ETH zürich

Institute for Dynamic Systems and Control

Albert Gassol Puigjaner

Constrained Bayesian Optimization for MPC tuning in autonomous racing

Semester Project

Institute for Dynamic Systems and Control
Swiss Federal Institute of Technology Zurich

Supervision

Manish Prajapat
Prof. Dr. Melanie Zeilinger

October 2022

IDSC-MZ-MP-01

IDSC

Abstract

This semester project presents a method to automatically tune the weights of the cost function of a Model Predictive Controller used to control a Formula Student Driverless car. The approach of this project relies on constrained and unconstrained Bayesian Optimization to acquire new sets of weights that are continuously tried in simulation to test their performance in terms of lap time. The algorithm keeps running the autonomous system pipeline in a simulated environment and changes the Model Predictive Controller weights according to the Bayesian Optimization proposals. Performance metrics are acquired at the end of the simulation and are used to model the performance function used in Bayesian Optimization. Finally, constraints are introduced in the optimization to also consider performance-based constraints. These constraints will allow testing the algorithm online in the real platform with safety guarantees.

Keywords: Bayesian Optimization, Model Predictive Control.

Contents

Nomenclature

Mathematical Symbols

$\alpha_f l$	Front left tyre slip angle
$\alpha_f r$	Front right tyre slip angle
$\alpha_r l$	Rear left tyre slip angle
$\alpha_r r$	Rear right tyre slip angle
β	Exploration vs. exploitation trade-off parameter
$\boldsymbol{\Theta}$	Cost function weights data-set
$\boldsymbol{\theta}$	Cost function weights vector
$\boldsymbol{f}_{aq}(\boldsymbol{\theta})$	Acquisition function
$\boldsymbol{f}_{d}(\cdot,\cdot)$	4 wheel model of the dynamics of the car
$\boldsymbol{f}_{LCB}(\boldsymbol{\theta})$	Lower Confidence Bounds function
$\boldsymbol{f}_{RK}(\cdot,\cdot)$	Runge-Kutta 2nd order integrator
$\boldsymbol{Q}$	State weight diagonal matrix
$\boldsymbol{R}$	Controls weight diagonal matrix
$\boldsymbol{S_2}$	Quadratic slack term weight diagonal matrix
$\boldsymbol{S_{lin}}$	Linear slack term weight vector
$\boldsymbol{S}$	Matrix of predicted slacks inside the horizon
$\boldsymbol{s}$	Slack variable vector
$\boldsymbol{U}$	Matrix of predicted controls inside the horizon
$\boldsymbol{u}$	Car controls vector
$\boldsymbol{X}$	Matrix of predicted states inside the horizon
$\boldsymbol{x}$	Car state vector
$\boldsymbol{y}_t$	lap-time samples data-set at time t
$\dot{s}$	Progress rate along the reference trajectory
$\mu(\boldsymbol{\theta})$	Mean function of a Gaussian Process
$\sigma(\boldsymbol{\theta})$	Variance function of a Gaussian Process
τ	lap-time threshold

$\tilde{f}(\boldsymbol{\theta})$	Simulation lap-time disrupted with Gaussian noise
$c_{ellipse}$	Exponential additive weight on tyre ellipse constraints
c_{track}	Exponential additive weight on track constraints
c_v	Exponential additive weight on longitudinal velocity constraints
$f(\boldsymbol{\theta})$	Lap-time black-box function
f_{opt}	Optimal lap-time
G_t	Expanders set at time t
$h_{ellipse}(\cdot,\cdot,\cdot)$	Tyre ellipse constraint
$h_{track}(\cdot,\cdot,\cdot)$	Track constraint
$h_v(\cdot,\cdot,\cdot)$	Longitudinal velocity constraint
$k(\boldsymbol{\theta},\boldsymbol{\theta}')$	Kernel function of a Gaussian Process
$k_{ellipse}$	Exponential multiplicative weight on tyre ellipse constraints
k_{track}	Exponential multiplicative weight on track constraints
k_v	Exponential multiplicative weight on longitudinal velocity constraints
$l(\boldsymbol{\theta})$	Lower confidence bound
M_t	Maximizers set at time t
$R(t)$	Cumulative regret over time at time t
$r(t)$	Instant regret at time t
S_t	Safe set at time t
T_s	MPC integration time-step
$u(\boldsymbol{\theta})$	Upper confidence bound
w_{α_f}	Weight on front tyres slip angle
w_{α_r}	Weight on rear tyres slip angle
$w_{\dot{s}}$	Weight on progress rate

Acronyms

AMZ Academic Motorsports Association of Zurich. 3

CEI Constrained Expected Improvement. 7

FSG Formula Student Germany. 3

KPI Key Performance Indicator. 4

LCB Lower Confidence Bounds. 12

LQR Linear Quadratic Regulator. 7

MPC Model Predictive Control. 3

PID Proportional Integral Derivative. 9

ROS Robot Operating System. 15

Chapter 1

Introduction

This semester project presents an automatic tuning method based on Bayesian Optimization to tune the weights of the cost function of a Model Predictive Control (MPC) developed to control Academic Motorsports Association of Zurich (AMZ)'s Formula Student Driverless Car: *Bernina*. This chapter presents a brief description of the project platform, its objectives, and the background of this project.

1.1 Formula Student Driverless

Formula Student is a university student international engineering competition whose purpose is to provide a testing ground to design and build single-seat, electric power-train racing cars. ETH's Formula Student club, AMZ, participated in the 2022 competition with its new prototype *Bernina* shown in Figure 1.1.

Figure 1.1: *Bernina*: AMZ's Formula Student 2022 prototype racing in Formula Student Germany (FSG)'s Skidpad.

The competition is divided into static and dynamic events. In the static events, teams present their engineering design process, as well as the cost of the car and their business plan. In dynamic events, teams compete in different manual and driverless disciplines. Two of these disciplines are the driverless skidpad and trackdrive, which are the target of this project.

In the driverless skidpad, the race car drives autonomously on an 8-shaped track for four laps (two laps in each circle). In trackdrive, the car races autonomously for ten laps on a track that has been previously mapped so that the global reference trajectory is known. Tracks are delimited by blue, yellow, and orange cones and are, on average, four meters wide. These disciplines are time trial based, therefore the goal is to achieve the lowest possible lap time without knocking down cones because each knocked down cone adds a penalty of 2 seconds in trackdrive and 0.2 seconds in the skidpad.

1.2 Motivation and goals

The driverless skidpad and trackdrive disciplines offer the best opportunities to put the car at its physical limits. Since the optimal race line is known, the car only needs to localize itself inside the previously mapped environment and continuously compute the desired control commands using a curvilinear MPC. Therefore, it becomes critical to fine-tune the controller parameters to achieve the best performance by pushing the car to its limits in a controlled manner.

In past seasons, manual tuning, aided by a Key Performance Indicator (KPI), was the only available option to obtain a successful set of parameters. However, two main issues arise with this option. The first one is the amount of time and resources needed to achieve a desired set of parameters. Control engineers had to allocate a considerable amount of time to first tune the MPC in simulation and, afterward, test and tune it with the race car. The second issue comes with the sub-optimal solutions of hand-tuning since hand-tuning is usually driven by intuition and requires a certain degree of previous experience.

Due to a lack of resources and testing time, along with the motivation to reduce the burden of many hours of hand-tuning, an automatic tuning approach is proposed in this project. Therefore, this project presents an automatic method to tune the MPC cost function weights with the assistance of a high-fidelity simulator to run the autonomous pipeline and acquire samples of the penalty function, which is the lap time in this project. The development of this project aims to satisfy the following goals:

- To converge to an optimal set of parameters that considerably reduce the lap time compared to the baseline. The algorithm must be able to tune any combination of cost function weights, including all of them. The parameters must converge to a certain value without major fluctuations.
- To avoid knocking down cones on the track. Besides incurring a time penalty, knocking down cones can lead to unsafe situations for the integrity of the car, the environment, and the testing team. Therefore, knocking down cones should be avoided.
- To provide a method to tune a reduced amount of parameters in an online manner using the race car as a platform instead of the simulation. A performance constraint must be satisfied to always keep the lap time lower than a certain threshold to reduce the testing time.

1.3 Problem statement

The main objective of this project is to tune the cost function weights of the MPC used for known tracks taking into account a performance constraint. This problem can be formulated as the optimization problem of Equation (1.1), where $\boldsymbol{\theta}$ represents the parameter vector that includes the cost function weights that need to be tuned, and it can be solved with Bayesian Optimization. The MPC also depends on the parameter vector and can be seen as the optimization problem shown in equation (1.2). We assume that $\mathcal{J}$ is a function of the set of parameters $\boldsymbol{\theta}$ and initial

state $\boldsymbol{x}_0$ which can have some added noise. Similarly, MPC provides a unique policy $\pi(\boldsymbol{\theta}, \boldsymbol{x}_0)$ which can also have added noise. This added noise comes from the MPC solver.

Note that the functions and constraints that are optimized by Bayesian Optimization are different than the objective function of the MPC. Furthermore, to be able to use Bayesian Optimization, it is assumed that every set of parameters $\boldsymbol{\theta}_i$ will be mapped to only one value $f(\boldsymbol{\theta}_i)$ which can have some Gaussian noise. It is also assumed that it cannot be possible that the same set of parameters $\boldsymbol{\theta}_i$ can produce very different values when evaluated with the black-box function. Furthermore, the function $f(\boldsymbol{\theta})$ needs to satisfy a regularity condition. Finally, we use the continuity argument that two close enough policies produced by two different $\boldsymbol{\theta}$ will produce close enough lap times.

$$\begin{aligned} \boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \quad & f(\boldsymbol{\theta}) \\ \text{s.t.} \quad & f(\boldsymbol{\theta}) \leqslant \tau \end{aligned} \tag{1.1}$$

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{U}} \quad & \mathcal{J}(\boldsymbol{X}, \boldsymbol{U}, \boldsymbol{\theta}) \\ \text{s.t.} \quad & \boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ & \boldsymbol{x}_k \in \mathcal{X} \\ & \boldsymbol{u}_k \in \mathcal{U} \end{aligned} \tag{1.2}$$

The MPC cost function $\mathcal{J}(\boldsymbol{X}, \boldsymbol{U}, \boldsymbol{S}, \boldsymbol{\theta})$ is formulated to maximize the progress rate along the reference trajectory, but it also includes control input smoothing terms as well as reference state tracking terms, slip angle penalties and a cost on the slack variables. Furthermore, the constraints are penalized in the cost function using exponential functions to penalize track boundaries, tire ellipse, and terminal velocity constraint violations. The cost function consists of two main costs: terms at every stage of the prediction horizon and a terminal cost. These costs are presented in Equations (1.3) and (1.4), respectively. Finally, the whole MPC formulation is shown in Equation (1.5).

