining some level of decentralization. This approach can work well in systems with moderate complexity, but can become difficult to manage in larger systems with a higher number of agents [9], [10].

In this project, we are focusing on the centralized approach for our smart city application since it allows us to make optimal control decisions by leveraging global information. This is particularly useful in simulations where we can coordinate the actions of a relatively small number of agents for improved performance and efficiency.

2.3.2 Centralized Multi-agent LMPC Formulation

The overall control objective is to design a controller that drives the global system from an initial state $x_S = [x_{1,S}^T, \ldots, x_{M,S}^T]^T$ to a target state $x_F = [x_{1,F}^T, \ldots, x_{M,F}^T]^T$. In the centralized case, where all computation is done on the same machine, we get the following formulation

$$
\begin{aligned}
J^{\text{LMPC},j}_{t\to t+N}(x^j_t) = \min_{u^j_{t|t},\dots,u^j_{t+N-1|t}} & \sum_{k=t}^{N-1} h(x^j_{k|t}, u^j_{k|t}) + V^{j-1}(x^j_{t+N|t}) && (2.24a)\\
\text{s.t.} \quad & x^j_{k+1|t} = f(x^j_{k|t}, u^j_{k|t}) && (2.24b)\\
& x^j_{k|t} \in \mathcal{X} \quad u^j_{k|t} \in \mathcal{U} && (2.24c)\\
& x^j_{t+N|t} \in \mathcal{SS}^j && (2.24d)\\
& g(x^j_{k|t}) \preccurlyeq 0 && (2.24e)\\
& x^j_{t|t} = x^j_t && (2.24f)\\
& \forall k \in \{t, \dots, t+N-1\}, && (2.24g)
\end{aligned}
$$

where $\preccurlyeq$ indicates the element-wise inequality and the state and input constraints in (2.24c) are the Cartesian products $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_M = \{(x_1, \dots, x_M) : x_i \in \mathcal{X}_i \in \mathbb{R}^{n_i} \ \forall i \in \mathcal{M}\}$ and $\mathcal{U} = \mathcal{U}_i \times \cdots \times \mathcal{U}_M = \{(u_1, \dots, u_M) : u_i \in \mathcal{U}_i \in \mathbb{R}^{m_i} \ \forall i \in \mathcal{M}\}$. The sets $\mathcal{X}_i$ and $\mathcal{U}_i$ are assumed to be closed, compact and contain the target state $x_{i,F}$ in their interior [9].

The multi-agent formulation (2.24) is almost identical to the LMPC problem presented in (2.21) with the additional global constraint $g(\cdot) \preceq 0$ which impose, for instance, safety criteria and collision avoidance constraints between the agents. In the decentralized case, the problem is deconstructed such that each agent is able to solve the problem independently during an iteration. After every iteration, the collected data is used to a synthesize the terminal components and a time-varying deconstruction of the global coupling constraint (2.24e) for the local LMPC problem of each agent. However, the decentralized implementation is outside the scope of the work and we refer the reader to [9] and [10] for further reading. In [9] it is shown that the multi-agent LMPC formulation (2.24) maintains the properties presented in section 2.2.7.

2.4 Related work

In this section we overview some of the work related to the project. We remind the reader that the aim of the project is to utilize the LMPC framework for finding time optimal paths in complex, multi-vehicle, smart city scenarios. As far as we are aware, the LMPC framework as presented in the previous chapter has not been used for any smart city scenarios yet. In the literature, we find several uses for LMPC in single vehicle racing in for instance [11],

[12], and [13]. In [14], we find the LMPC framework applied to minimum time trajectories for a quadrotor.

The LMPC framework is distinct from other techniques such as dynamic programming optimization [15], multi-agent motion planning (see [16], [17]), and reinforcement learning (see [18], [19], and [20] for traffic related literature) in that it is a data-driven method which converges in few iterations and inherently guarantees that safety constraints are fulfilled by construction. This makes it particularly useful for safety-critical systems in smart cities. In contrast to LMPC, reinforcement learning applied to traffic scenarios requires thousands of episodes to converge to a good policy ([18], [19], [20]). However, the LMPC framework might suffer for tasks that take a long time to complete. In that case the sampled safe sets would grow very large between iteration and increasing the optimization horizon might speed up the learning but it would slow down the solution for each time step [6].

Chapter 3

Method

3.1 System and Scenario Overview

The aim of the project is to utilize the LMPC framework for finding time optimal trajectories in complex, multi-vehicle, smart city scenarios. Since LMPC is a reference free control scheme, it lends itself well for tasks where the optimal trajectory is not easy to compute for a global system with multiple agents; each having their own constraints as well as coupling constraints between the agents.

Overall, the procedure used in this work begins by formulating a smart city scenario, for example an intersection of three vehicles. Formulating the scenario entails defining the dynamics of the vehicles involved as well as the initial and final states for each vehicle. Then, we generate an initial feasible trajectory by taking the system from the initial state to the target state, in simulation, using a safe and conservative algorithm. We use the initial trajectory to synthesize the terminal components for the LMPC problem corresponding to the scenario. Then, we run the LMPC training over multiple iterations until convergence is achieved, *i.e.,* we have found the locally time optimal trajectory for all vehicles. Finally, we store the time optimal trajectory in a database of time optimal trajectories corresponding to each scenario and variation within the scenario. This procedure is depicted in more detail in fig. 3.1.

In this project, we split the study into three different scenarios of increasing complexity (see fig. 3.2):

1. **Single Vehicle:** We begin with a simple scenario where a single vehicle (the “Ego” vehicle) is driving on a two-lane road segment with a right turn followed by a left turn.

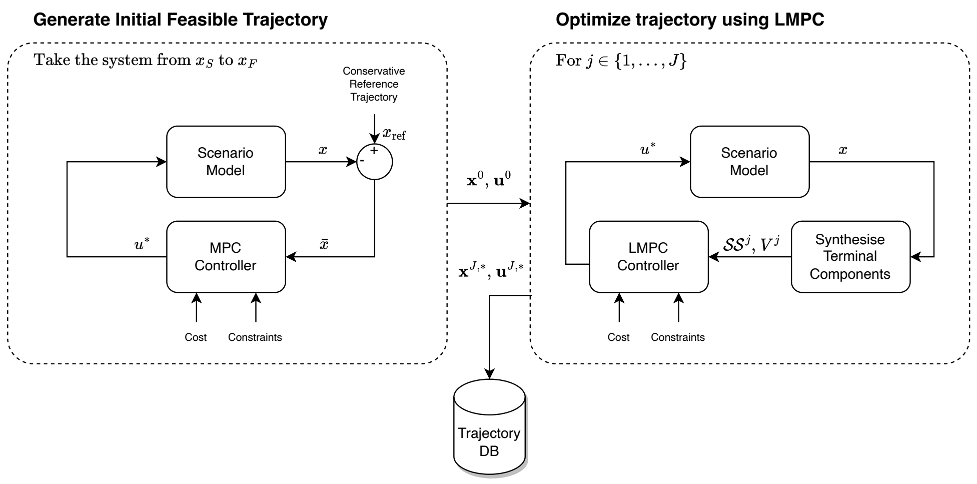

Generate Initial Feasible Trajectory
Take the system from x_S to x_F
Conservative Reference Trajectory
x_{ref}
Scenario Model
x
+
-
u^*
MPC Controller
$\bar{x}$
Cost
Constraints
$\mathbf{x}^0, \mathbf{u}^0$
$\mathbf{x}^{J,*}, \mathbf{u}^{J,*}$
Trajectory DB
Optimize trajectory using LMPC
For $j \in \{1, \ldots, J\}$
u^*
Scenario Model
x
LMPC Controller
$\mathcal{SS}^j, V^j$
Synthesise Terminal Components
Cost
Constraints

Figure 3.1: Detailed overview of the data-driven approach used to optimize smart-city trajectories.

2. **Single Vehicle with Obstacle:** In the second scenario, we add an obstacle (the "Obs" vehicle), resembling a broken down vehicle, on the main lane mid-way through the road segment.

3. **Oncoming traffic:** The last scenario introduces oncoming traffic (the "Onc" vehicle) in a multi-vehicle scenario.

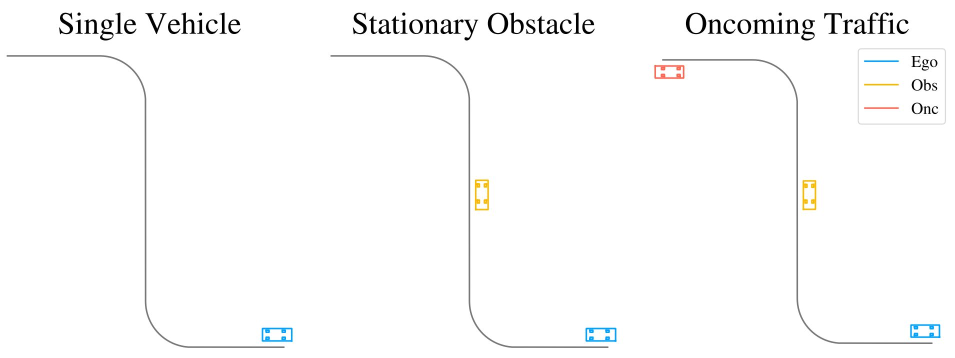


Figure 3.2: The three different scenarios considered in this project.

Since we are using a minimum time objective in the LMPC formulations, our hypothesis is that the ego vehicle should behave like a racing car upon convergence. This is particularly true for the first and second scenarios. However, in the third scenario the ego vehicle has no option but to meet the oncoming traffic head on. This gives us the opportunity to investigate how the converged solution in the LMPC framework depends on the initial feasible trajectory. In essence, we can construct a scenario where the ego vehicle initially can either wait for the oncoming vehicle to pass or to over take the obstacle before the oncoming vehicle passes. As we have discussed before, the terminal components constructed in the LMPC framework are mere approximations of the complete sets. Therefore, the converged trajectories might not be the global optimum and this would partially depend on how the initial feasible trajectory looks like.

3.2 Road Model

We use a well-known approach in tracking control theory known as the Frenet frame method to model the road. We follow the modelling approach used in [21] and [22]. The Frenet frame describes the movement of a point-mass object along a differentiable curve in $\mathbb{R}^2$. The coordinates used in the Frenet frame consists of the tangential direction $\hat{s}$ along the curve and a normal direction $\hat{e}$ pointing to the left in the positive direction. Thus, the coordinate s denotes the distance travelled along the curve, while the coordinate e denotes the deviation from the center line, see fig. 3.3.

At a high-level, the road is constructed from a combination of K curve segments. Each curve segment consists of a circle arc (indexed $k \in \mathcal{K} = \{1, \dots, K\}$) with a particular curvature κ_k and a length L_k or coverage angle θ_k. We have the following relation between the curvature κ and radius r of a circle arc

$$\kappa(s) = \frac{1}{r(s)}, \tag{3.1}$$

where s denotes the dependence on the distance along the curve. Thus, a circle with zero radius gives an infinite curvature and a circle with an infinite radius gives zero curvature which makes up a straight line. The sign of the curvature determines whether the circle arc points in the positive or negative $\hat{e}$-direction, *i.e.,* it determines whether it is a left or a right turn.

In practice, we construct the road segments from a set of discrete points. Suppose that the road length is L meters and the desired point density is ρ points/m. Then, the distance between two consecutive points is $\Delta_L = \frac{L}{\rho L} = \frac{1}{\rho}$. Assuming that each curve segment consists of a circle arc with a known curvature, we get the following relations between two consecutive points on

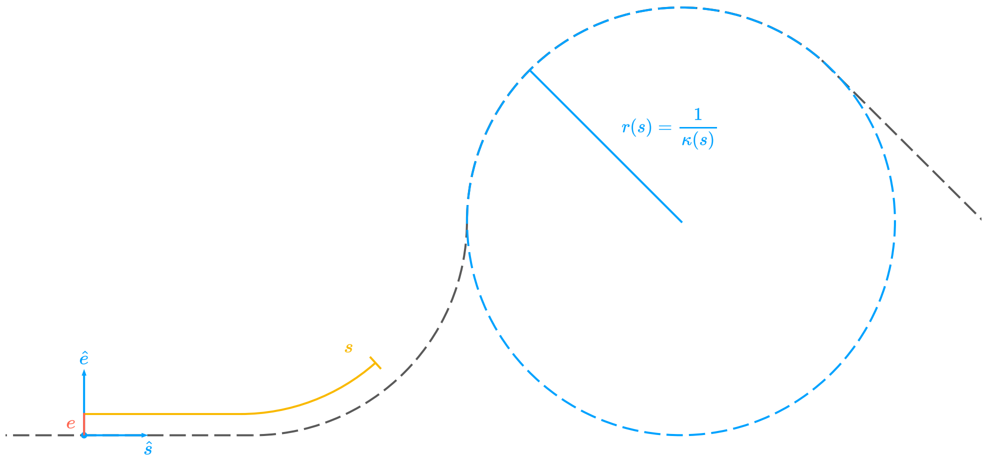


Figure 3.3: Representation of the coordinate system in the Frenet frame along the curve s

the k-th arc

$$s_{i+1} = s_i + \Delta_L \tag{3.2}$$

$$x_{s_{i+1}} = x_{s_i} - \Delta_L \cos(\pi - \phi_{s_i}) \tag{3.3}$$

$$y_{s_{i+1}} = y_{s_i} - \Delta_L \sin(\pi - \phi_{s_i}) \tag{3.4}$$

$$\phi_{s_{i+1}} = (\phi_{s_i} + \Delta_{\theta_k}) \pmod{2\pi}, \tag{3.5}$$

where $\Delta_{\theta_k} = \Delta_L \kappa_k$. Each point encodes its distance in the Frenet Frame and the position and tangent values in the Cartesian frames, *i.e.*, $(s_i, x_{s_i}, y_{s_i}, \phi_{s_i})$. We initialise the construction process with $(0, x_{s_0}, y_{s_0}, \phi_{s_0})$, where x_{s_0}, y_{s_0}, and ϕ_{s_0} denote the desired starting positions and heading of the road in the Cartesian frame. This process is repeated for each arc, where the last point of the previous arc becomes the starting point of the next arc. To transform a point (s, e) in the Frenet to the point (x, y) in the Cartesian frame, we have the relations

$$i = \arg\min_i |s - s_i| \tag{3.6}$$

$$x = x_{s_i} - e \sin(\phi_{s_i}) \tag{3.7}$$

$$y = y_{s_i} + e \cos(\phi_{s_i}). \tag{3.8}$$

In the opposite direction, we transform the point (x, y) in the Cartesian frame to the point (s, e) in the Frenet frame, through the relations

$$i = \arg\min_{i} ||[x, y]^T - [x_{s_i}, y_{s_i}]^T||_2 \tag{3.9}$$

$$s = s_i \tag{3.10}$$

$$e = \begin{cases} \frac{(y - y_{s_i})}{\cos(\phi_{s_i})} & \text{if } \phi_{s_i} = 0 \\ \frac{(x_{s_i} - x)}{\sin(\phi_{s_i})} & \text{else.} \end{cases} \tag{3.11}$$

The road used for all scenarios in this projects represents a two-lane street consisting of five arcs with the parameters detailed in table 3.1. In fig. 3.4 we see the road represented in the Cartesian and Frenet frames side-by-side. Figure 3.4 demonstrates how complicated curved roads in the Cartesian frame become simple rectangular shapes in the Frenet frame. This is particularly useful when modelling vehicles in a smart city scenario since it allows us to express the state constraints for the vehicles as convex box constraints. We assume a lane width of 0.5 meters since the vehicles in this project are modelled after the 1/10 scale vehicle platform presented in [23].

Table 3.1: Arc parameters for the road used in the project

Arc index k	**Curvature** κ_k	**Length** L_k	**Angle** θ_k
1	0	1.85	-
2	$\frac{-1}{0.65\sqrt{2}}$	-	$\pi/2$
3	0	4	-
4	$\frac{1}{0.65\sqrt{2}}$	-	$\pi/2$
5	0	1.85	-

3.3 Vehicle Model

Our simulated vehicles are modelled after the 1/10 scale car known as the Small Vehicle for Autonomy (SVEA) test-bed, presented in [23]. We take advantage of the kinematic bicycle model to describe the motion of the vehicle. The dynamic bicycle model (see [24]) is not used in this project because we assume that the vehicle will not be driven at its dynamical limits and can therefore neglect the tire forces in both lateral and longitudinal directions. Furthermore, we are only considering planar scenarios and omit the pitch state of the vehicle caused by ground inclinations. We find further evidence

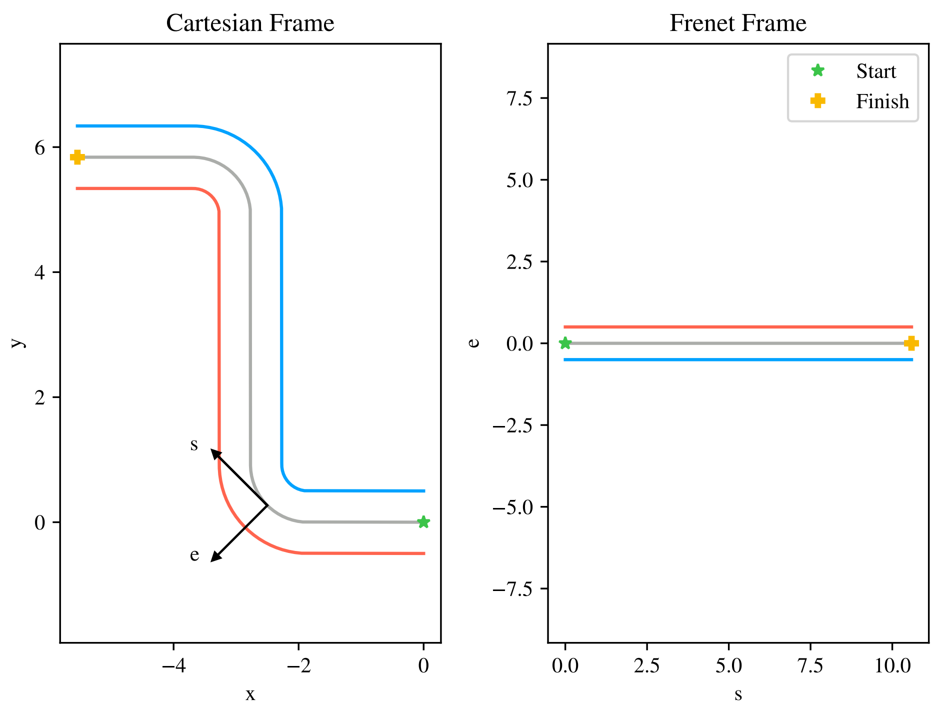


Figure 3.4: Representation of the Cartesian and Frenet frames side-by-side.

confirming the justification to use the kinematic bicycle model, for our case, in the following analysis [25] which compares the two bicycle models. Note that despite using a minimum time objective function, we will add a constraint limiting the velocity to $0.7\ \mathrm{m\,s^{-1}}$ which represent the assumed speed limit. This speed is well below the maximum velocity of the SVEA car, which at low gear is approximately $1.7\ \mathrm{m\,s^{-1}}$.

3.3.1 Kinematic Bicycle Model in the Cartesian Frame

We derive the bicycle model in the Cartesian frame using a similar approach to [26]. However, in our derivation, we place the point of reference at the center of the rear axle. In the XY-plane, the kinematic bicycle model is described by the state vector

$$\overline{x} = [x, y, v, \phi]^T \tag{3.12}$$

where x and y denote the positions in the XY-plane, while v and ϕ denotes the longitudinal velocity and heading angle, respectively. We can see a representation of these quantities in fig. 3.5. The equations describing the motion of the kinematic model are given by

$$\dot{x} = v\cos(\phi) \tag{3.13}$$
$$\dot{y} = v\sin(\phi) \tag{3.14}$$
$$\dot{v} = \frac{v_u - v}{\tau} \tag{3.15}$$
$$\dot{\phi} = \frac{v\tan\delta}{l_{wb}}, \tag{3.16}$$

where l_{wb} denotes the wheelbase and δ denotes the steering angle. The SVEA cars take the velocity v_u as input, instead of the acceleration a as in [26]. Therefore, we introduce the gain τ and compute $\frac{v_u - v}{\tau}$ to emulate the acceleration dynamics of the electrical speed controller on the vehicle. This introduces an additional layer of model mismatch. However, in practice, this approach is found to be sufficient.

3.3.2 Kinematic Bicycle Model in the Frenet Frame

The model used in the project is the kinematic bicycle model expressed in the Frenet frame. In the Frenet frame, the xy-positions of the vehicle are replaced by the s and e coordinates which denote the distance traveled along the road and the lateral deviation from the center line, respectively. To express the kinematic bicycle model in the Frenet frame, we follow the transformation approach described in [21].