$$\begin{aligned} \mathcal{J}_k = - \underbrace{w_{\dot{s}} \dot{s}_k}_{progress\ rate} + \underbrace{\boldsymbol{x}_k^T \boldsymbol{Q} \boldsymbol{x}_k}_{state\ cost} + \underbrace{\boldsymbol{u}_k^T \boldsymbol{R} \boldsymbol{u}_k}_{controls\ cost} + \underbrace{w_{\alpha_f}(\alpha_{fr_k} + \alpha_{fl_k})^2 + w_{\alpha_r}(\alpha_{rr_k} + \alpha_{rl_k})^2}_{slip\ angle\ cost} \\ + \underbrace{\boldsymbol{S_{lin}} \boldsymbol{s}_k + \boldsymbol{s}_k^T \boldsymbol{S_2} \boldsymbol{s}_k}_{slack\ cost} + \underbrace{e^{c_{track} + k_{track} h_{track}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{s}_k)}}_{track\ constraints\ cost} + \underbrace{e^{c_v + k_v h_v(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{s}_k)}}_{terminal\ velocity\ constraint\ cost} \\ + \underbrace{\sum_{i=1}^{4} e^{c^i_{ellipse} + k^i_{ellipse} h^i_{ellipse}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{s}_k)}}_{ellipse\ constraints\ costs} \end{aligned} \tag{1.3}$$

$$\begin{aligned} \mathcal{J}_{\mathcal{N}} = -w_{\dot{s}}^N \dot{s}_N + \boldsymbol{x}_N^T \boldsymbol{Q}_N \boldsymbol{x}_N + \boldsymbol{u}_N^T \boldsymbol{R}_N \boldsymbol{u}_N + w_{\alpha_f}^N (\alpha_{fr_N} + \alpha_{fl_N})^2 + w_{\alpha_r}^N (\alpha_{rr_N} + \alpha_{rl_N})^2 \\ + \boldsymbol{S_{lin}}_N \boldsymbol{s}_N + \boldsymbol{s}_N^T \boldsymbol{S_2}_N \boldsymbol{s}_N + e^{c_{track_N} + k_{track_N} h_{track}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{s}_k)} + e^{c_{v_N} + k_{v_N} h_v(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{s}_k)} \\ + \sum_{i=1}^{4} e^{c^i_{ellipse_N} + k^i_{ellipse_N} h^i_{ellipse}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{s}_k)} \end{aligned} \tag{1.4}$$

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{U},\boldsymbol{S}} \quad & \mathcal{J}_N + \sum_{k=1}^{N-1} \mathcal{J}_k \\ \text{s.t.} \quad & \boldsymbol{x}_{k+1} = \boldsymbol{f}_{RK}(\boldsymbol{f}_d(\boldsymbol{x}_k, \boldsymbol{u}_k), T_s) \\ & \boldsymbol{x}_k \in \mathcal{X} \\ & \boldsymbol{u}_k \in \mathcal{U} \end{aligned} \tag{1.5}$$

where $\boldsymbol{f}_{RK}(\cdot)$ is the Runge-Kutta integrator, $\boldsymbol{f}_d(\cdot)$ is the car model and T_s is the time integration step.

From the cost function terms formulated in Equations (1.3) and (1.4) we can extract the parameter vector $\boldsymbol{\theta}$. The parameter vector can include any selection of cost function weights, for instance, Equation (1.6) includes all the cost function weights. The Bayesian Optimizer goal is to obtain the optimal set of parameters $\boldsymbol{\theta}^*$ that minimize the underlying black-box function $f(\boldsymbol{\theta})$ and satisfy the performance constraint on the black-box function, as presented in Equation (1.1).

$$\boldsymbol{\theta} = \begin{bmatrix} w_{\dot{s}} \\ \text{diag}(\boldsymbol{Q}) \\ \text{diag}(\boldsymbol{R}) \\ w_{\alpha_f} \\ w_{\alpha_r} \\ \text{diag}(\boldsymbol{Q}_N) \\ \text{diag}(\boldsymbol{R}_N) \\ w_{\alpha_f}^N \\ w_{\alpha_r}^N \\ \text{diag}(\boldsymbol{S_{lin}}) \\ \text{diag}(\boldsymbol{S_2}) \\ \text{diag}(\boldsymbol{S_{lin}}_N) \\ \text{diag}(\boldsymbol{S_2}_N) \\ c_{track} \\ k_{track} \\ c_{ellipse} \\ k_{ellipse} \\ c_v \\ k_v \end{bmatrix} \tag{1.6}$$

In order to perform the optimization and tune the weights, the Bayesian Optimization module needs to communicate with the MPC as well as with the simulator, the rest of the modules of the pipeline are of no interest to the Bayesian Optimizer. Figure 1.2 presents AMZ's Autonomous Systems pipeline, where the Bayesian Optimization tuning is represented as *Autotuner on FSSim*. The *Autotuner* communicates with the MPC module, denoted as *High-Level Control*, and with the Simulation, denoted as *FSSim* (Formula Student Simulator).

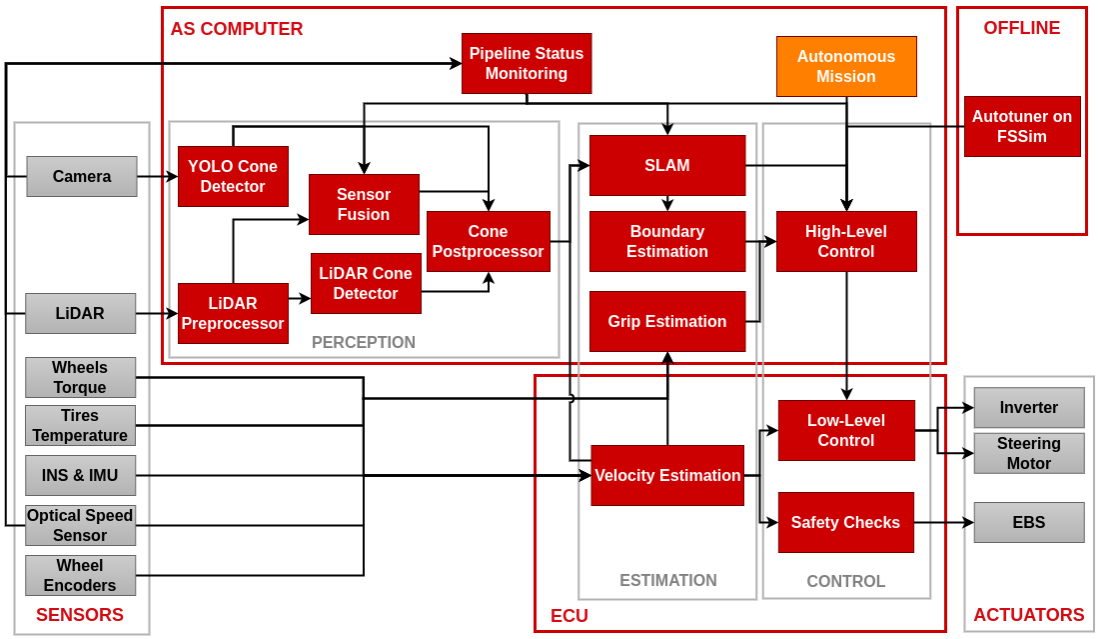

Figure 1.2: *Bernina*'s Autonomous System Pipeline

1.4 Related work

Constrained Bayesian Optimization has been a very active field of research in the past years and, therefore, a considerable amount of literature is available. In this section, the most relevant literature about constrained Bayesian Optimization for controller tuning and Bayesian Optimization for autonomous racing is reviewed.

In the context of Autonomous Racing, the authors of [1] present a method to tune the cost function weights of an MPC. Furthermore, they provide a method to learn the residual error of the model using Bayesian Linear Regression and they include context in the Bayesian Optimization. However, this work does not include constraints in the optimization formulation.

In [2] the authors present *SafeOpt*, an approach capable of optimizing a black-box function using Bayesian Optimization while satisfying a constraint on the surrogate model with a high probability. They achieved it by leveraging the properties of Gaussian Processes and making regularity assumptions on the black-box function. The authors of [3] extend *SafeOpt* by decoupling the objective function and constraints, as well as introducing multiple constraints into the problem. They apply their method to tune two parameters of a Linear Quadratic Regulator (LQR) to control the x position of a Quadrator. These two papers focus on optimizing simple problems or partially tuning a decoupled controller, whilst this project aims to extend these papers to tune a much more complex, completely coupled controller.

In [4] the authors propose to use Constrained Expected Improvement (CEI), an acquisition function presented in [5], to include constraints in the optimization of MPC parameters. However, they only tune the parameters of a simplified model and compare it to the real model that they use to run the simulation loop. Furthermore, using CEI as an acquisition function does not guarantee that the constraints will be satisfied and allows for violations of these.

Chapter 2

Methodology

Within the frame of hyper-parameter and controller parameters tuning, Bayesian Optimization arises as one of the preferred methods due to its low computational cost, its capability of representing nonlinear utility functions, and its successful performance in many black-box optimization tasks. However, the majority of the existing literature focuses on tuning simple controllers such as Proportional Integral Derivative (PID) or tuning just a small set of parameters of the MPC. This section aims to provide the formulation of the Bayesian Optimization approach used to tune any combination of MPC cost function weights, as well as to introduce a performance constraint in terms of lap time. Furthermore, the implementation details are provided at the end of the section.

2.1 Formulation

Bayesian Optimization consists of the sequential use of the Bayes Theorem to optimize continuous black-box functions, which are typically expensive to evaluate, without making any assumptions on their form (except for regularity and continuity). As the objective function is unknown, Bayesian Optimization uses a prior model to model the function and once it starts collecting data about the utility function, it keeps computing and updating the posterior of the model. Therefore, we need an "oracle" that can provide noisy evaluations of the objective function. Furthermore, the approach needs to make use of function modeling to model the black-box and be able to acquire new proposals of parameters. Finally, constraints on the optimization problem are introduced to provide performance guarantees for the system.

2.1.1 Bayesian Optimization workflow

The workflow of Bayesian Optimization consists of five different phases, as presented in the diagram of Figure 2.1, which consist of acquiring a new sample of the black-box via the "oracle", adding the sample to the data set, updating the posterior of the black-box's surrogate model, performing the optimization over the black-box surrogate model and acquiring a new proposal of parameters. Figure 2.2 presents the pipeline of this project, which contains the five different phases mentioned above. AMZ's simulator FSSim is used to act as the "oracle" and, given a set of cost function weights of the MPC, it provides information about the lap time achieved. Furthermore, the simulator interface keeps track of the number of knocked-down cones as well as the deviation of the reference trajectory.

The algorithm starts by trying a default set of parameters using the simulator. The simulation interface provides a lap time that includes the penalties of the knocked cones or deviation of the reference trajectory. This lap time sample is sent to the Bayesian Optimizer, which uses it to update the black-box surrogate model and consecutively perform the optimization to acquire a new set

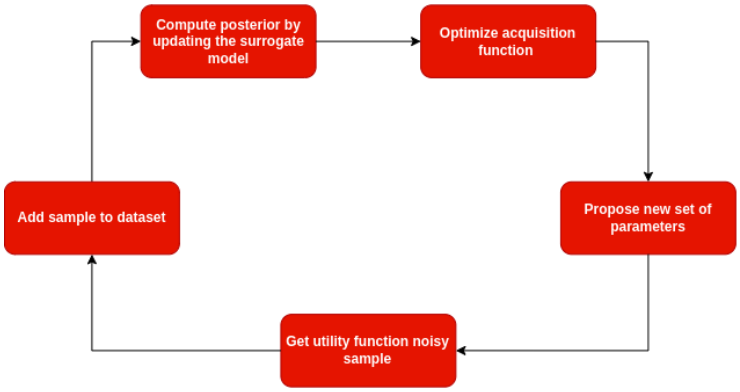

Figure 2.1: Bayesian Optimization diagram.

of parameters. This new set of parameters is sent to the MPC via the MPC Interface and a new simulation starts. The process is iterative and runs for a certain number of iterations defined by the user. At the end of the optimization, two sets of parameters are proposed as the optimal: the parameters that provided the best lap time sample and the parameters that minimize the mean of the surrogate model. These tow sets are again tried in simulation to evaluate their performance ans select the best one.