The state of the car in the Frenet frame is illustrated in fig. 3.5. We define the term

$$e_\phi = \phi - \phi_s(s), \tag{3.17}$$

which computes the deviation of the car heading from the curve tangent. The curve tangent is itself a function of the distance travelled, s. Furthermore, we have that the tangential component of the vehicles longitudinal velocity is

$$v_s = v\cos(e_\phi) = (r(s) - e)\dot{\phi}_s(s) \overset{\text{eq. (3.1)}}{=} (1/\kappa(s) - e)\dot{\phi}_s(s). \tag{3.18}$$

From (3.18), we can factor out the rate of change of the curve tangent

$$\dot{\phi}_s(s) = \kappa(s)\frac{v\cos(e_\phi)}{1 - \kappa(s)e} \tag{3.19}$$

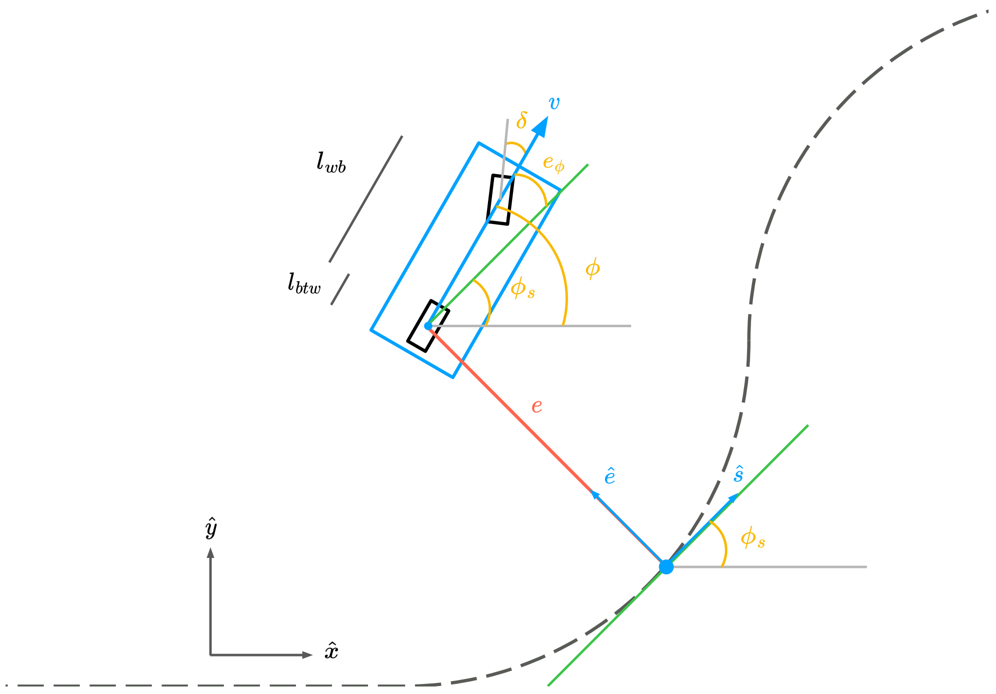


Figure 3.5: Kinematic Bicycle model in the Cartesian and Frenet frames. l_{wb} represents the wheelbase, and l_{btw} is the distance from the rear-end to the rear-axle.

From fig. 3.5, we can deduce the following relation

$$\dot{s} = r(s)\dot{\phi}_s(s), \tag{3.20}$$

which combined with (3.1) and (3.19) gives us

$$\dot{s} = \frac{v\cos(e_\phi)}{1 - \kappa(s)e}. \tag{3.21}$$

We also have that

$$\dot{e} = v\sin(e_\phi), \tag{3.22}$$

which can be derived through inspection of fig. 3.5 or from the relations presented in [21]. Lastly, we differentiate (3.17) and plug in (3.16) as well as (3.19) which gives us

$$\dot{e_\phi} = \dot{\phi} - \dot{\phi}_s(s) = \frac{v \tan \delta}{l_{wb}} - \kappa(s)\frac{v\cos(e_\phi)}{1-\kappa(s)e}. \tag{3.23}$$

Combining the above derivation gives us the state vector in the Frenet frame

$$x = [s, e, v, \phi, e_\phi], \tag{3.24}$$

where e_ϕ is defined in (3.17), and the heading angle ϕ is not strictly necessary. However, we keep it for visualization purposes. Putting together the above derivations gives us the equations describing the motion of the kinematic bicycle model in the Frenet frame

$$\dot{s} = \frac{v\cos(e_\phi)}{1-\kappa(s)e} \tag{3.25}$$

$$\dot{e} = v\sin(e_\phi) \tag{3.26}$$

$$\dot{v} = \frac{v_u - v}{\tau} \tag{3.27}$$

$$\dot{\phi} = \frac{v \tan \delta}{l_{wb}} \tag{3.28}$$

$$\dot{e_\phi} = \frac{v \tan \delta}{l_{wb}} - \kappa(s)\frac{v\cos(e_\phi)}{1-\kappa(s)e}. \tag{3.29}$$

where $l_{wb} = 0.324$m denotes the wheelbase and $\dot{v}$ is described in details in section 3.3.1. In our implementation, we discretize the above equations using the fourth order Runge-Kutta method [27], with the discretization step $dt = 0.1$ and express the discrete-time system dynamics for the vehicles as

$$x_{t+1} = f_{\text{rk4}}(x_t, u_t), \tag{3.30}$$

with the state vector

$$x_t = [s_t, e_t, v_t, \phi_t, e_{\phi,t}]^T, \tag{3.31}$$

and the input control vector

$$u_t = [v_{u,t}, \delta_t]^T, \tag{3.32}$$

where $v_{u,t}$ denotes the input velocity and δ_t denotes the steering angle. See section 3.3.1 for an explanation on why we use the input velocity v_u instead of the acceleration a. The system in (3.30) is the one used in all subsequent MPC and LMPC formulations.

As explained in section 3.2, the road model consists of a set of discrete points. Therefore it is important that the discretization step for the road, Δ_L, is proportional to the vehicle velocity and the time discretization step, *i.e.,* $\Delta_L = v_t dt$. This ensures that the curvature term $\kappa(s)$ in (3.25) and (3.29) reflects the accurate road curvature value when given to a predictive control of a certain horizon N.

Notice that the state space in the feedback loop is assumed to be fully observable with no noise and that the system dynamics are known perfectly. This is not true in the real world. There are methods to extend the LMPC framework with robust control approaches to mitigate these factors in the real world, see for example [28] and [29]. However, in this work, we are focusing on optimizing trajectories in simulation and intend to use optimal trajectories as references for an online reference tracking controller on the real system. Therefore, the use of robust control approaches fall outside the scope of our implementation.

3.4 Hyperellipse Obstacle Constraints

In two of the three scenarios studies in this project, we introduce obstacle vehicles on the road. There are different ways of modelling an obstacle vehicles. For example, one can model the obstacle vehicles as rectangles or inflated ellipses. We opt for modelling the obstacles as superellipse which gives us a smooth and convex shape that is in between a rectangle and an ellipse. Superellipses were discovered by Lamé in 1818 and have the following implicit formulation

$$\left|\frac{x}{a}\right|^n + \left|\frac{y}{b}\right|^n = 1, \quad a, b, n > 0 \tag{3.33}$$

which gives a convex shape centered at $(0, 0)$ and bounded inside a rectangle with side lengths $2a$ in the x-axis and $2b$ in the y-axis [30]. When $n = 2$ we get an ellipse and when $n > 2$ we get a a shape that looks like a rounded rectangle of size $a \times b$ known as a “hyperellipse”. The case where $a = b$ and $n = 4$ is known as a “squircle” [31].

In our implementation we want to derive the formula for a hyperellipse which inscribes a rectangle of dimensions (l, w). Suppose we have a square with side length l centered at $(0, 0)$. Then, we know that the squircle inscribed inside the square is given by

$$\left(\frac{2x}{l}\right)^4 + \left(\frac{2y}{l}\right)^4 = 1. \tag{3.34}$$

The squircle which inscribes the square of size (l, l) must be slightly larger. Let's call the additional length required Δ_l. By definition, we know that such a squircle should contain the corners of the square. We plug the top right corner $(\frac{l}{2}, \frac{l}{2})$ into (3.34) and get

$$\left(\frac{2(l/2)}{l+\Delta_l}\right)^4 + \left(\frac{2(l/2)}{l+\Delta_l}\right)^4 = 1. \tag{3.35}$$

Solving for Δ_l gives us

$$\Delta_l = (2^{1/4} - 1)l. \tag{3.36}$$

Thus, the formula for a squircle which inscribes a square of size (l, l), centered at (x_0, y_0), is given by

$$\left(\frac{2(x-x_0)}{l+\Delta_l}\right)^4 + \left(\frac{2(y-y_0)}{l+\Delta_l}\right)^4 = 1, \tag{3.37}$$

where $\Delta_l = (2^{1/4} - 1)l$.

To extend this result to the hyperellipse, we use the fact that the aspect ratio between the hyperellipse and the rectangle inside it must be the same. Thus, if the hyperellipse has the dimensions $(l + \Delta_l, w + \Delta_w)$ and the rectangle it inscribes has the dimensions (l, w), the following relation must hold

$$\frac{l+\Delta_l}{w+\Delta_w} = \frac{l}{w}, \tag{3.38}$$

which solving for Δ_w gives us

$$\Delta_w = \frac{w}{l}\Delta_l \tag{3.39}$$

By combining the results for the squircle inscribing a square (3.37) and (3.39) we get the following general formula for a hyperellipse which inscribes a rectangle of size (l, w), centered at (x_0, y_0)

$$\left(\frac{2(x-x_0)}{l+\Delta_l}\right)^n + \left(\frac{2(y-y_0)}{w+\Delta_w}\right)^n = 1, \quad n \geq 2 \tag{3.40}$$

where $\Delta_l = (2^{1/n} - 1)l$ and $\Delta_w = \frac{w}{l}\Delta_l$. To express the hyperellipse conditions in the Frenet frame we simply replace the points (x, y) with (s, e), which means that our hyperellipse will also conform to the curved shape of the road. This is a desired side-effect of the Frenet frame. Furthermore, we assume that the ego vehicle is a point mass object. Therefore, we need to inflate the hyperellipse and replace the denumerators in (3.40) by $(2l + \Delta_l)$ and $(2w + \Delta_w)$ to account for the dimensions of the ego vehicle in the obstacle avoidance formulation. This is illustrated in fig. 3.6.

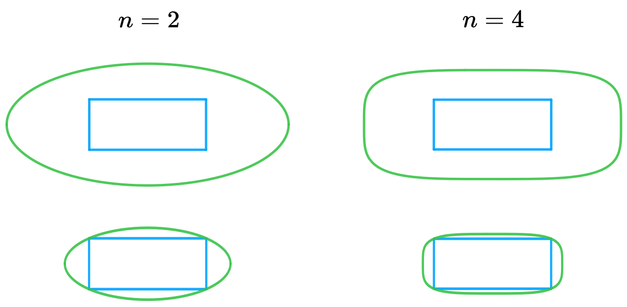


Figure 3.6: Hyperellipses of different orders. With inflation on the top row and without inflation on the bottom.

A limitation of this approach is the fact that we do not account for the heading angle of the obstacle vehicles. In the Cartesian frame, this would not be sufficient and we would need to define rotated hyperellipses. However, in the Frenet Frame, the hyperellipses are always rotated in parallel with the tangent of the road curve, which is sufficient for expressing an inflated collision avoidance condition, since the ego vehicle are modelled as point mass objects. If we want to avoid inflating the obstacles and express the ego vehicle with its own hyperellipse, we would need to compute the minimum distance between two hyperellipses to ensure safety. Computing the minimum distance between hyperellipses is itself an optimization problem [32], which would further complicate the overall LMPC problem. In that case, we refer to the work in [33] and [34], which develops an optimization based collision avoidance constraint formulation that can account for the dimensions of both the obstacles and the ego vehicle.

3.5 Generating Initial Feasible Trajectory

The initial trajectories for each scenario are generated using a reference tracking MPC with a quadratic cost-to-go and no terminal components, which at time t is formulated as

$$J_{t\to t+N}^{MPC}(x_t) = \min_{u_{t|t},\cdots,u_{t+N-1|t}} \sum_{k=t}^{N-1} \bar{x}_{k|t}^T Q \bar{x}_{k|t} + u_{k|t}^T R u_{k|t} \tag{3.41a}$$

$$\text{s.t.} \quad x_{k+1|t} = f_{\text{rk4}}(x_{k|t}, u_{k|t}) \tag{3.41b}$$

$$|x_{k|t}| \le \begin{bmatrix} 2L \\ w_{\text{lane}} - w_{\text{car}}/2 \\ 0.5 \\ 2\pi \\ 2\pi \end{bmatrix} \qquad |u_{k|t}| \le \begin{bmatrix} 1.7 \\ \pi/4 \end{bmatrix} \tag{3.41c}$$

$$x_{t|t} = x_t \tag{3.41d}$$

$$\forall k \in \{t, \ldots, t+N-1\}, \tag{3.41e}$$

with $Q = diag(1, 500, 100, 20, 20)$, $R = diag(1, 2)$, L denotes the track length, $w_{\text{lane}} = 0.5$ denotes the lane width, and $w_{\text{car}} = 0.2485$ denotes the vehicle track width. The vector $\bar{x}_t$ computes the deviation from the reference, *i.e.,* $\bar{x}_t = x_t - x_{\text{ref},t}$. The above nonlinear MPC and all subsequent LMPC controllers are implemented in Python using the CasADi library [35] with the IPOPT optimizer and horizon length $N = 7$.

The references vector for the single vehicle scenario is given by

$$x_{\text{ref},t} = \begin{bmatrix} s_{\text{ref},t} \\ -w_{\text{lane}}/2 \\ 0.5 \\ \phi_{s,t} \\ 0 \end{bmatrix}, \quad \forall t \ge 0 \tag{3.42}$$

where $s_{\text{ref},t}$ and $\phi_{s,t}$ are provided by the discrete implementation of the road curve (see section 3.2), going from $s_{\text{ref},0} = 0$ to $s_{\text{ref},T^0} = L$. In other words, we follow the road curve on the right lane. For the second and third scenarios, we use a similar approach and ensure that the ego vehicle overtakes the obstacle vehicle by changing the reference on the e_t state when the ego vehicle is within a distance of two vehicle lengths, *i.e.,* $2l_{\text{car}}$ in the s-direction. For the oncoming vehicle, we generate the reference trajectory in the opposite direction, going from $s_{\text{ref},0} = L$ to $s_{\text{ref},T^0} = 0$.

3.6 Algebraic Sigmoid Mini-time Stage cost

To avoid defining a discrete minimum time stage cost as in (2.12), we define the following approximation of the stage cost using the algebric Sigmoid function, similar to [14],

$$h(x_t, u_t) = \begin{cases} 1 & ([1, 0, 0, 0, 0]x_t = s_t) < L \\ 0 & \text{otherwise} \end{cases} \tag{3.43}$$

$$\approx \frac{1}{2}\left(\frac{k(s_t - L)}{\sqrt{1 + (k(s_t - L))^2}} + 1\right), \tag{3.44}$$

where $k = -4$ and L denotes the road length. This stage cost is used in all subsequent LMPC formulations.

3.7 Modelling the Single Vehicle Scenario

The first scenario consists of a single car starting and finishing at the states x_S and x_F, respectively, with

$$x_S = \begin{bmatrix} 0 \\ -w_{\text{lane}}/2 \\ 0 \\ \pi \\ 0 \end{bmatrix} \qquad x_F = \begin{bmatrix} L \\ -w_{\text{lane}}/2 \\ 0 \\ \pi \\ 0 \end{bmatrix} \tag{3.45}$$

Since the first scenario has no obstacles, we can define the state and input constraints as convex sets and, hence, we can adopt the following relaxed LMPC formulation

$$J^{\text{LMPC},j}_{t\to t+N}(x^j_t) = \min_{\boldsymbol{u}^j,\boldsymbol{\lambda}^j} \sum_{k=t}^{N-1} h(x^j_{k|t}, u^j_{k|t}) + \sum_{i=0}^{j-1}\sum_{k=0}^{T^i} \lambda^i_k J^i_{k\to T^i} x^i_k \tag{3.46a}$$

$$\text{s.t.} \quad x^j_{k+1|t} = f_{\text{rk4}}(x^j_{k|t}, u^j_{k|t}) \tag{3.46b}$$

$$|x^j_{k|t}| \leq \begin{bmatrix} 2L \\ w_{\text{lane}} - w_{\text{car}}/2 \\ 0.7 \\ 2\pi \\ 2\pi \end{bmatrix} \quad |u^j_{k|t}| \leq \begin{bmatrix} 1.7 \\ \pi/4 \end{bmatrix} \tag{3.46c}$$

$$|u^j_{k+1|t} - u^j_{k|t}| < dt \begin{bmatrix} 1 \\ 0.7 \end{bmatrix} \tag{3.46d}$$

$$|u^j_{t|t} - u^j_{t-1}| < dt \begin{bmatrix} 1 \\ 0.7 \end{bmatrix} \tag{3.46e}$$

$$\sum_{i=0}^{j-1}\sum_{k=0}^{T^i} \lambda^i_k x^i_k = x^j_{t+N|t} \tag{3.46f}$$

$$\sum_{i=0}^{j-1}\sum_{k=0}^{T^i} \lambda^i_k \lambda^i_k = 1 \tag{3.46g}$$

$$x^j_{t|t} = x^j_t \tag{3.46h}$$

$$\forall k \in \{t, \dots, t+N-1\}, \tag{3.46i}$$

where (3.46e) and (3.46e) include additional input rate constraint.

3.8 Modelling the Stationary Obstacle Scenario

In this scenario, we add a stationary vehicle on the same lane as the ego vehicle in the middle of the road. This could, for example, represent a broken-down vehicle or a very slow tractor. In this case, the ego vehicle will be forced to switch lanes and overtake the obstacle vehicle. We represent the obstacle avoidance constraint with an inflated hyperellipse that inscribes the rectangular shape of the vehicle of dimensions $(l_{\text{car}}, w_{\text{car}}) = (0.586, 0.2485)$ given by

$$l(x^j_{k|t}) = \left(\frac{2(s^j_{k|t} - s_{obs})}{2l_{\text{car}} + \Delta_l}\right)^n + \left(\frac{2(e^j_{k|t} - e_{obs})}{2w_{\text{car}} + \Delta_w}\right)^n \succcurlyeq 1, \tag{3.47}$$

where $(s_{\text{obs}}, e_{\text{obs}})$ denotes the position of the obstacle vehicle in the Frenet frame, and Δ_l as well as Δ_w are defined in section 3.4. Adding this inequality as a state constraint means that we no longer have a convex set of state constraints and therefore have to solve the mixed-integer LMPC problem, meaning that for each time step, we need to solve the optimization problem for every state in the sampled safe set, compute the optimal cost and pick the terminal state which gives the smallest cost. We get the following LMPC formulation

$$J^{\text{LMPC},j}_{t\to t+N}(x^j_t) = \min_{u^j_{t|t},\dots,u^j_{t+N-1|t}} \sum_{k=t}^{N-1} h(x^j_{k|t}, u^j_{k|t}) + V^{j-1}_t(x^j_{t+N|t}) \tag{3.48a}$$

$$\text{s.t.} \quad x^j_{k+1|t} = f_{\text{rk4}}(x^j_{k|t}, u^j_{k|t}) \tag{3.48b}$$

$$|x^j_{k|t}| \le \begin{bmatrix} 2L \\ w_{\text{lane}} - w_{\text{car}}/2 \\ 0.7 \\ 2\pi \\ 2\pi \end{bmatrix} \qquad |u^j_{k|t}| \le \begin{bmatrix} 1.7 \\ \pi/4 \end{bmatrix} \tag{3.48c}$$

$$|u^j_{k+1|t} - u^j_{k|t}| < dt \begin{bmatrix} 1 \\ 0.7 \end{bmatrix} \tag{3.48d}$$

$$|u^j_{t|t} - u^j_{t-1}| < dt \begin{bmatrix} 1 \\ 0.7 \end{bmatrix} \tag{3.48e}$$

$$x^j_{t+N|t} \in \mathcal{SS}^j_t \tag{3.48f}$$

$$x^j_{t|t} = x^j_t \tag{3.48g}$$

$$l(x^j_{k|t}) \succcurlyeq 1 \tag{3.48h}$$

$$\forall k \in \{t, \dots, t+N-1\}, \tag{3.48i}$$

3.9 Modelling the Oncoming Traffic Scenario

This scenario builds on the previous one by introducing another vehicle into the overall system dynamics. The oncoming vehicle will have the same parameters with the only difference that it will be traveling in the negative s direction as oncoming traffic on the left lane, relative to ego vehicle. We use the centralized multi-agent LMPC formulation described in section 2.3.2 and stack the dynamics, states and inputs of the vehicles into new column vectors representing the global system. Besides the obstacle, state, and input constraints, the multi-agent formulation requires the introduction of a global constraint that represents the collision avoidance coupling between the vehicles. At each time step, we know the position of every vehicle. Combining this with the terminal state constraint from the sampled safe set, we form an elongated hyperellipse representing the overall area covered by the other vehicle over the horizon N, see fig. 3.7. This is expressed by the following global constraint condition