In order to be able to perform the optimization, an approximate model of the utility function and a procedure to optimize it are needed. This can be achieved via surrogate models and acquisition functions, which are introduced in the following sections.

2.1.2 Utility function surrogate model

Bayesian optimization uses surrogate models to place a belief on a black-box utility function $f(\boldsymbol{\theta})$ and update it with the noisy samples provided by the "oracle". In the case of this project, the surrogate function models the lap time provided by the simulator. Note that this lap time is perturbed by the noise of the perception and car actuators, which in turn are modeled in the simulator. We assume that the lap time samples are perturbed by i.i.d. Gaussian noise, thus we have that the sample at time t is $\tilde{f}_t(\boldsymbol{\theta}_t) = f(\boldsymbol{\theta}) + \epsilon_t$, where $\epsilon_t \sim N(0, \sigma^2)$. These samples are saved in the data-sets $\boldsymbol{y}_t = [\tilde{f}(\boldsymbol{\theta}_1)_1, \ldots, \tilde{f}(\boldsymbol{\theta}_t)_t]$ and $\boldsymbol{\Theta}_t = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_t]$.

Machine Learning is a powerful tool that provides different methods that can be used as surrogate models such as neural networks, nonlinear regression, or Gaussian Processes. In the Bayesian Optimization literature, Gaussian Processes are the most popular method to use as surrogate models due to their ability to model a wide variety of continuous nonlinear functions, their sample efficiency characteristic even in sequential approaches, their ability to model epistemic and aleatoric uncertainty and the possibility to derive high probability confidence bounds when the system is well calibrated. This last characteristic of Gaussian Processes becomes very important in the context of Bayesian Optimization, as it means that a new sample will lie within the model confidence bounds with a very high probability, as shown in Equation (2.1).

$$\mathbb{P}[|\mu_t(\boldsymbol{\theta}) - f(\boldsymbol{\theta})| \leqslant \beta_t(\delta)\sigma_t(\boldsymbol{\theta}) \,\forall\, t, \boldsymbol{\theta} \in D] \geqslant 1 - \delta \tag{2.1}$$

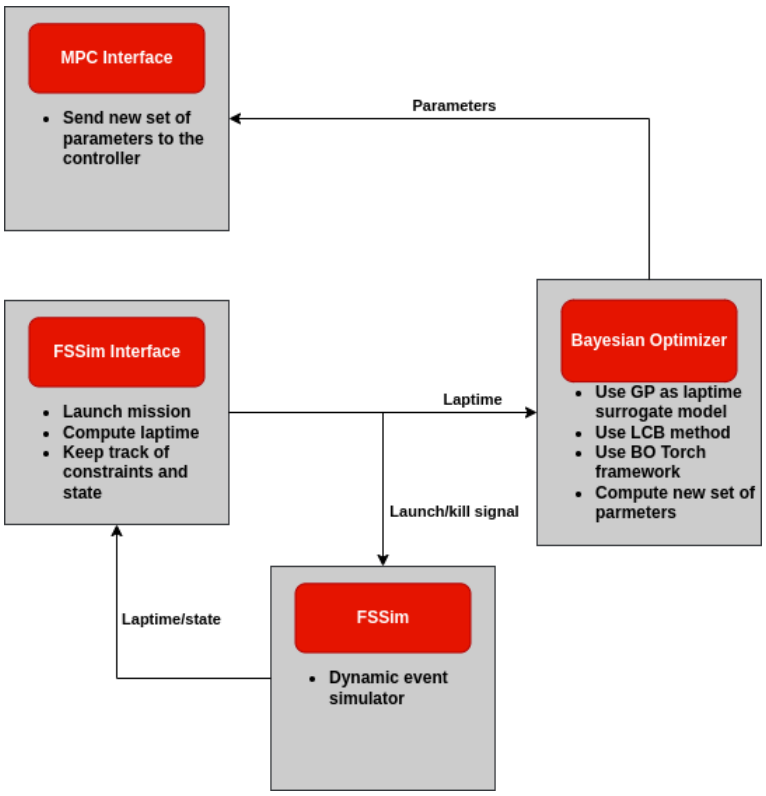

Figure 2.2: Bayesian Optimization for Autonomous Racing pipeline.

where $\mu_t(\boldsymbol{\theta})$ and $\sigma_t(\boldsymbol{\theta})$ are the mean and the variance of the Gaussian Process at time t, β_t is a parameter that models the confidence bounds, D is the domain of the parameters and δ is a low probability ($\delta \in [0, 1]$).

In this project, Gaussian Processes are chosen as the surrogate models of the lap time, $f \sim \mathcal{GP}(\mu(\boldsymbol{\theta}), k(\boldsymbol{\theta}, \boldsymbol{\theta}^{'}))$. The model assumes a zero mean prior and uses the lap time samples to compute its posterior. Similarly, a kernel $k(\boldsymbol{\theta}, \boldsymbol{\theta}^{'})$ is defined to model the covariance of the Gaussian Process. Equations (2.2) and (2.3) present the posterior calculation of the mean and kernel, respectively, given their prior and the new sample. A Matérn 5/2 kernel, Equation (2.4), is used in this project due to its ability to represent non-smooth nonlinear functions. Furthermore, due to the smoothess of the Matérn kernel, we can inherently assumes that f is L-Lipschitz continuous with respect to some metric d on D. This is a necessary condition to use the constrained Bayesian Optimization method presented in [2].

$$\mu_t(\boldsymbol{\theta}) = \boldsymbol{k}_t^T(\boldsymbol{\theta})(\boldsymbol{K}_t + \boldsymbol{I}_t\sigma^2)^{-1}\boldsymbol{y}_t \tag{2.2}$$

$$k_t(\boldsymbol{\theta}, \boldsymbol{\theta}^{'}) = k_t(\boldsymbol{\theta}, \boldsymbol{\theta}^{'}) - \boldsymbol{k}_t^T(\boldsymbol{\theta})(\boldsymbol{K}_t + \boldsymbol{I}_t\sigma^2)^{-1}\boldsymbol{k}_t(\boldsymbol{\theta}^{'}) \tag{2.3}$$

$$k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = \frac{1}{\Gamma(\nu)2^{\nu-1}}(\frac{\sqrt{2\nu}}{l}d(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j))^{\nu}K_{\nu}(\frac{\sqrt{2\nu}}{l}d(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)) \tag{2.4}$$

In Equations (2.4), (2.2), and (2.3) $\nu = 5/2$, $d(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$ is the euclidean distance between two parameter vectors, $K_{\nu}(\cdot)$ is a modified Bessel function, $\Gamma(\cdot)$ is the gamma function, l is the length-scale of the kernel, $\boldsymbol{k}_t(\boldsymbol{\theta}) = [k(\boldsymbol{\theta}_1, \boldsymbol{\theta}), \ldots, k(\boldsymbol{\theta}_t, \boldsymbol{\theta})]^T$, and $\boldsymbol{K}_t$ is the kernel matrix $[k(\boldsymbol{\theta}, \boldsymbol{\theta}^{'})]_{\boldsymbol{\theta}, \boldsymbol{\theta}^{'} \in D}$.

Figure 2.3 provides a visual representation of a 1-D Gaussian Process, where the samples, the mean, and the confidence bounds that represent the uncertainty are shown. It is worth noting that the uncertainty is lower in regions where more data is available and it grows in regions where there is no data.

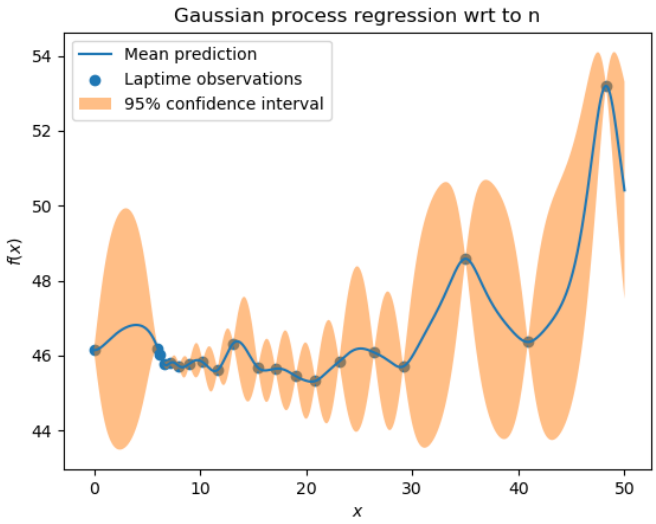

Figure 2.3: 1-D Gaussian Process surrogate model example. Uncertainty grows in regions of the parameter space where the amount of collected samples is low.

2.1.3 Acquisition function

An acquisition function models how the parameter space will be explored during the sequential optimization. This function models the exploration vs. exploitation trade-off to determine which areas of the parameter space are worth exploring and exploiting. Hence, the acquisition function increases in regions where the surrogate model $f(\boldsymbol{\theta})$ is optimal and in areas that have not been explored. Ideally, the acquisition function should balance being high at uncertain and optimal regions. In our formulation we want the contrary, we want the acquisition function to provide low values in optimal and unexplored areas and minimize over it. Therefore, the optimization problem follows Equation (2.5).

$$\boldsymbol{\theta}_p = \arg\min_{\boldsymbol{\theta}} \quad f_{aq}(\boldsymbol{\theta}) \tag{2.5}$$

where $f_{aq}(\boldsymbol{\theta})$ denotes the acquisition function.

There are many acquisition functions in the Bayesian Optimization literature, however, in this project, we focus on the Lower Confidence Bounds (LCB) acquisition function and an extension of it to introduce the performance constraint, *SafeOpt* [2], which is described in section 2.1.4. Although different acquisition functions were studied and tested, such as Expected Improvement and Constrained Expected Improvement [5], LCB and *SafeOpt* showed significantly better results compared to the other methods.

The LCB acquisition function follows Equation (2.6), which is the mean of the utility function surrogate model minus a trade-off parameter, β, times the variance of the surrogate model.

$$f_{LCB}(\boldsymbol{\theta}) = \mu(\boldsymbol{\theta}) - \beta\sigma(\boldsymbol{\theta}) \tag{2.6}$$

Therefore, this acquisition function potentiates zones in the parameter space where the mean of the surrogate model is low, which are potential optimal points, and zones where the variance is high, which are unexplored. The parameter β controls this exploration vs. exploitation trade-off, as if β is large (between 1.5 and 3) the optimization will tend to explore more the parameter space. For instance, Figure 2.4 presents an example of optimizing the Gaussian Process using the LCB as an acquisition function and a value for β of 1.95. This value was empirically chosen, as it proved to perfrom well in the 1-D and 2-D cases.