$$g(x^j_{k|t}) = \begin{bmatrix} 1 - \left(\frac{2(c^j_{k|t,ego} - c^j_{onc})}{2l_{\text{onc}} + \Delta_{l_{\text{onc}}}}\right)^n + \left(\frac{2(e^j_{k|t,ego} - e^j_{k|t,onc})}{2w_{\text{car}} + \Delta_{w_{\text{onc}}}}\right)^n \\ 1 - \left(\frac{2(c^j_{k|t,onc} - c^j_{ego})}{2l_{\text{ego}} + \Delta_{l_{\text{ego}}}}\right)^n + \left(\frac{2(e^j_{k|t,onc} - e^j_{k|t,ego})}{2w_{\text{car}} + \Delta_{w_{\text{ego}}}}\right)^n \end{bmatrix} \preccurlyeq 0, \qquad (3.49)$$

with

$$\begin{aligned}
c^j_{k|t,ego} &= s^j_{k|t,ego} + \frac{l_{car}}{2} - l_{btw} \\
c^j_{k|t,onc} &= s^j_{k|t,onc} - \frac{l_{car}}{2} + l_{btw} \\
c^j_{ego} &= \frac{c^j_{t+N|t,ego} + c^j_{t|t,ego}}{2} \\
c^j_{onc} &= \frac{c^j_{t+N|t,onc} + c^j_{t|t,onc}}{2} \\
l_{ego} &= c_{t+N|t,ego} - c_{t|t,ego} + 2l_{car} \\
l_{onc} &= c_{t|t,onc} - c_{t+N|t,onc} + 2l_{car} \\
\Delta_{l_{ego}} &= (2^{1/n} - 1)l_{ego} \\
\Delta_{l_{onc}} &= (2^{1/n} - 1)l_{onc} \\
\Delta_{w_{ego}} &= \frac{w_{car}}{l_{ego}}\Delta_{l_{ego}} \\
\Delta_{w_{onc}} &= \frac{w_{car}}{l_{onc}}\Delta_{l_{onc}},
\end{aligned}$$

where $l_{btw} = 0.16\text{m}$ is the distance from the rear-end of the vehicle to the rear-axle, $c^j_{k|t,ego}$ denotes the center of the Ego vehicle, $c^j_{k|t,onc}$ denotes the center of the Onc vehicle, c^j_{onc} denotes the center of the elongated Onc vehicle, c^j_{onc} denotes the center of the elongated Ego vehicle, and l_{ego} and l_{onc} denote the elongation lengths for the Ego and Onc vehicle respectively. The overall multi-agent LMPC formulation of the two vehicles is given by

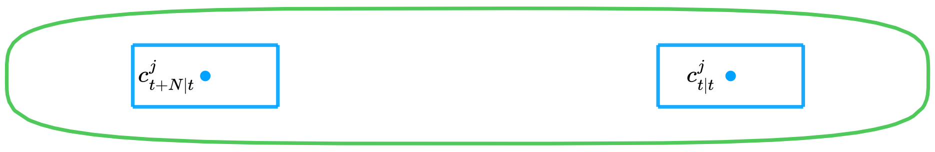


Figure 3.7: Elongated hyperellipse forming the area covered by the vehicle during N time-steps.

$$J^{\text{LMPC},j}_{t\rightarrow t+N}(x^j_t) = \min_{u^j_{t|t},\ldots,u^j_{t+N-1|t}} \sum_{k=t}^{N-1} h(x^j_{k|t}, u^j_{k|t}) + V^{j-1}_t(x^j_{t+N|t}) \tag{3.50a}$$

$$\text{s.t.} \quad x^j_{k+1|t} = f_{\text{rk4}}(x^j_{k|t}, u^j_{k|t}) \tag{3.50b}$$

$$|x^j_{k|t}| \le \begin{bmatrix} 2L \\ w_{\text{lane}} - w_{\text{car}}/2 \\ 0.7 \\ 2\pi \\ 2\pi \\ 2L \\ w_{\text{lane}} - w_{\text{car}}/2 \\ 0.5 \\ 2\pi \\ 2\pi \end{bmatrix} \qquad |u^j_{k|t}| \le \begin{bmatrix} 1.7 \\ \pi/4 \\ 1.7 \\ \pi/4 \end{bmatrix} \tag{3.50c}$$

$$|u^j_{k+1|t} - u^j_{k|t}| < dt \begin{bmatrix} 1 \\ 0.7 \\ 1 \\ 0.7 \end{bmatrix} \tag{3.50d}$$

$$|u^j_{t|t} - u^j_{t-1}| < dt \begin{bmatrix} 1 \\ 0.7 \\ 1 \\ 0.7 \end{bmatrix} \tag{3.50e}$$

$$x^j_{t+N|t} \in \mathcal{SS}^j_t \tag{3.50f}$$

$$x^j_{t|t} = x^j_t \tag{3.50g}$$

$$l(x^j_{k|t}) \succcurlyeq 1 \tag{3.50h}$$

$$g(x^j_{k|t}) \preccurlyeq 0 \tag{3.50i}$$

$$\forall k \in \{t, \ldots, t+N-1\}, \tag{3.50j}$$

Chapter 4

Results

In this chapter, we present the results of this work. First, we present the results related to the single vehicle scenario, then we present the results of the LMPC framework applied to the stationary obstacle scenario and, lastly, we provide the results for the multi-agent case in the oncoming traffic scenario. All code used to generate the results is availabe on the following GitHub repository: https://github.com/aljanabim/city-lmpc.

4.1 Single Vehicle Scenario

In fig. 4.1, the trajectories taken by the car over the different iterations are presented. As expected, we see that the vehicle follows what seems to be the shortest possible trajectory, similar to a racing driver, in the final iteration. From fig. 4.2 we observe that the converged iteration saturates both the input velocity v_u and the steering angle δ, whereas the initial trajectory, generated by the reference tracking MPC in section 3.5, is more conservative with respect to the control inputs since we have a quadratic term applied to the cost. The overall iteration performances are presented in table 4.1.

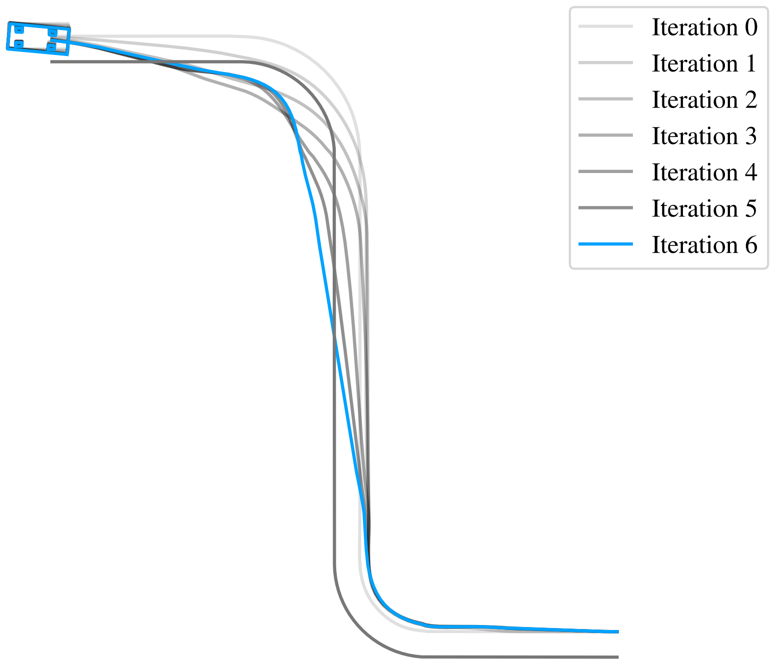


Figure 4.1: Single vehicle trajectories over the the LMPC iterations. The final iteration is indicated by the blue line.

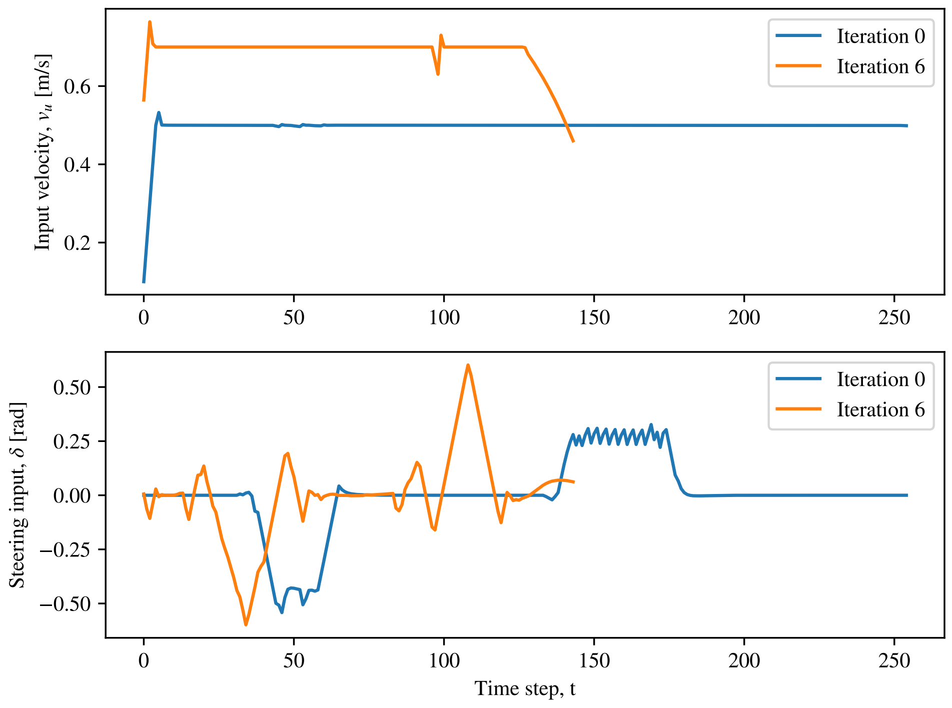


Figure 4.2: Single vehicle control inputs for the initial and final iterations. The top plot shows the input velocity v_u and the bottom plot shows the steering inputs δ.

Table 4.1: Iteration performance for the single vehicle scenario

Iteration j	**Travel Time**
0	215
1	152
2	150
3	146
4	145
5	144
6	144

4.2 Stationary Obstacle Scenario

Similar to the previous scenario, we present the trajectories for the stationary obstacle scenario together in fig. 4.3. Again, we see that the ego vehicle follows the shortest path possible and gets as close as possible to the obstacle vehicle without breaking the hyperellipse constraint. This suggest that we might need to inflate the obstacle further to take into account a safety margin for the overtake. We notice, in fig. 4.3, that the vehicle appears to be slightly rotated in its final position at the converged iteration, *i.e.,* $x^0_{T^0} \neq x^9_{T^9}$, which would suggest that the LMPC assumption of $x_F = x^j_{T^j}$ is no longer satisfied. However, this is not a problem since our stopping condition only specifies that a successful mission depends on reaching the end of the road, *i.e.,* $s_t < L$, as seen in (3.43). Hence, the final rotation of the vehicle is not relevant for an iteration to succeed.

Furthermore, we observe in fig. 4.4 that the vehicle uses more aggressive steering, δ, and saturates the input velocity, v_u, upon convergence. This scenario is slightly more complicated than the single vehicle and as seen in table 4.2, it takes 3 more iterations before it converges compared to the results in table 4.1.

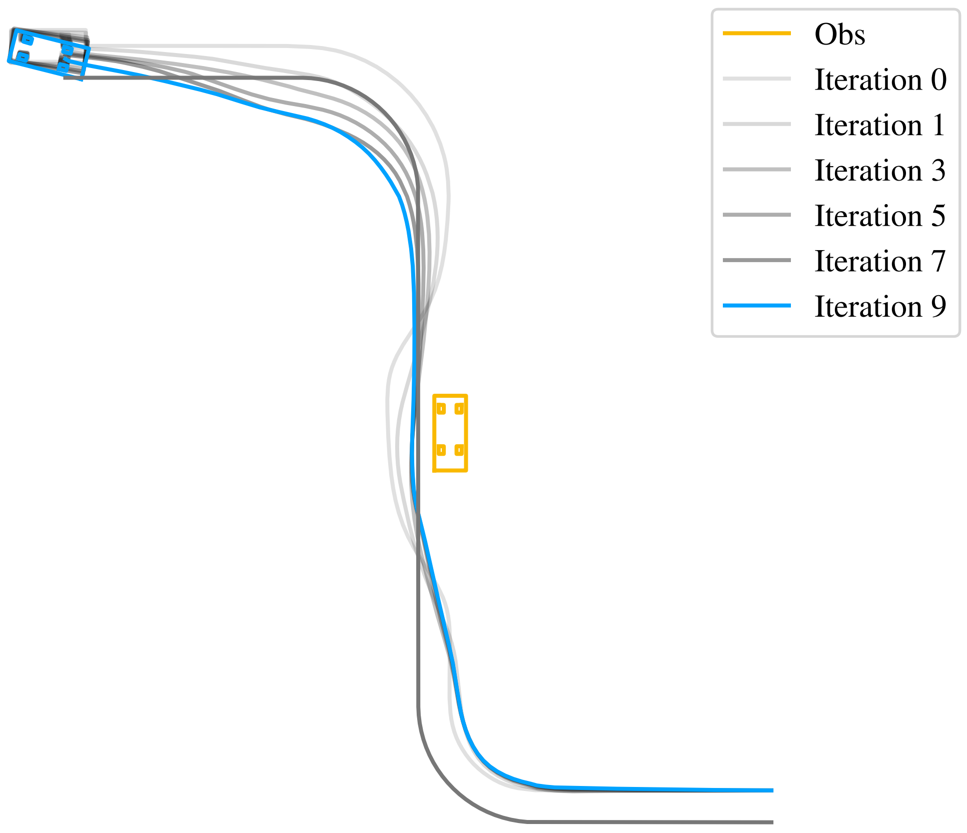


Figure 4.3: Vehicle trajectories for the obstacle scenario over the odd-numbered iterations. The final iteration is indicated by the blue line.

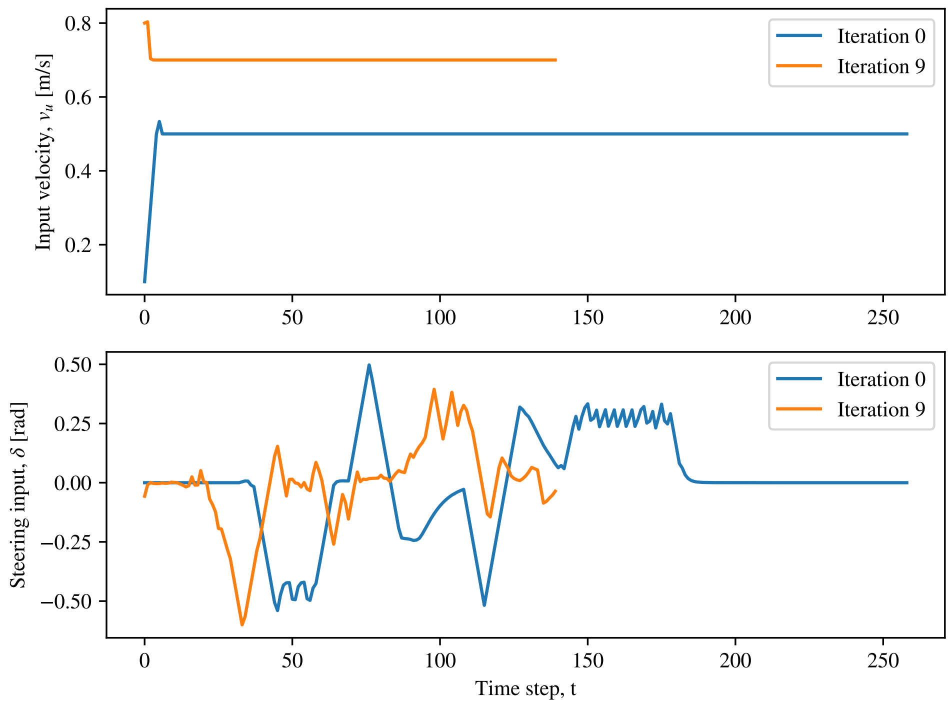


Figure 4.4: The control inputs applied in the stationary obstacle scenario for the initial and final iterations. In the top plot we see the input velocity v_u and the bottom plot shows the steering inputs δ.

Table 4.2: Iteration performance for the obstacle scenario

Iteration j	**Travel Time**
0	219
1	151
2	146
3	145
4	144
5	143
6	142
7	141
8	140
9	140

4.3 Oncoming Traffic Scenario

For the third scenario, we provide two different initial trajectories, one where we make the ego vehicle wait behind the obstacle vehicle until the oncoming vehicle passes, and one where we overtake the obstacle before the oncoming vehicle gets too close. We run the LMPC trajectory optimization procedure for both initial trajectories and find, from table 4.3 and table 4.4, that they lead to the same trajectory cost upon convergence. However, we note that the more conservative initial trajectory requires an additional iteration to converge; 5 when waiting and 4 without waiting.

Table 4.3: Iteration performance for the oncoming scenario for the case where the ego vehicle waits for the oncoming vehicle to pass in the initial trajectory.

Iteration j	**Travel Time**
0	386
1	149
2	148
3	146
4	145
5	145

Table 4.4: Iteration performance for the oncoming scenario for the case where the ego vehicle overtakes the obstacle before the oncoming vehicle passes in the initial trajectory.

Iteration j	**Travel Time**
0	240
1	148
2	147
3	145
4	145

In fig. 4.5, we show snapshots of the different initial trajectories and the time-optimal trajectory. Both initial trajectories lead to very similar optimal trajectories and therefore, we only show one of them. We note that the optimal trajectory, again, attempts to drive as close as possible to the obstacle vehicle and appears to follow the racing lines by cutting corners and driving recklessly close to the other vehicles in the road.

Figure 4.6 shows the control inputs for the different iterations. Since both initial trajectories lead to the same trajectory cost at iteration 4, we omit showing control inputs from other iterations. We note that the input velocity, v_u, is set to 0 for almost half the trajectory time when waiting for the oncoming vehicle to pass, while the initial trajectory that does not wait finishes approximately when the waiting vehicle would have started to move again. As with the previous scenarios, we see in fig. 4.6, that the optimal trajectory saturates the inputs and achieved a faster trajectory than both initial conditions.

To confirm that the collision avoidance constraints are indeed satisfied, we evaluate the hyperellipse conditions for the time-optimal trajectory in fig. 4.7. We note that both conditions remain above the safety line. However, for the collision condition with the Onc vehicle, we notice that the hyperellipse condition drops below the safety line at approximately step 105. This is caused by the fact that our implementation includes a slack variable on that particular safety condition to ensure LMPC feasibility in practice. In this case, this implies that the Ego vehicle would make contact with the Onc vehicle near the left bend of the track. To mitigate this, we would increase the inflation factor in the collision hyperellipse formulation.

Note that we got infeasible solutions when we tried shorter optimization horizons than $N = 7$, for the cases where the ego vehicle waits for the oncoming vehicle to pass in the initial trajectory. In fact, for the results of this scenario we used $N = 15$.