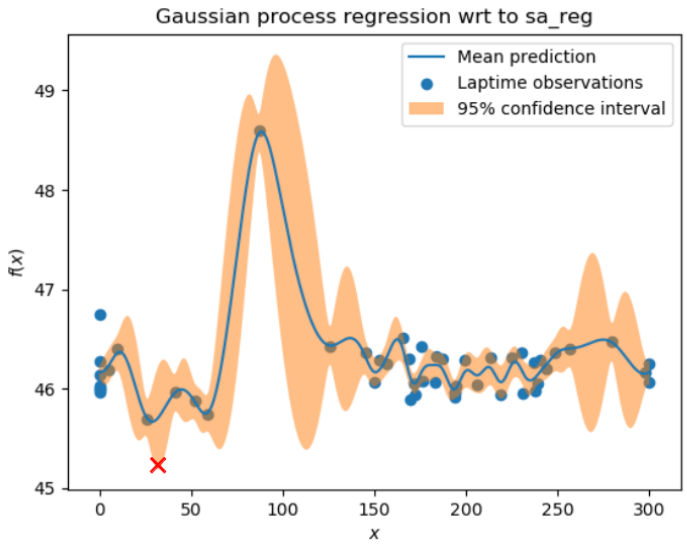

Figure 2.4: Gaussian Process surrogate model example. The red cross denotes the optimal point when using the Lower Confidence Bounds as an acquisition function and $\beta = 1.95$.

2.1.4 Constrained Bayesian Optimization

Unconstrained Bayesian Optimization is a very useful tool to tune controller parameters in simulated environments where the agent (an autonomous racing car in this project) is allowed to use parameters that can lead to a decrease in performance. In future iterations, Bayesian Optimization will avoid the use of these parameters, as they lead to poor performance. However, there will still exist a risk that the optimization explores regions of the parameter space that can again lead to a decrease in performance.

In order to provide performance guarantees at all times, constraints need to be present in the optimization process. In this project, the approach presented in [2], called *SafeOpt*, is followed to introduce constraints on the surrogate model. Thus, the optimization problem becomes the one shown in Equation (2.7). Note that this algorithm tries to find the maximum of the surrogate model, therefore we need to multiply by -1 the lap time samples used to train the model so that we can find the parameters that minimize the lap time.

$$\begin{aligned} \boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \quad & f(\boldsymbol{\theta}) \\ \text{s.t.} \quad & f(\boldsymbol{\theta}) \geqslant \tau \end{aligned} \tag{2.7}$$

The authors of [2] present an approach capable of finding optimal points while satisfying the safety constraint with high probability at all times. This algorithm solves the optimization problem of equation (2.8), which greedily selects the point of maximum uncertainty that is within the so-called maximizers (M_t) and expanders (G_t) sets. To understand the maximizers and expanders sets, first,

the safe set (S_t) needs to be defined. First, the algorithm assumes that the parameter space is discretized. Then, the safe set contains all the discrete points of the parameter space that satisfy the constraint. The safe set is initialized as S_0 and contains the initial point of the optimization, which must satisfy the constraints. We make use of the L-Lipschitz continuity assumption to ensure that the points nearby are safe even under the worst realization of the surrogate model. Only the points that satisfy this assumption are included in the safe set.

$$\boldsymbol{\theta}_t = \arg \max_{\boldsymbol{\theta} \in M_t \bigcup G_t} \quad w_t(\boldsymbol{\theta}) \tag{2.8}$$

In Equation (2.8) $w_t(\boldsymbol{\theta}) = u_t(\boldsymbol{\theta}) - l_t(\boldsymbol{\theta})$, where $u_t(\boldsymbol{\theta})$ and $l_t(\boldsymbol{\theta})$ are the upper and lower confidence bounds of the surrogate model, respectively. These confidence bounds are defined as $u_t(\boldsymbol{\theta}) = \mu_{t-1}(\boldsymbol{\theta}) + \beta_t \sigma_{t-1}(\boldsymbol{\theta})$ and $l_t(\boldsymbol{\theta}) = \mu_{t-1}(\boldsymbol{\theta}) - \beta_t \sigma_{t-1}(\boldsymbol{\theta})$.

As mentioned before, the M_t set contains the possible maximizers within the safe set S_t. This set is defined in equation (2.9), where it is shown that the set only includes points of the safe set whose upper confidence bound is larger than the maximum lower bound. Furthermore, the expanders set G_t, defined in equation (2.10), contains all the points within the safe set that when evaluated with the auxiliary function $g_t(\boldsymbol{\theta})$ return a positive value. This auxiliary function uses the L-Lipschitz continuity assumption to provide new points that with a high probability will be inside the safe set.

$$M_t = \{\boldsymbol{\theta} \in S_t \,|\, u_t(\boldsymbol{\theta}) \geqslant \max_{\boldsymbol{\theta}' \in S_t} l_t(\boldsymbol{\theta}')\} \tag{2.9}$$

$$\begin{aligned} G_t &= \{\boldsymbol{\theta} \in S_t \,|\, g_t(\boldsymbol{\theta}) > 0\} \\ g_t(\boldsymbol{\theta}) &= |\{\boldsymbol{\theta}' \in D \setminus S_t \,|\, u_t(\boldsymbol{\theta}) - Ld(\boldsymbol{\theta}, \boldsymbol{\theta}') \geqslant \tau\}| \end{aligned} \tag{2.10}$$

where L is the Lipschitz constant and $d(\cdot, \cdot)$ is a metric on D.

The *SafeOpt* algorithm trades-off expanding the safe set and exploiting the possible maximums within it to find the reachable optimum point. Note that the algorithm relies on the Lipschitz continuity and the Gaussian Process confidence bounds property, which ensures that the real black-box function will lie within the surrogate model confidence bounds with a high probability.

The approach of introducing a constraint works well in a reduced set of cost function weights. However, the method starts to become infeasible when trying to tune a larger number of parameters due to its scalability issue. This is due to the discretization of the parameter space which escalates exponentially. Although there exist some methods to escalate the approach, such as using particle swarms and relying on heuristics [6], these methods did not produce good results in this project setup when tuning a large number of parameters and they did not produce better results than *SafeOpt* even when a reduced number of parameters as tuned. Thus, the constrained Bayesian Optimization is limited to tuning 1 to 3 cost function weights. To achieve the best performance, the most sensitive parameters are chosen to be tuned with *SafeOpt*. These are the cost of the deviation from the reference line (w_n), the cost of the reference longitudinal velocity tracking (w_{v_x}), and the cost of the derivative of the rear and front longitudinal tire forces ($w_{\dot{F}_{x,R}}$, $w_{\dot{F}_{x,F}}$).

2.2 Implementation

This section describes the implementation of the Bayesian Optimization method. The three main blocks of the algorithm are the interface with the simulator, the Bayesian Optimization module, and the addition of constraints to the problem. The whole pipeline of the algorithm was implemented in

Python and it uses the Robot Operating System (ROS) library to communicate with the simulator and the MPC.

2.2.1 Simulator interface

The simulator interface handles the communication with the simulation and keeps track of the state of the car. The interface triggers the simulation to try a new set of parameters and ends it in the following cases:

- The desired number of laps are completed.
- The car gets stuck and cannot recover. A high lap time is applied.
- The car goes completely out of track. A high lap time is applied.

Furthermore, this interface handles the penalties applied for knocking down cones which are added to the total lap time. In the early development stages of the algorithm, these penalties were applied discretely with respect to the state of the car, in such a way that a penalty of 2 seconds was added for every cone that the envelope of the car came in contact with. This approach was adopted to simulate a real Formula Student Driverless event as much as possible, as the rules of the competition state that a knocked-down cone adds a penalty of two seconds to the lap time. However, it was found that applying these discrete penalties caused very steep and abrupt changes in the real black-box utility function. Hence, the samples of lap time extracted from the simulation were falling out of the surrogate model confidence bounds with a considerable frequency. This was not a problem when using unconstrained Bayesian Optimization, as the Gaussian Process was able to efficiently update its posterior and change its mean and confidence bounds accordingly. However, this behavior became a problem in the constrained Bayesian Optimization case, as points that were violating the constraint were frequently added to the safe set and as a consequence, it generated instability in the optimization.

For the aforementioned reason, a continuous penalty function for knocking down cones was introduced. The main concept of this penalty function is the use of the sum of linear functions ($f_{dev\_penalty}(n) = 0.075n$) with respect to the deviation from the reference trajectory as a penalty function. At each point in time, the deviation is extracted and the linear function is evaluated and added to the penalty value. The linear function provides low penalties when the deviation is close to zero and increases linearly providing higher values when the deviation increase. The slope was selected as $m = 0.075$ to smoothen the effect of penalties. Figure 2.5 presents the linear function used in this project.

The approach of applying linear penalties with respect to the deviation instead of discrete penalties smoothens the shape of the black-box lap time function and allows *SafeOpt* to perform better and satisfy the safety constraint at all times.

2.2.2 Bayesian Optimization

The Bayesian Optimization module implements the unconstrained optimization described in Section 2.1.1. It makes use of the surrogate model and acquisition function described in Sections 2.1.2 and 2.1.3, respectively.

Algorithm 2.1 below summarizes the implementation of the Bayesian Optimization module described in Section 2.1.1. This algorithm iteratively calls the simulator with a proposed set of parameters and acquires a lap time sample. This lap time is used to update the surrogate model posterior to consecutively obtain the parameters that minimize the LCB. Furthermore, both the parameters and the lap time samples are pre-processed to stabilize the training of the Gaussian Process and, hence, improve the performance of the optimization. Parameters are scaled to be between 0 and 1 so that $\boldsymbol{\theta} \in [0, 1]^d$, where d is the number of parameters. This is done to to leave the length scale of the Gaussian Process as an optimizable hyper-parameter. By scaling the parameters between 0 and 1 the optimization of the length scale improves. Regarding the lap time

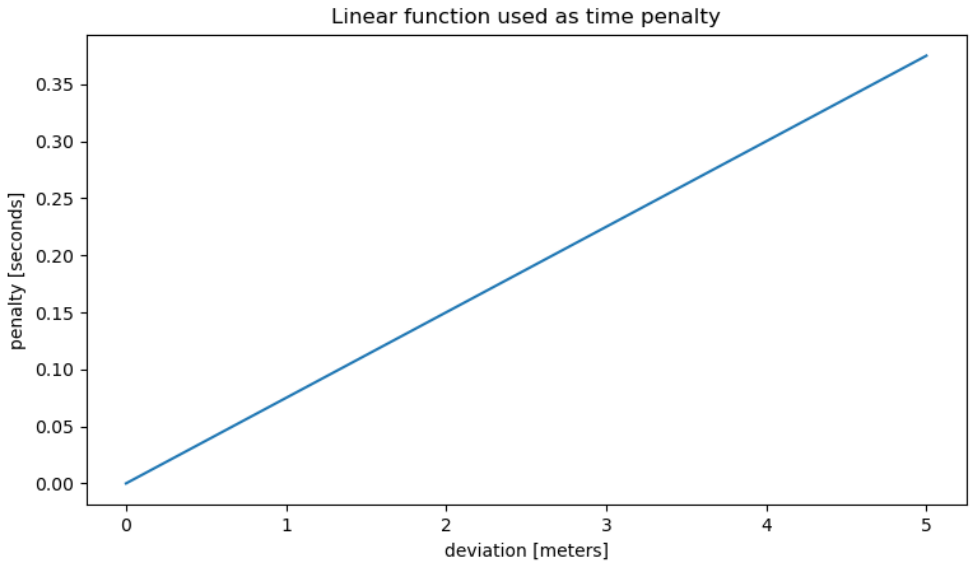

Figure 2.5: Linear penalty function used to penalize deviation from the reference trajectory.

samples, they are first normalized using the optimal lap time, given a track, and, afterward, they are standardized to a normal distribution of zero mean and unit variance. The mean and variance of the standarization is computed every N iterations of the algorithm.

Algorithm 2.1 Bayesian Optimization

Input $\boldsymbol{\theta}_0$, $f \sim \mathcal{GP}(\mu_0(\boldsymbol{\theta}), k_0(\boldsymbol{\theta}, \boldsymbol{\theta}^{'}))$, D, f_{opt}, β

$y_0 \leftarrow \tilde{f}(\boldsymbol{\theta}_0)/f_{opt}$ ▷ Run simulation with an initial set of parameters and normalize
$\boldsymbol{\Theta}[0] \leftarrow normalize(\boldsymbol{\theta}_0, D)$ ▷ Normalize parameters between 0 and 1
for $t = 1, \ldots$ **do**
  if $t\%N == 0$ **then**
    $m \leftarrow mean(y_0 \ldots y_t)$ ▷ Every N iterations recompute mean and variance of samples
    $v \leftarrow variance(y_0 \ldots y_t)$
  end if
  $\tilde{\boldsymbol{y}}_t \leftarrow standardize(\boldsymbol{y}_t, m, v)$ ▷ Standardize samples to 0 mean and unit variance
  $f \sim \mathcal{GP}(\mu_t(\boldsymbol{\theta}), k_t(\boldsymbol{\theta}, \boldsymbol{\theta}^{'})) \leftarrow Update(\boldsymbol{\Theta}, \tilde{\boldsymbol{y}}_t)$ ▷ Update model posterior
  $\boldsymbol{\theta}_t \leftarrow \arg\min_{\boldsymbol{\theta} \in [0,1]} \quad \mu_t(\boldsymbol{\theta}) - \beta\sigma_t(\boldsymbol{\theta})$ ▷ Optimize LCB
  $\boldsymbol{\Theta}[t] \leftarrow \boldsymbol{\theta}_t$
  $y_t \leftarrow \tilde{f}(denormalize(\boldsymbol{\theta}_t, D))/f_{opt}$ ▷ Run simulation and normalize
end for

The training of the Gaussian Process surrogate model is implemented using GPyTorch [7], a GP Python library implemented with PyTorch [8]. In addition, the optimization step is implemented using BoTorch [9], an efficient Bayesian Optimization library implemented in PyTorch that provides multiple optimization methods and acquisition functions.

2.2.3 Constrained Bayesian Optimization

The implementation of constrained Bayesian Optimization follows the formulation presented in Section 2.1.4 and the algorithm is shown in Algorithm 2.2. The *SafeOpt* framework is used to introduce a constraint on the lap time. This constraint ensures that the lap time will always be

below a scaling factor (τ_{scale}) times the initial lap time. The initial lap time is acquired using a set of parameters that guarantee stability and baseline performance. Thus, this constraint guarantees that the car will perform in terms of lap time and save useful testing time.

Algorithm 2.2 Constrained Bayesian Optimization

Input S_0, $f \sim \mathcal{GP}(\mu_0(\boldsymbol{\theta}), k_0(\boldsymbol{\theta}, \boldsymbol{\theta}^{'}))$, τ_{scale}, D, f_{opt}

1: $\boldsymbol{\theta}_0 \leftarrow S_0$
2: $y_0 \leftarrow -\tilde{f}(\boldsymbol{\theta}_0)/f_{opt}$ ▷ Run simulation with an initial set of parameters and normalize
3: $\tau \leftarrow \tau_{scale} y_0$ ▷ Set the performance constraint
4: **for** $t = 1, \ldots$ **do**
5: $f \sim \mathcal{GP}(\mu_t(\boldsymbol{\theta}), k_t(\boldsymbol{\theta}, \boldsymbol{\theta}^{'})) \leftarrow Update(\boldsymbol{\theta}_{t-1}, y_{t-1})$ ▷ Update model posterior
6: $S_t \leftarrow \bigcup_{\boldsymbol{\theta} \in S_{t-1}} \{\boldsymbol{\theta}^{'} \in D | \max(\tau, l_t(\boldsymbol{\theta})) - Ld(\boldsymbol{\theta}, \boldsymbol{\theta}^{'}) \geqslant \tau\}$ ▷ Update safe set
7: $G_t \leftarrow \{\boldsymbol{\theta} \in S_t | g_t(\boldsymbol{\theta}) > 0\}$
8: $M_t \leftarrow \{\boldsymbol{\theta} \in S_t | u_t(\boldsymbol{\theta}) \geqslant \max_{\boldsymbol{\theta}' \in S_t} \max(\tau, l_t(\boldsymbol{\theta}^{'}))\}$
9: $\boldsymbol{\theta}_t \leftarrow \arg\max_{\boldsymbol{\theta} \in M_t \bigcup G_t} \quad w_t(\boldsymbol{\theta})$ ▷ Perform greedy optimization
10: $y_t \leftarrow -\tilde{f}(\boldsymbol{\theta}_t)/f_{opt}$ ▷ Run simulation and normalize
11: **end for**

As previously mentioned in the formulation section, the lap time samples need to be multiplied by -1 to be able to minimize the black-box function, as the algorithm tries to find a maximum of a given surrogate model. This is done in step 2 in Algorithm 2.2, together with the normalization of the lap time. Note that in this implementation neither the samples are standardized nor the parameters are normalized. The samples cannot be standardized as we want to use a constant constraint that is related to the initially normalized lap time. Furthermore, the parameters are not normalized as this did not show any improvement, on the contrary, the results were worse than when not applying the normalization. This behavior might be the consequence of using GPy to implement the Gaussian Process. On the other hand, using GPyTorch benefits from normalization.

The training of the surrogate model is implemented using the GPy library [10], as the SafeOpt library requires the use of GPy. The optimization of the surrogate model is implemented using the *SafeOpt* library and the domain is discretized in 200 points.

Chapter 3

Results

This chapter presents the results of the optimization methods presented in this project, namely Bayesian Optimization and its constrained variant. Simple regret and cumulative regret over time (see Equation (3.1)) are used as the main metrics to test the performance. Instant regret represents the difference between the current lap time compared to the optimal lap time, whilst cumulative regret over time is the mean of the instant regrets at time t. The target performance is to make the cumulative regret over time converge to the lowest possible value, as this means that the optimization has found the parameters that provide a lap time close to the optimal under our modelling assumptions. As for the instant regret, it provides an intuition of how the current iteration performs and how far away is from the optimal lap time. Furthermore, the minimum lap time sample is also taken into account, as well as the number of constraint violations and the sampling process for the case of constrained Bayesian Optimization.

$$\begin{aligned} r(t) &= \tilde{f}(\boldsymbol{\theta_t}) - f_{opt} \\ R(t) &= \frac{1}{t}\sum_{k=1}^{t} r(k) \end{aligned} \tag{3.1}$$

3.1 Bayesian Optimization

3.1.1 Regrets and lap time improvement

In order to evaluate the effectiveness of Bayesian Optimization using LCB as acquisition function, we run the algorithm using the whole parameter space, thus the parameter vector $\boldsymbol{\theta}$ resembles the vector of Equation (1.6). The optimization is able to reduce the lap time by almost 2 seconds, which is an impressive improvement in autonomous racing. Table 3.1 presents the improvement in lap time concerning the number of iterations of the optimization. Even though the lap time keeps decreasing with the number of iterations, as the parameter vector contains more than 30 parameters, the optimization needs a high number of iterations to find global optimums. As the number of parameters decreases, the optimization requires fewer iterations to converge to an optimal point.

Iteration	1^{st} lap	2^{nd} lap	3^{rd} lap	4^{th} lap	Average
Initial	24.65	24.34	24.36	24.28	**24.41**
10	24.48	24.22	24.24	24.19	**24.28**
50	23.59	23.41	23.32	23.36	**23.42**
100	23.24	23.03	22.95	22.98	**23.05**
450	22.71	22.62	22.51	22.48	**22.58**

Table 3.1: lap times and average lap time (in seconds) achieved with the parameters selected by Bayesian Optimization at certain iterations.

Figure 3.1 presents the regret plots achieved with Bayesian Optimization. The method achieves sublinear cumulative regret as seen in the top figure. Furthermore, it is converging to low cumulative regrets which indicates that the parameters are converging to an optimal point. Note that the regret cannot converge to 0, as the model used to compute the optimal lap time is simple compared to the high-fidelity model used in the simulation. Thus, the optimal lap time overestimates the capacities of the race car.

In Figure 3.1 the instant regret is also visualized. It can be seen that, on average, the optimization selects parameters that produce low regrets. However, some spikes are present in the regret. These spikes are the result of the Bayesian Optimization trying to explore new points with little or no samples around them. This exploration can sometimes lead to high lap times or the car going out of the track, which is penalized with a high lap time.

Another interesting feature that can be seen in Figure 3.1 is that the instant regret suddenly goes down on average around the 300th iteration. This is the result of exploring new regions that had little samples around them that led to an improvement in terms of lap time.

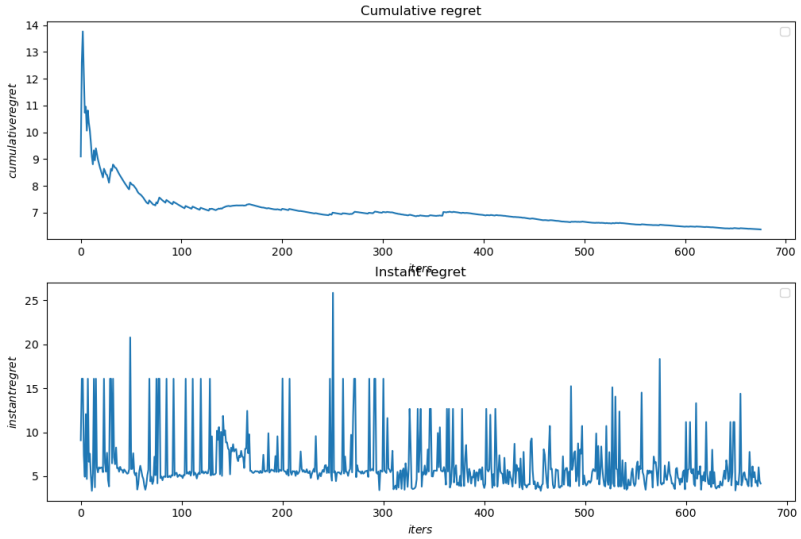

Figure 3.1: Regret plots using Bayesian Optimization. Top: cumulative regret over time. Bottom: instant regret.

Besides the regret plots, it is important to account for the number of knocked-down cones at each iteration and the evolution throughout the optimization. Figure 3.2 presents the number of knocked-down cones with respect to the iterations. On average, the number of knocked-down cones decreases with the number of iterations. This result makes sense, as each cone down adds a lap time penalty, which is what we are trying to minimize by using Bayesian Optimization.