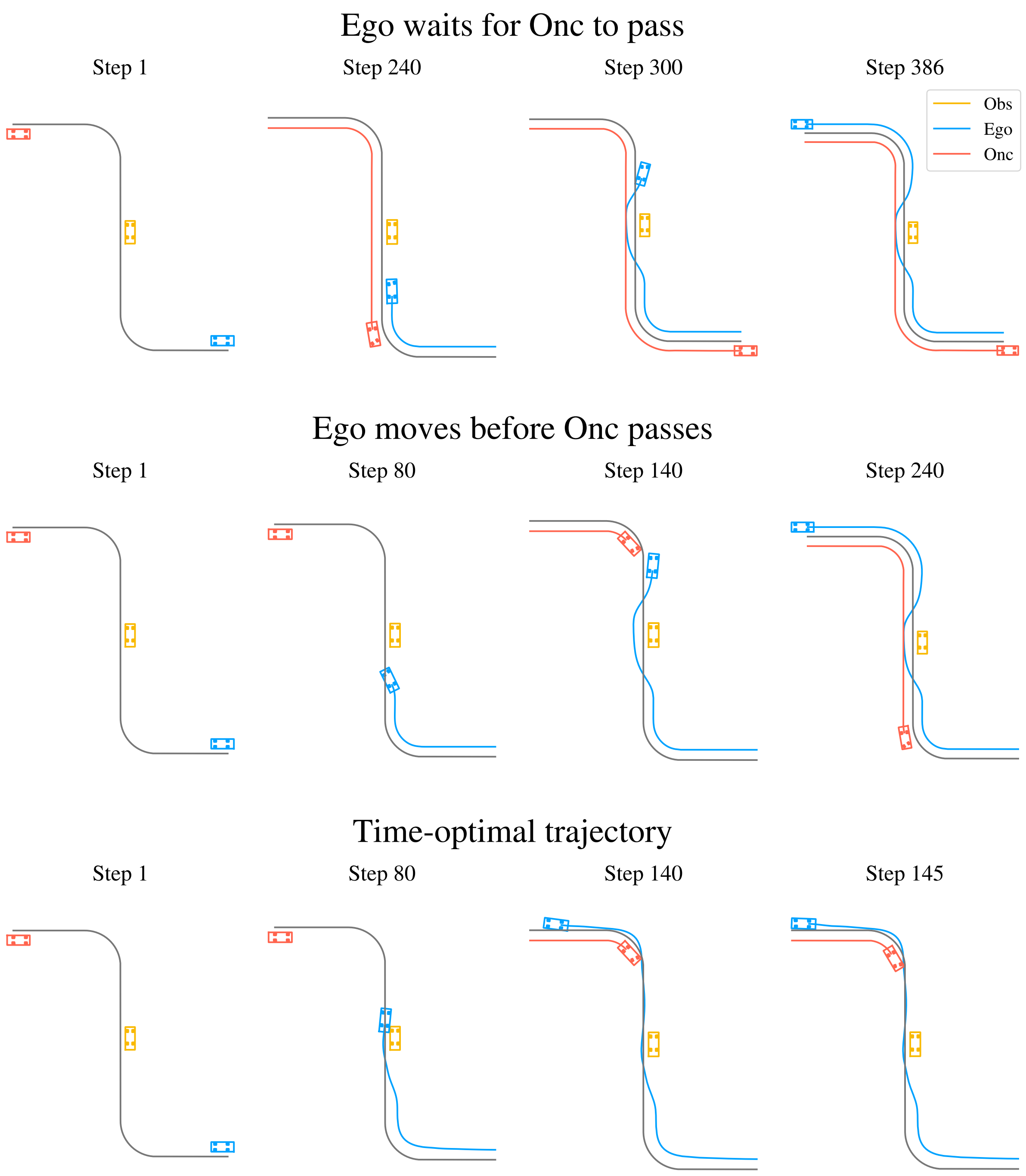


Figure 4.5: Snapshots of the vehicle trajectories at different time steps for the different initial trajectories and the time-optimal trajectory in the oncoming traffic scenario.

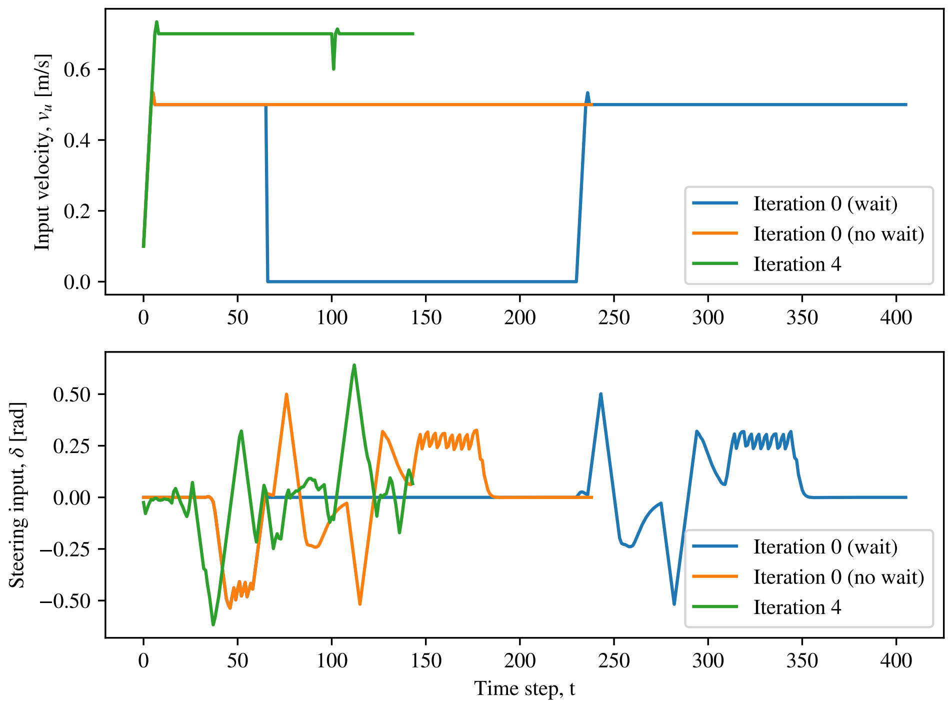


Figure 4.6: The control inputs applied in the oncoming traffic scenario for the different initial trajectories and the time-optimal trajectory achieved at iteration 4. In the top plot see the input velocity v_u and the bottom plot shows the steering inputs δ.

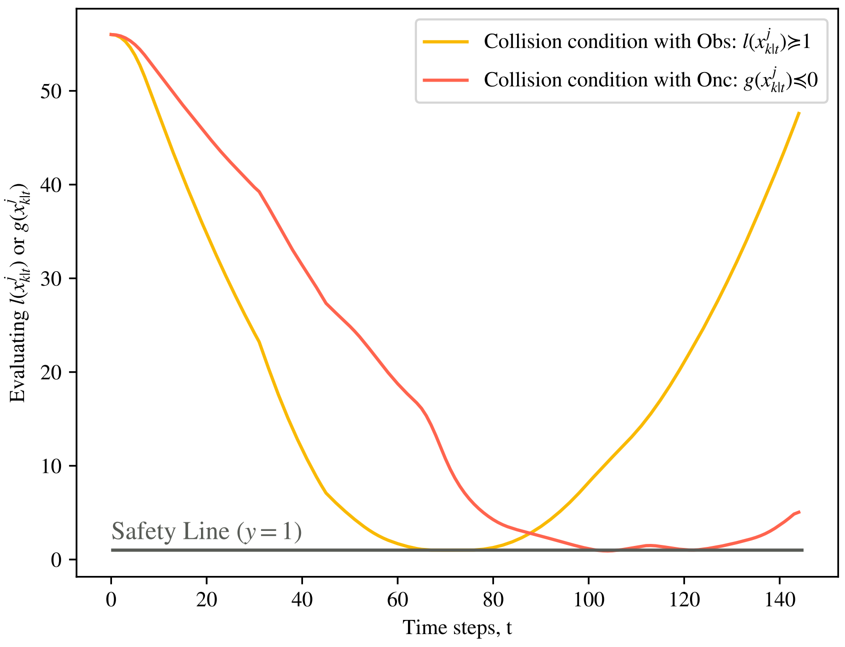


Figure 4.7: Evaluating the hyperellipse obstacle avoidance constraint with respect to the Ego vehicle in the oncoming traffic scenario.

Chapter 5

Discussion

Overall, we see that LMPC is a data-driven method that proves to be sample-efficient compared to other approaches such as reinforcement learning where convergence toward reasonable behavior might take thousands of episodes instead of just a few iterations as in LMPC. We see from tables 4.1, 4.2, 4.3, and 4.4 that all scenarios converged in less than 10 iterations.

Due to the mixed-integer nature of the scenarios, which cannot adopt relaxed constraints, we find that LMPC is not well-suited for real-time control and lends itself better as an optimal trajectory generator running on generated data and trained in simulations. In this mode of operation, we see from figures 4.1, 4.3, and 4.5 that LMPC is capable of generating time-optimal trajectories which satisfy all constraints. It is also possible to modify the optimality criteria. For instance we could add a penalty on actuation. Another reason for why LMPC is not suitable for physical, real-time scenarios is due to the assumption that every task execution has to have the same initial and final states, whereas when running online, traffic scenarios tend to be non-iterative with varying starting and final states. In contrast, LMPC is well-suited for iterative tasks such as racing, where lap times can be improved after each lap is completed. However, in fig. 4.5 we see that given the same initial and final states, LMPC can converge to the optimal solution even when the initial trajectory might not be close to the optimal solution, such as in the third scenario, when one of the initial trajectories had the ego vehicle wait for the oncoming traffic to pass.

We found that a step horizon shorter than $N = 7$ led to infeasible solutions in the third scenario with an initial trajectory that had the ego vehicle waiting. With a shorter horizon, LMPC was subject to making short-sighted decisions which led to an overall increased travel time. For instance, if it begins the overtake too late, it would need to reverse to make way for the oncoming vehicle before it can drive towards the goal. We conclude that the horizon length N is a parameter which plays an important role in finding an optimal solution for initial feasible trajectories that might not be close to the optimal trajectory.

In the multi-vehicle LMPC formulation of the third scenario, we use the centralized approach since we are generating the optimal trajectories in simulation and can afford the increased computational efficiency while in return getting a solution which takes advantage of the complete system. However, if the learning was to happen on real vehicles, we believe that the centralized approach becomes intractable due to the computation times we experienced.

We note that the LMPC assumption on having an initial feasible trajectory is not difficult to satisfy for a smart city scenario because we can find demonstrations by humans or use a conservative strategy to achieve this. In future smart cities, this approach can be suitable for offline training on any trajectory data that may be generated from autonomous vehicles and the infrastructure sensors. Over time, the accumulated data from multiple sections of a smart city can be used to construct large data sets of optimal trajectories that will inform autonomous connected vehicles in most urban scenarios.

Chapter 6

Conclusions and Future work

6.1 Conclusions

The LMPC framework gives us a sample-efficient method to utilize vehicle and infrastructure data to improve traffic efficiency while guaranteeing safety. Multi-level autonomy can be assessed by varying the observability and controllability of the vehicles in a multi-vehicle scenario. As discussed above, the framework does have some shortcomings which make it not suitable for online use. However, for offline trajectory optimization, it's an effective method that is well-suited for smart-city scenarios.

There are multiple factors that make the LMPC framework suitable for trajectory optimization in vehicles. Traffic efficiency can be improved through the use of this framework to achieve shorter travel time. The specific objective that is targeted to be optimized can also change according to other desired criteria. For example, fuel efficiency can be a targeted outcome that can be optimized for through LMPC. The great advantage that this framework offers is that it is sample efficient; it does not require thousands of episodes which allows it to have fast convergence. These qualities of LMPC make it particularly useful in optimization of iterative tasks and is suitable for offline trajectory optimization. One drawback of the LMPC framework is that optimality could depend on the initial trajectory and horizon length of the optimization problem. This means that without a sufficient initial trajectory which satisfies all constraints, the resulting LMPC trajectory may not serve the intended objective. However, LMPC can be very useful as a data-driven trajectory generator for multi-agent systems which can cover large sets of scenarios depending on the available initial feasible trajectories.