Results presented in Figures 3.1 and 3.2 focus on a specific instance of Bayesian Optimization and, therefore, do not represent the average performance. To be able to evaluate the average

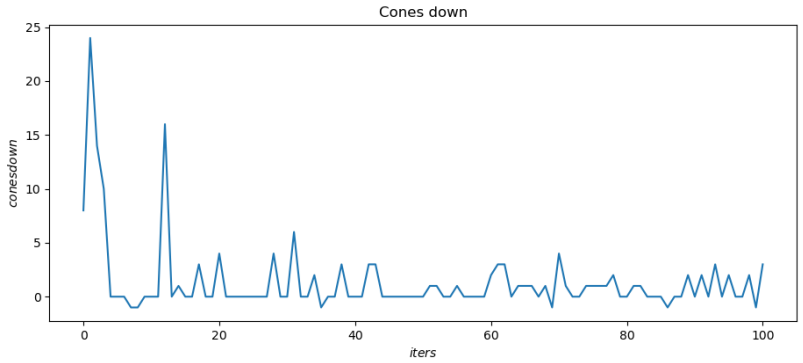

Figure 3.2: Number of cones down at each iteration of the optimization.

performance, average regrets are introduced. Figure 3.3 presents the mean and standard deviation of the cumulative regret computed with 7 evaluations of the algorithm for 95 iterations. From this figure, it can be inferred that on average the algorithm always achieves sub-linear regret, although the steepness of the regret changes depending on the evaluation as shown by the variance. Furthermore, from the average and the deviation of the number of cones down displayed in Figure 3.3 it is clear that the number of cones down quickly decreases over time and approaches zero.

3.1.2 Parameter convergence

The previous section was devoted to presenting the results of the optimization in terms of lap times. However, it is also important to identify if the parameters are converging to a value within acceptable flickering bounds. Figure 3.4 shows the values of a reduced number of parameters with respect to the optimization iterations. As seen in the figures, the parameters converge to a value, which represents the optimum. The flickering is related to the fine-tuning of Bayesian Optimization, which keeps optimizing to fine-tune the parameters.

Plots in Figure 3.4 also show the behavior at the beginning of the optimization process, with parameters jumping around and having a considerable amount of flicker. This phenomenon is due to the initial lack of data, which in turn makes the Bayesian Optimization sample at uncertain points that practically comprise the whole parameter space. Thus, this lack of data leads to jumps in the parameter space. This initial flickering phase increases with the size of the state space. This initial flickering is natural in BO as it will explore more at the begining and provide better results at the end. In addition, some spikes occasionally appear on these convergence plots, which are associated with either the algorithm trying to explore a completely different region or the optimizer not being able to solve the problem at that particular iteration and, hence, providing a value that is not optimum.

3.1.3 Surrogate model

To validate the performance of the surrogate model, Bayesian Optimization is run to tune 1 or 2 parameters only so that the Gaussian Processes can be visualized. With this method, the capacity of the model to represent the data and underlying black-box function can be evaluated. Figures 3.5 and 3.6 present the surrogate models at different stages of the optimization for the 1-D and 2-D cases, respectively. It is visible that the surrogate model is capable of modeling the lap time function and therefore the Bayesian Optimization can produce the expected results which were introduced in the previous sections. Even though the data is disturbed with noise, the variance of the Gaussian Process can capture and model this aleatoric noise, this is visible in the 1-D case where the variance can be easily plotted together with the mean and confidence bounds. Furthermore, it is visible that the LCB method is sampling at potential minimums, which is the desired behavior

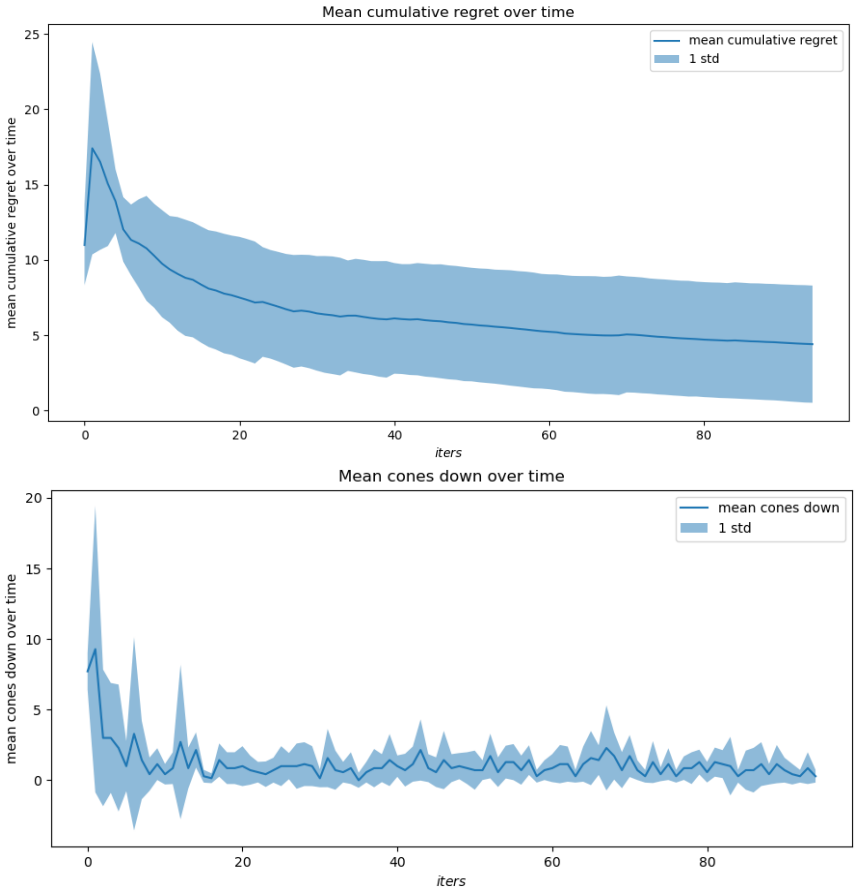

Figure 3.3: Average regret over time and cones down of Bayesian Optimization.

3.2 Constrained Bayesian Optimization

3.2.1 Surrogate model

As mentioned in Section 2.1.4, the *SafeOpt* Bayesian Optimization is evaluated with 1 or 2 parameters only. However, to evaluate the mean and confidence bounds of the Gaussian Process and the sampling of the algorithm only the 1-D case is presented in this section, while the other case is evaluated using the regret plots and constraint violations. Figure 3.7 shows how the algorithm samples the new possible maximizers or expanders that maximize the variance within the safe set in the case of tuning $\boldsymbol{\theta} = w_{v_x}$ and setting a constraint of $\tau_{scale} = 1.07$. In the beginning, the algorithm tries to quickly expand the safe set due to the Gaussian Process shape, but it realizes that a constraint is violated and therefore it reduces the safe set. At this point, the algorithm expands the safe set until the point where the lower bound of the surrogate model reaches the constraint. Thus the algorithm finds the maximizers inside this safe zone. As the shape of the Gaussian Process changes with the samples, the algorithm may expand but it is limited by the sample in the unsafe zone, which constrains the Gaussian Process shape. Furthermore, there is another safe zone in the region $[125, 200]$. However, the algorithm cannot sample in that zone because it expands using the Lipschitz continuity assumption and, as all safe set samples are in regions $[0, 60]$, the unsafe zone prohibits expanding to zone $[125, 200]$.

3.2.2 Regrets and constraint violations

In this section, various regret plots of the 1-D and 2-D cases are presented, together with the lap times. Figure 3.8 and 3.9 present the regret achieved with *SafeOpt* in the 1-D case by tuning w_{v_x} and w_n, respectively, as well as the lap time. Furthermore, the number of constraint violations is shown in these figures. From the regrets, it can be inferred that the algorithm achieves sub-linear regret while satisfying the constraint at almost all times. Furthermore, if a constraint is violated, as in the case of Figure 3.8, the safe set quickly adapts, as well as the Gaussian Process shape, to avoid further violations. In the figures, the convergence of the parameters can also be inferred and it can be concluded that the parameters converge to an optimal point after a few iterations of the optimization.

For the 2-D case, Figure 3.10 presents the cumulative regret, the lap time, and the number of violations. As in the case of 1-D, *SafeOpt* achieves sub-linear regret and satisfies the constraints, except for two points where the car hits too many cones, as seen in the lap time plot. In the beginning, the cumulative regret increases due to the expansion of the safe state that yield high lap times, but after some iterations, the surrogate model captures this behavior and the regret starts going down in a sub-linear fashion. The most sensitive parameters (w_{v_x}, w_n) are chosen to conduct the experiments to properly test the algorithm.

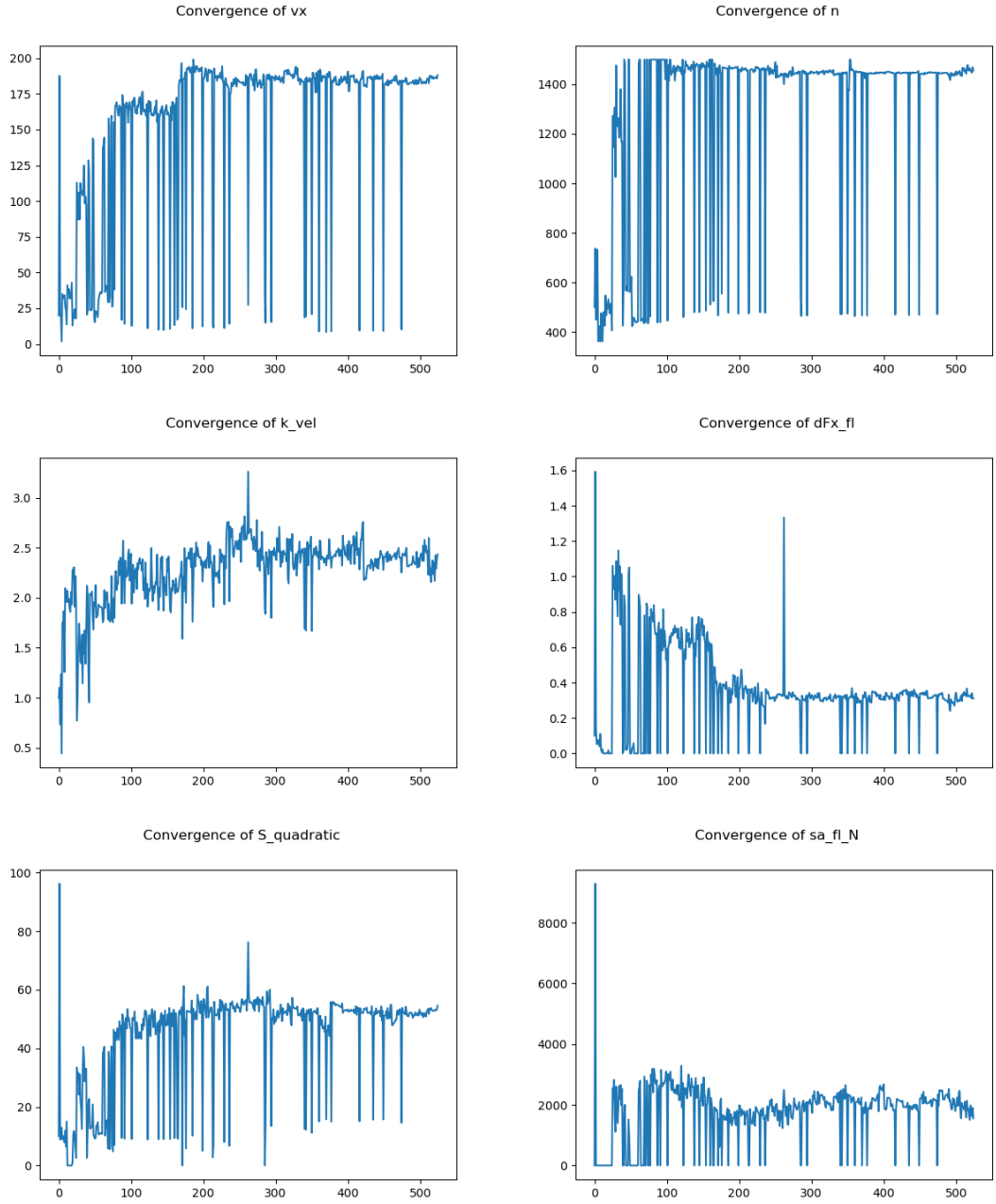

Figure 3.4: Values of w_{v_x}, w_n, $w_{k_{vel}}$, $w_{\dot{F}_{x\_fl}}$, w_{S_2}, $w_{\alpha_{fl\_N}}$ throughout the iterations of Bayesian Optimization.

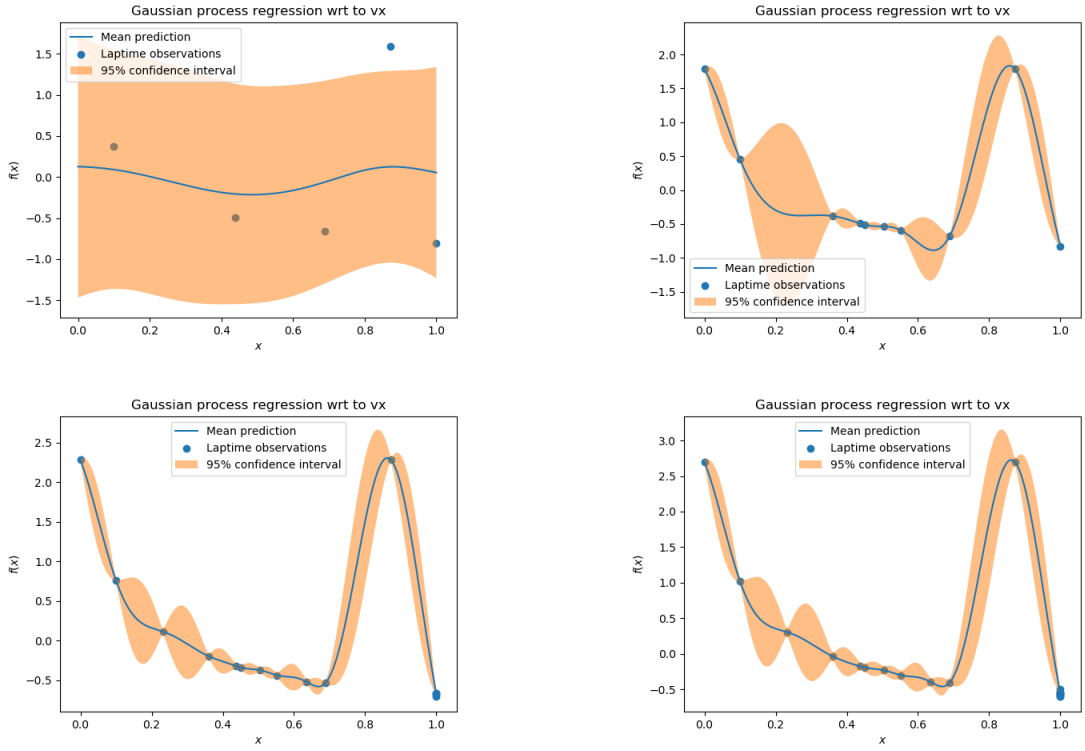

Figure 3.5: Gaussian process of lap time with respect to w_{v_x} with 5, 10, 15, and 20 samples.

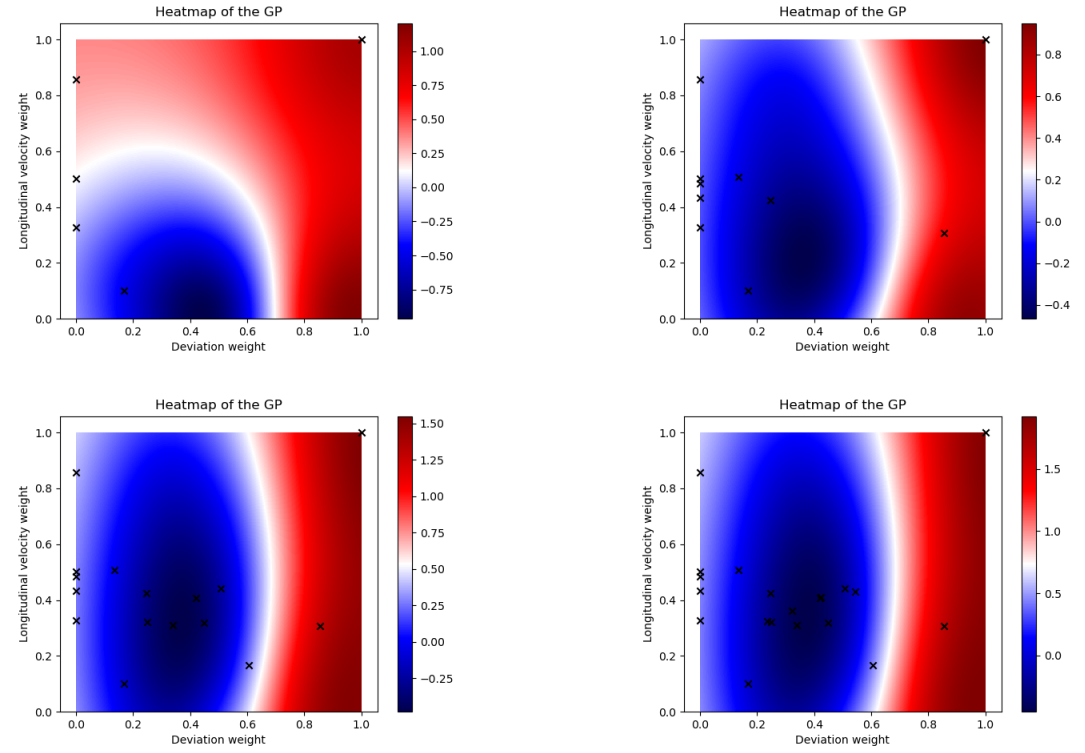

Figure 3.6: Gaussian process of lap time with respect to w_{v_x} and w_n with 5, 10, 15, and 20 samples.

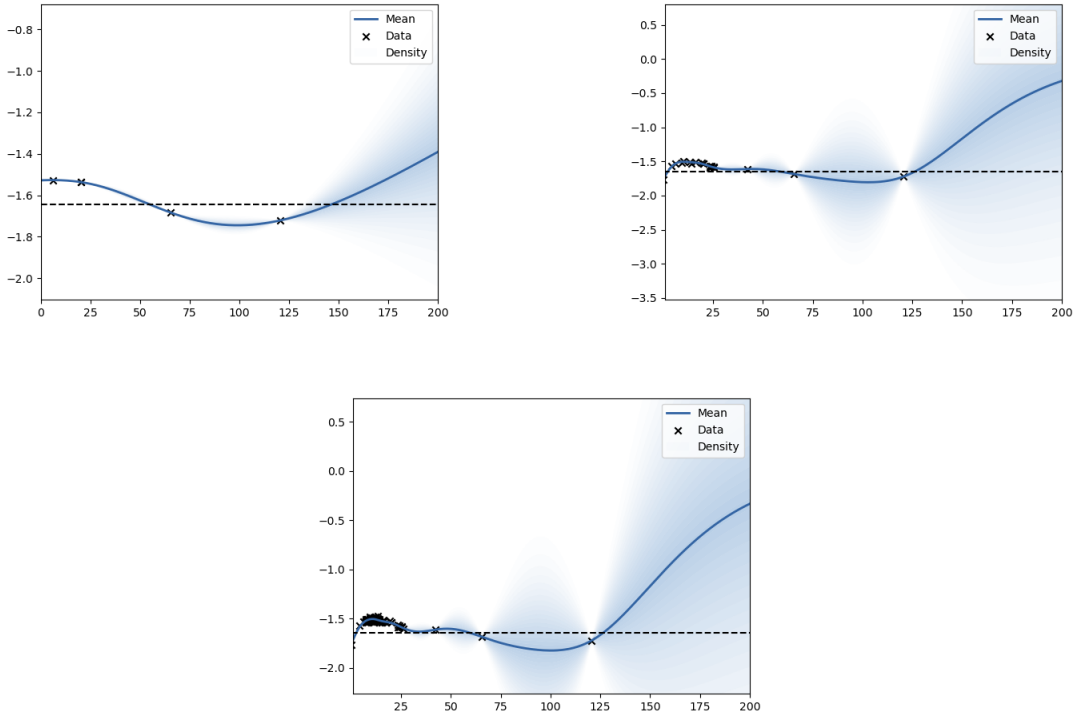

Figure 3.7: Gaussian Process, samples and constraint at iteration 5, 20, and 100 of w_{v_x} tuning

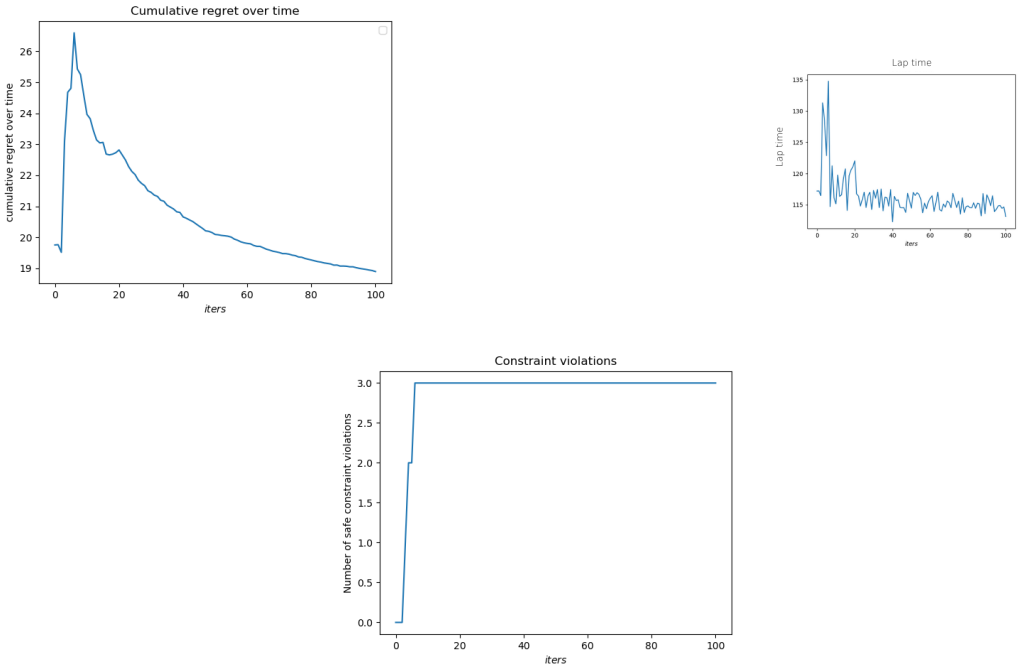

Figure 3.8: Cumulative regret, lap time, and number of safe constraint violations over time of w_{v_x}