Given arbitrary trajectory data that could be collected in a future smart-city, LMPC can unlock some generalized optimization possibilities across multiple city scenarios. For example, in a certain city, different intersections may have varying driving behaviours depending on a plethora of variables such as the number of lanes, congestion level of the intersection, and area of the city. By using LMPC, a generalized pattern of optimized driving across all intersections may be derived. Moreover, we can further increase sample efficiency by augmenting scenarios coming from the real world (such as varying the starting and goal positions) and running the training offline preemptively.

6.2 Future work

Future work includes developing an algorithm for systematically augmenting traffic scenarios and running the trajectory optimization offline. There is also an interest in testing this on a more complicated four-way intersection with more vehicles including vehicle platoons with coupled dynamics.

Another direction for future work is to implement an extension of multi-vehicle LMPC based on the work in [9] that turns the centralized multi-vehicle problem into a decentralized optimization problem solved on each vehicle independently. We see that the formulation is similar to the centralized approach. However, the global constraints and the time-varying sample safe sets are synthesized in the central layer between each iteration based on new trajectory data. This method yields a marginally lower performance, but it is much more efficient since the computation for each vehicle can be done in parallel. It is worth investigating whether this enables online trajectory optimization for smart-city scenarios.

References

[1] U. Rosolia, "Learning Model Predictive Control: Theory and Applications," Tech. Rep., 2019. [Pages 6, 8, 10, and 16.]

[2] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control*, ser. Communications and Control Engineering. London: Springer London, 2011. ISBN 978-0-85729-500-2. [Online]. Available: http://link.springer.com/10.1007/978-0-85729-501-9 [Page 7.]

[3] F. Borrelli, A. Bemporad, and M. Morari, "Predictive Control for linear and hybrid systems," Tech. Rep., 2016. [Page 7.]

[4] H. Chen and F. Allgöwer, "Nonlinear Model Predictive Control Schemes with Guaranteed Stability," in *Nonlinear Model Based Process Control*. Dordrecht: Springer Netherlands, 1998, pp. 465–494. [Online]. Available: http://link.springer.com/10.1007/978-94-011-5094-1_16 [Page 7.]

[5] U. Rosolia and F. Borrelli, "Learning Model Predictive Control for iterative tasks. A Data-Driven Control Framework," *CoRR*, vol. abs/1609.01387, 9 2016. [Online]. Available: http://arxiv.org/abs/1609.01387 [Pages 9, 10, 11, and 16.]

[6] ——, "Minimum time learning model predictive control," *International Journal of Robust and Nonlinear Control*, vol. 31, no. 18, pp. 8830–8854, 12 2021. doi: 10.1002/rnc.5284. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/rnc.5284 [Pages 12, 14, 15, 16, 17, and 20.]

[7] ——, "Learning Model Predictive Control for Iterative Tasks: A Computationally Efficient Approach for Linear System," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3142–3147, 7 2017. doi: 10.1016/j.ifacol.2017.08.324. [Online]. Available: https:

//linkinghub.elsevier.com/retrieve/pii/S2405896317306523 [Pages 13, 15, and 16.]

[8] G. Egger, D. Chaltsev, A. Giusti, and D. T. Matt, "A deployment-friendly decentralized scheduling approach for cooperative multi-agent systems in production systems," in *Procedia Manufacturing*, vol. 52. Elsevier B.V., 2020. doi: 10.1016/j.promfg.2020.11.023. ISSN 23519789 pp. 127–132. [Page 18.]

[9] E. L. Zhu, Y. R. Stürz, U. Rosolia, and F. Borrelli, "Trajectory Optimization for Nonlinear Multi-Agent Systems using Decentralized Learning Model Predictive Control," 4 2020. [Online]. Available: http://arxiv.org/abs/2004.01298 [Pages 18, 19, and 54.]

[10] Y. R. Stürz, E. L. Zhu, U. Rosolia, K. H. Johansson, and F. Borrelli, "Distributed Learning Model Predictive Control for Linear Systems," 6 2020. [Online]. Available: http://arxiv.org/abs/2006.13406 [Pages 18 and 19.]

[11] U. Rosolia, A. Carvalho, and F. Borrelli, "Autonomous racing using learning Model Predictive Control," in *Proceedings of the American Control Conference*. Institute of Electrical and Electronics Engineers Inc., 6 2017. doi: 10.23919/ACC.2017.7963748. ISBN 9781509059928. ISSN 07431619 pp. 5115–5120. [Page 19.]

[12] U. Rosolia and F. Borrelli, "Learning How to Autonomously Race a Car: A Predictive Control Approach," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2713–2719, 11 2020. doi: 10.1109/TCST.2019.2948135 [Page 20.]

[13] M. Brunner, U. Rosolia, J. Gonzales, and F. Borrelli, "Repetitive learning model predictive control: An autonomous racing example," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 12 2017. doi: 10.1109/CDC.2017.8264027. ISBN 978-1-5090-2873-3 pp. 2545–2550. [Online]. Available: http://ieeexplore.ieee.org/document/8264027/ [Page 20.]

[14] G. Li, A. Tunchez, and G. Loianno, "Learning Model Predictive Control for Quadrotors," 2 2022. [Online]. Available: http://arxiv.org/abs/2202.07716 [Pages 20 and 34.]

[15] M. J. Alam and M. A. Habib, “A dynamic programming optimization for traffic microsimulation modelling of a mass evacuation,” *Transportation Research Part D: Transport and Environment*, vol. 97, 8 2021. doi: 10.1016/j.trd.2021.102946 [Page 20.]

[16] J. Karlsson, C. I. Vasile, J. Tumova, S. Karaman, and D. Rus, “Multi-Vehicle Motion Planning for Social Optimal Mobility-on-Demand,” in *Proceedings - IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers Inc., 9 2018. doi: 10.1109/ICRA.2018.8462968. ISBN 9781538630815. ISSN 10504729 pp. 7298–7305. [Page 20.]

[17] J. Chen, J. Li, C. Fan, and B. Williams, “Scalable and Safe Multi-Agent Motion Planning with Nonlinear Dynamics and Bounded Disturbances,” 12 2020. [Online]. Available: http://arxiv.org/abs/2012.09052 [Page 20.]

[18] D. Cairano, A. P. Vinod, S. Safaoui, A. Chakrabarty, R. Quirynen, N. Yoshikawa, and S. Di Cairano, “Safe multi-agent motion planning via filtered reinforcement learning,” Tech. Rep., 2022. [Online]. Available: https://www.merl.com [Page 20.]

[19] D. Chen, M. Hajidavalloo, Z. Li, K. Chen, Y. Wang, L. Jiang, and Y. Wang, “Deep Multi-agent Reinforcement Learning for Highway On-Ramp Merging in Mixed Traffic,” 5 2021. [Online]. Available: http://arxiv.org/abs/2105.05701 [Page 20.]

[20] A. Rana and A. Malhi, “Building Safer Autonomous Agents by Leveraging Risky Driving Behavior Knowledge,” 3 2021. doi: 10.1109/CCCI52664.2021.9583209. [Online]. Available: http://arxiv.org/abs/2103.10245http://dx.doi.org/10.1109/CCCI52664.2021.9583209 [Page 20.]

[21] A. Micaelli, C. Samson, and C. Samson Trajectory, “Trajectory tracking for unicycle-type and two-steering-wheels mobile robots,” Tech. Rep., 1993. [Online]. Available: https://hal.inria.fr/inria-00074575 [Pages 23, 27, and 28.]

[22] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, “Optimal trajectory generation for dynamic street scenarios in a frenét frame,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2010.

doi: 10.1109/ROBOT.2010.5509799. ISBN 9781424450381. ISSN 10504729 pp. 987–993. [Page 23.]

[23] F. J. Jiang, M. Al-Janabi, T. Bolin, K. H. Johansson, and J. Martensson, "SVEA: an experimental testbed for evaluating V2X use-cases," in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2022-October. Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ITSC55140.2022.9922544. ISBN 9781665468800 pp. 3484–3489. [Page 25.]

[24] R. N. Jazar, *Vehicle Dynamics: Theory and Application*. Boston, MA: Springer US, 2008. ISBN 978-0-387-74243-4. [Online]. Available: http://link.springer.com/10.1007/978-0-387-74244-1 [Page 25.]

[25] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2015-August. Institute of Electrical and Electronics Engineers Inc., 8 2015. doi: 10.1109/IVS.2015.7225830. ISBN 9781467372664 pp. 1094–1099. [Page 26.]

[26] R. Rajamani, *Vehicle Dynamics and Control*, ser. Mechanical Engineering Series. Boston, MA: Springer US, 2012. ISBN 978-1-4614-1432-2. [Online]. Available: https://link.springer.com/10.1007/978-1-4614-1433-9 [Pages 26 and 27.]

[27] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3rd ed. Cambridge University Press, 2007. ISBN 0521880688. [Online]. Available: http://www.amazon.com/Numerical-Recipes-3rd-Scientific-Computing/dp/0521880688/ref=sr_1_1?ie=UTF8&s=books&qid=1280322496&sr=8-1 [Page 29.]

[28] U. Rosolia, X. Zhang, and F. Borrelli, "Robust Learning Model Predictive Control for Linear Systems Performing Iterative Tasks," 11 2019. [Online]. Available: http://arxiv.org/abs/1911.09234 [Page 30.]

[29] ——, "Robust learning model predictive control for iterative tasks: Learning from experience," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 12 2017. doi: 10.1109/CDC.2017.8263812. ISBN 978-1-5090-2873-3 pp. 1157–1162.

[Online]. Available: http://ieeexplore.ieee.org/document/8263812/ [Page 30.]

[30] E. W. Weisstein, "Superellipse." [Online]. Available: https://mathworld.wolfram.com/Superellipse.html [Page 30.]

[31] "Superellipse - Wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Superellipse [Page 30.]

[32] D. Eberly, *Robust and Error-Free Geometric Computing*. CRC Press, 1 2021. ISBN 9781000056624. [Online]. Available: https://www.taylorfrancis.com/books/9781000056624 [Page 32.]

[33] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-Based Collision Avoidance," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, 5 2021. doi: 10.1109/TCST.2019.2949540 [Page 32.]

[34] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli, "Autonomous Parking Using Optimization-Based Collision Avoidance," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 12 2018. doi: 10.1109/CDC.2018.8619433. ISBN 978-1-5386-1395-5 pp. 4327–4332. [Online]. Available: https://ieeexplore.ieee.org/document/8619433/ [Page 32.]

[35] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, M. Diehl, J. A. E. Andersson, J. Gillis, J. B. Rawlings, and M. Diehl, "CasADi-A software framework for nonlinear optimization and optimal control," Tech. Rep. [Page 33.]

TRITA-EECS-EX-2023:125