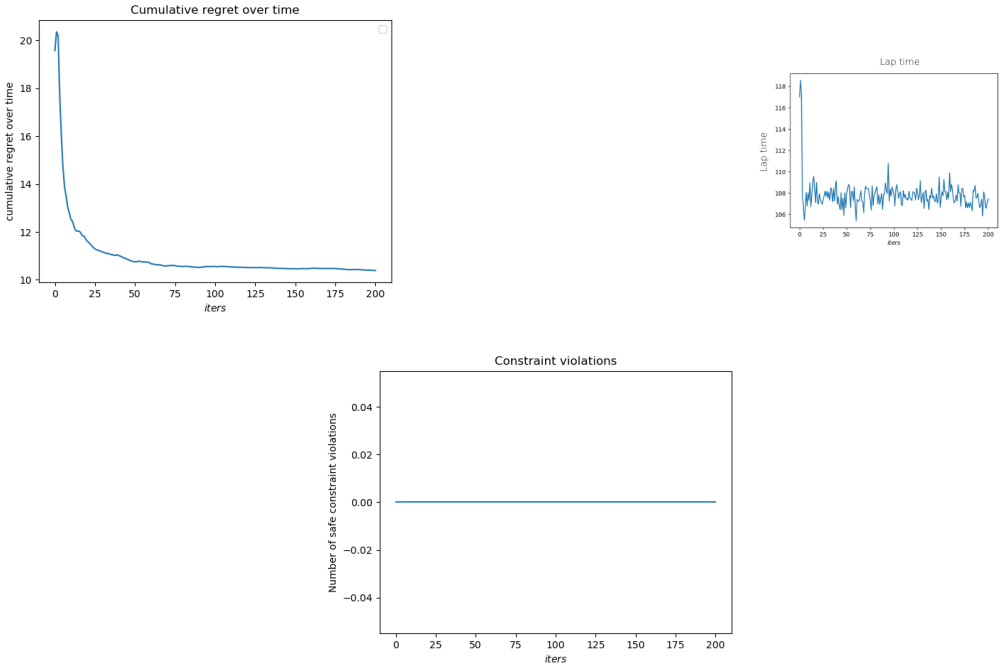

Figure 3.9: Cumulative regret, lap time, and number of safe constraint violations over time of w_n

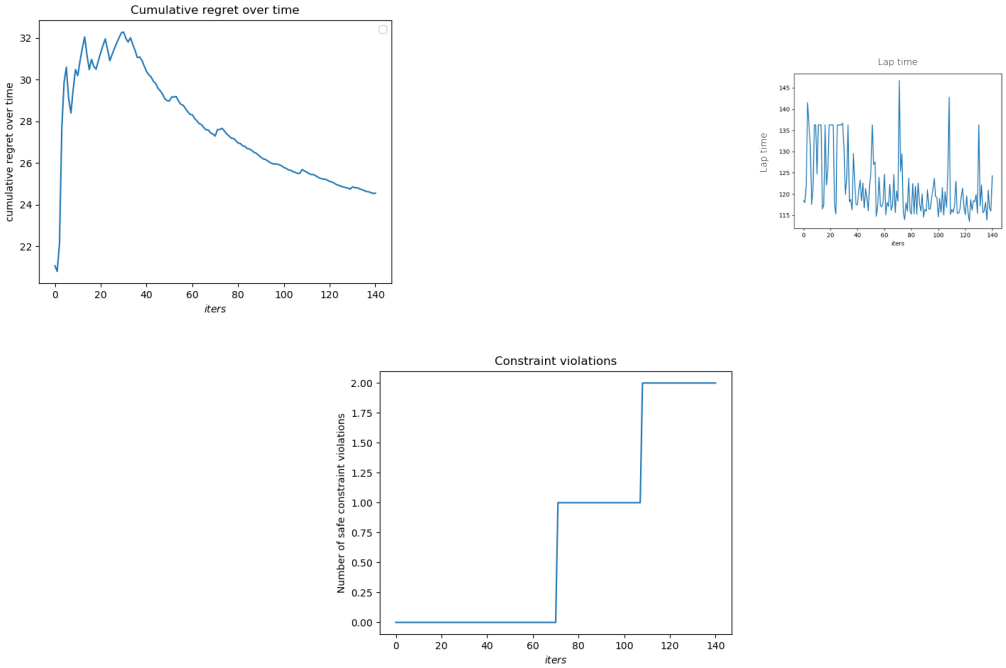

Figure 3.10: Cumulative regret,lap time, and number of safe constraint violations over time of w_{v_x}, and w_n

Chapter 4

Conclusion and future work

4.1 Conclusion

This project aims to use Bayesian Optimization to find the cost function weights of an MPC used in an autonomous racing environment that maximize the performance of the race car, which is equivalent to minimizing the lap time. The tuned weights are used for AMZ's prototype race car, *Bernina*, at the Trackdrive and Skidpad disciplines of Formula Student. The goal is to achieve the best possible lap time while satisfying a performance constraint. This constraint is formulated in terms of lap time so that the race car always achieves a lap time lower than the constraint.

In the first approach, unconstrained Bayesian Optimization is used to tune the whole set of cost function weights in an offline manner and rely on a high-fidelity simulator to provide the lap times and the state of the car at all times, which is used to add a time penalty for every knocked-down cone. A Gaussian Process is used as a surrogate model to approximate the lap time black box function and the Lower Confidence Bounds acquisition function is minimized to acquire new proposals of parameters.

The unconstrained Bayesian Optimization approach achieves sub-linear regret and is capable to reduce the lap time by two seconds compared with the initial hand-tuned parameters. Furthermore, the results show that the number of cones being knocked down decreases over the iterations. However, the number of iterations needed to achieve such a lap time improvement increases due to the large parameter space, which contains more than 30 parameters.

Constrained Bayesian Optimization is introduced in an attempt to be able to tune a reduced number of parameters in an online manner with the real platform in the near future. To be able to tune with the real platform, a performance constraint to decrease the testing time is introduced. This performance constraint is introduced in the lap time surrogate model in order to make sure that the lap time will always stay below a certain threshold. This constraint usually avoids knocking down cones and going out of the track, as these are penalized with higher lap times.

Safe Bayesian Optimization is tested with the 1-D and 2-D cases using the most sensitive parameters. *SafeOpt* [2] is used to implement the safe Bayesian Optimization to tune the MPC weights. The approach achieves sub-linear regret and the constraints are rarely violated. These results validate the safe approach and open the door to online testing of the algorithm in the near future. However, constrained Bayesian Optimization is only useful to tune a reduced amount of parameters due to its scalability issues, as the parameter space needs to be discretized. Thus, the methodology to obtain the maximum performance of the MPC in terms of lap time is to run the unconstrained Bayesian Optimization offline in simulation and, then, to use the results as initial parameters of

the safe Bayesian Optimization to fine-tune the most sensitive parameters in an online manner with the race car.

4.2 Future works

The main issues of the constrained Bayesian Optimization approach are the impossibility to scale to a larger set of parameters, as well as the unavailability of exploring possible global optimums due to the existence of disjoint safe parameter spaces. Furthermore, the tuning of the cost function weights of the MPC depends on external factors such as the tire temperature and the grip conditions. These external factors bring context into the Bayesian Optimization and can be taken into account if they can be measured. By taking the contextual factors into account, the optimization becomes more robust and generalizes better against these external factors.

The first approach to tackle scalability would be to implement the method used in [11], which decomposes the global optimization into a sequence of 1-D optimizations, which are more efficient to solve. However, this method is still vulnerable to disjoint safe spaces.

Another way to introduce scalability would be to use the method proposed in [12]. Besides providing scalability, this approach tackles the issue of not being able to explore disjoint safe parameter spaces, which we encounter with *SafeOpt.* The method leverages the Markov property of the dynamic system state to learn backup policies without actively exploring the parameter space and uses these policies to provide safety when evaluating policies outside the safe set.

Contextual information could be used achieve probably optimal behavior with respect to the context, which yields an overall better performance of the optimization. This method relies on adding a kernel to the Gaussian Process surrogate model to capture the contextual information, as presented in [13]. Furthermore, this method has proven to work in the case of MPC tuning for autonomous racing for modeling contexts such as weather conditions [1].

Bibliography

[1] L. P. Fröhlich, C. Küttel, E. Arcari, L. Hewing, M. N. Zeilinger, and A. Carron, "Model learning and contextual controller tuning for autonomous racing," *ArXiv*, vol. abs/2110.02710, 2021.

[2] Y. Sui, A. Gotovos, J. Burdick, and A. Krause, "Safe exploration for optimization with gaussian processes," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 997–1005. [Online]. Available: https://proceedings.mlr.press/v37/sui15.html

[3] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with gaussian processes," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE Press, 2016, p. 491–496. [Online]. Available: https://doi.org/10.1109/ICRA.2016.7487170

[4] F. Sorourifar, G. Makrygirgos, A. Mesbah, and J. Paulson, "A data-driven automatic tuning method for mpc under uncertainty using constrained bayesian optimization," *IFAC-PapersOnLine*, vol. 54, pp. 243–250, 01 2021.

[5] J. Gardner, M. Kusner, Zhixiang, K. Weinberger, and J. Cunningham, "Bayesian optimization with inequality constraints," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. P. Xing and T. Jebara, Eds., vol. 32, no. 2. Bejing, China: PMLR, 22–24 Jun 2014, pp. 937–945. [Online]. Available: https://proceedings.mlr.press/v32/gardner14.html

[6] R. R. Duivenvoorden, F. Berkenkamp, N. Carion, A. Krause, and A. P. Schoellig, "Constrained bayesian optimization with particle swarms for safe adaptive controller tuning," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 11 800–11 807, 2017, 20th IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896317326290

[7] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson, "Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration," in *Advances in Neural Information Processing Systems*, 2018.

[8] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[9] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy, "BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization," in *Advances in Neural Information Processing Systems 33*, 2020. [Online]. Available: http://arxiv.org/abs/1910.06403

[10] GPy, "GPy: A gaussian process framework in python," http://github.com/SheffieldML/GPy, since 2012.

[11] J. Kirschner, M. Mutný, N. Hiller, R. Ischebeck, and A. Krause, "Adaptive and safe bayesian optimization in high dimensions via one-dimensional subspaces," in *ICML*, 2019.

[12] B. Sukhija, M. Turchetta, D. Lindner, A. Krause, S. Trimpe, and D. Baumann, "Scalable safe exploration for global optimization of dynamical systems," in *arXiv preprint arXiv:2201.09562*, January 2022.

[13] A. Krause and C. Ong, "Contextual gaussian process bandit optimization," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., vol. 24. Curran Associates, Inc., 2011. [Online]. Available: https://proceedings.neurips.cc/paper/2011/file/f3f1b7fc5a8779a9e618e1f23a7b7860-Paper.pdf

ETH zürich

Institute for Dynamic Systems and Control

Prof. Dr. R. D'Andrea, Prof. Dr. E. Frazzoli, Prof. Dr. Lino Guzzella, Prof. Dr. C. Onder, Prof. Dr. M. Zeilinger

Title of work:

Constrained Bayesian Optimization for MPC tuning in autonomous racing

Thesis type and date:

Semester Project, October 2022

Supervision:

Manish Prajapat
Prof. Dr. Melanie Zeilinger

Student:

Name: Albert Gassol Puigjaner
E-mail: agassol@student.ethz.ch
Legi-Nr.: 21-947-445
Semester: HS 2022

Statement regarding plagiarism:

By signing this statement, I affirm that I have read and signed the Declaration of Originality, independently produced this paper, and adhered to the general practice of source citation in this subject-area.

Declaration of Originality:

`https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/declaration-originality.pdf`

Zurich, 3.3.2023: