Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Learning-based Model Predictive Control for Automotive Applications

Ph.D. Candidate
Enrico Picotti

Advisor
Prof. Mattia Bruschetta

Director & Coordinator
Prof. Fabio Vandin

Ph.D. School in
Information Engineering

Department of
Information Engineering
University of Padova
2023

University of Padova
Department of Information Engineering

UNIVERSITÀ DEGLI STUDI DI PADOVA

Ph.D. course in: Information Engineering
Curriculum: Information Science and Technology

Learning-based Model Predictive Control for Automotive Applications

Director: Prof. Fabio Vandin
Advisor: Prof. Mattia Bruschetta

Ph.D. candidate: Enrico Picotti
Cycle: 36th (2020-2023)

Year: 2023

Abstract

In recent decades, the automotive industry has largely adopted advanced controllers, such as advanced driver assistance systems (ADASs), traction control systems, and anti-lock braking systems. Moreover, the significance of virtual prototyping tools has grown substantially in vehicle and controller design. These applications often depend on a controller responsible for generating vehicle input signals, e.g., accelerator, brake, and steering wheel angle. Within this context, Nonlinear Model Predictive Control (NMPC) has been widely employed due to its systematic approach in handling constrained multi-input multi-output systems. However, implementing NMPC strategies involves the necessity for a vehicle model, including complex subsystems like tires and suspensions. Hence, the model must be tailored to suit real-time applications while ensuring enough precision for achieving high-performance control. Furthermore, the versatility of NMPC controllers introduces a large number of parameters that require careful tuning, posing a significant challenge in NMPC design.

To deal with these issues, Learning-based Nonlinear Model Predictive Control (LbNMPC) could be exploited. It is an innovative control strategy that integrates NMPC techniques with data-driven methods, allowing to leverage the advantages of both the methodologies. In its framework, different categories have been explored. On one side, the learning dynamics branch aims at obtaining or refining the dynamics model by using data-driven techniques. On the other side, the learning design branch exploits learning-based strategies to maximize the closed-loop performance of the underlying NMPC controller, by algorithmically tuning the NMPC parameters.

In this thesis, implementations of LbNMPC methods for two-wheels and four-wheels vehicles are presented. On the learning dynamics branch, the development of grey-box and black-box dynamics models for both two-wheels and four-wheels vehicles is detailed, with specific focus on meeting the real-time constraint. Gaussian Process (GP) Regression has been chosen to model the data-driven components of the dynamics, due to the possibility of limit the dimensionality while preserving the model characteristics. Different offline and online reduction techniques have been explored, e.g., feature selection procedures, sparse GP approximations, and local GP approximations, and they have been adapted to the specific case of interest. On the learning design branch, the application of a genetic algorithm to obtain a high-performance virtual rider that considers the sensitivity with respect to parameters changes is described.

The results show proficiency of the presented methodologies in both realizing data-driven model dynamics for control purposes while maintaining real-time applicability, and in obtaining an appropriate tuning of the NMPC controller for a virtual motorcycle. Within the former strategies, the development of grey-box models resulted very effective in improving the tracking performance of an NMPC controller for a virtual motorcycle, and in enhancing the efficacy of a nonlinear model predictive contouring controller for lap-time minimization for a virtual high-performance car. Moreover, the implementation of a LbNMPC based on a black-box model for a real go-kart shows the applicability of the strategy using purely data-driven models in a challenging scenario. Finally, the data-driven tuning of the virtual rider allowed to establish the trade-off between performance and robustness of the controller.

Ringraziamenti

Il percorso di ricerca che mi ha portato alla scrittura di questa tesi è stato per me di grande stimolo sia scientifico/professionale che personale, insegnandomi ad esplorare nuovi argomenti da diversi punti di vista ed approfondire con passione le tematiche incontrate. Credo quindi che questa esperienza mi permetterà di affrontare con maggiore sicurezza le sfide che incontrerò, ed a guardare con consapevolezza la strada che mi aspetta. Ho molte persone da ringraziare per avermi accompagnato e sostenuto in questo periodo, e dedico loro questo traguardo. In particolare, vorrei menzionare alcune di queste persone con cui ho condiviso momenti importanti.

Prof. Mattia Bruschetta, che mi ha permesso di accedere a questo percorso ed è stato mio mentore durante questi anni, con cui ho potuto discutere appassionatamente e con soddisfazione non solo di risultati scientifici, ma anche di politica, società ed economia. Prof. Alessandro Beghi, che mi ha accolto nel suo gruppo di ricerca e mi ha sostenuto in tante occasioni, e Prof. Ruggero Carli, per la sua disponibilità e la fiducia che mi ha concesso. I colleghi di Dipartimento, Matthias e Nicola, per la passione e i progetti condivisi dai tempi della magistrale, Enrico, per avermi introdotto a questo percorso e accompagnato nel mio periodo all'estero, Giulia, per la sua entusiastica presenza fin dai primi anni, Alberto, per le illuminanti conversazioni sia personali che scientifiche, Tommaso, per gli scambi di idee rivoluzionarie (ma non troppo), Luca, per le lunghe chiacchiere durante le notti in missione o girando intorno agli uffici, e tutti gli Automatici, con i quali ho condiviso tanto e che hanno reso divertenti anche i momenti più difficili.

I tanti coinquilini di Padova ai tempi dell'Università, Marco, Tommaso, Michele, Margherita, Clara, Lorenzo, Giacomo, Noemi, Mattia, Fiammetta, e tutti coloro che sono passati per Via San Francesco 170. Grazie a voi quegli anni sono stati un periodo indimenticabile, che rimarrà negli annali. Insieme a voi (e forse nonostante) ho perseguito quell'agognata laurea che mi ha permesso di proseguire fino a qui.

Tutti gli amici, che, anche senza nominarli personalmente, sanno di avermi supportato in questi anni, distraendomi dalle preoccupazioni e appoggiando i miei interessi personali, dai viaggi in moto a quelli in posti esotici, dalle gite in montagna ai cammini zaino in spalla, dalle cene tranquille ai festoni nei campi.

La mia famiglia, i miei genitori e mio fratello, coloro che più di tutti mi hanno accompagnato lungo questo percorso. Le nonne Enrichetta e Giovanna, che hanno raccomandato a noi nipoti con tanta intensità l'importanza dello studio. Mia mamma Giusi, per il suo sostegno sempre presente ed i consigli discreti che ha saputo darmi in ogni occasione. Mio papà Michele, per avermi insegnato fin da piccolo il pensiero scientifico e aver risposto alle mie infinite curiosità sul funzionamento degli oggetti quotidiani. Mio fratello Giovanni, che è sempre disponibile quando c'è bisogno di lui e da bravo fratello maggiore mi ha dato il buon esempio.

Infine, la mia compagna Margherita, che in questi anni mi è stata vicina in tutte le mie scelte, sapendo comprendere le mie necessità e spronandomi a seguire i miei desideri. Sempre al mio fianco, nei periodi all'estero, nei viaggi di lavoro, nelle giornate difficili o di sconforto, aiutandomi a ritrovare la serenità che mi auspico manterremo per gli anni a venire.

Contents

1 Introduction

Contents

1.1 State of the Art

In recent decades, the use of advanced controllers has been spreading in all types of vehicles, e.g. advanced driver assistance systems (ADASs), traction control systems, anti-lock braking systems, etc. [1]. A large research effort has been dedicated to the design of ADASs with autonomy-like characteristics, aimed at assisting the driver in increasing both safety in emergency situations and comfort in highway and urban scenarios. At the same time, virtual prototyping tools acquired great importance in designing new vehicles and controllers. Indeed, offline tests in simulative scenarios allow to highly reduce both cost and time-to-market of newly developed products, making them established tools in the automotive industry. These applications often rely on a vehicle controller which generates input signals for the actual vehicle commands (e.g., accelerator, brake, steering wheel angle), in order to follow a given reference or to compute the optimal velocity and trajectory profiles. In this context, MPC has become widely used due to its ability to methodically handle constrained multi-input multi-output systems [2]. Indeed, MPC allows to exploit the dynamics model of a vehicle, to consider the maximum performance in terms of accelerations, and to include the physical constraints for inputs and states. In particular, Non-linear MPC (NMPC), i.e., MPC that exploits a non-linear model of the plant, is a well established methodology in the automotive framework, and promising results have been obtained in (semi-)autonomous driving tasks, for both two-wheel and four-wheel vehicles, e.g. in [3–6].

For four-wheel vehicles, different NMPC solutions have been proposed. The real-time trajectory planning problem and the tracking of the corresponding planned path has been addressed in [3, 4] applying NMPC. In [3], the problem of real-time obstacle avoidance on low-friction road surfaces is addressed, while in [4] a tailored NMPC strategy is proposed, aiming at tracking the lane center line while avoiding collisions with obstacles. NMPC has also been effectively applied to autonomous 1:43 scale RC electric vehicles, aiming at minimizing the lap time, either by maximizing the progress on track over the prediction horizon of the NMPC [7], or by addressing a minimum traveling time objective [8], and a NMPC controller for an autonomous Formula Student racing car is presented in [9].

For two-wheel vehicles, different implementations of path tracking (N)MPC controllers have been studied. In [10], a model consisting in an inverted pendulum mounted on a non-holonomic cart is used to track given path and velocity profiles. The roll angle is assumed to be a virtual input and the steering and longitudinal controls are obtained by inverting the dynamics. An MPC controller is designed in [11], where the model is linearized by assuming time-invariant dynamic response over the prediction horizon. An extensive parameter study on the predictive steering control is reported by the same authors in [12]. An NMPC strategy based on an internal model linearized at the steady state condition of motorcycle motion (i.e. constant speed and constant radius turn) is adopted in [5]. NMPC has been aslo used in [6], where the internal model is linearized at the quasi steady state condition taking into account longitudinal acceleration and roll derivatives. In [13], the usage of NMPC for controlling the rider-bicycle system by roll-angle tracking is investigated, while using a linear model for the control design.

However, the application of the NMPC strategy is characterized by the need for a complex nonlinear model, comprehensive of involved subsystems, e.g., tires, suspensions, etc. Indeed, modest inaccuracies can highly affect performance, e.g. lead to undesired under- or over-steer behavior, hence the model must be able to predict the dynamics very accurately. At the same time, the evaluation of the model and the calculation of its sensitivities account for the major part of the computational complexity of the NMPC, and thus its definition must be explicitly tailored for a real-time application. Therefore, obtain the most suitable characterization of the system is critical for high-performance control. Moreover, the versatility of NMPC controllers imply a high number of parameters that must be properly tuned, making the tuning procedure an important challenge in NMPC design, as it is often accomplished by the control engineer through a trial-and-error procedure [14]. In particular, a critical tuning parameter of the controller is the set of weights used in the objective function, and finding the best configuration is essential to obtain a proficient controller.

To cope with the modeling and tuning issues, the usage of data-driven techniques to enhance MPC strategies is an innovative solution, namely Learning-based MPC (LbMPC) [15], that allows to improve the control capability of the closed-loop system while preserving the MPC advantages. The arising problem from the combination of these strategies can be formulated as a Stochastic Optimal Control Problem (SOCP), where the minimization function is an expected value, the differential equation constraint (i.e. the dynamics of the system) is subject to uncertainties and the system constraints must be satisfied in probability. However, the direct solution of SOCP is computationally hardly possible. In this framework, the MPC scheme can be then used to approximate the SOCP iteratively, solving a simplified version of the problem initialized at the current state on a shorter prediction horizon. The LbMPC framework, also referred to as Learning-based Nonlinear Model Predictive Control (LbNMPC) when applied to nonlinear systems, can then be divided into three main categories: (i) *Learning Dynamics*, a data-driven improvement of the underlying dynamics model for enhancing the prediction accuracy, (ii) *Learning Design*, a data-driven optimization of the controller parametrization for achieving maximum closed-loop performance, and (iii) *MPC for safe learning*, where MPC is used to obtain safety guarantees in learning-based controllers (as depicted in Fig. 1.1). Throughout the thesis, the first two categories will be considered.

The learning dynamics technique aims specifically at enhancing or deriving the prediction model of the (N)MPC through the exploitation of system data acquired interacting with the system. Two categories can be defined based on the model uncertainty representation, i.e., if the uncertainty is assumed to belong to a compact set, they fall in the robust models category, e.g., as in [16, 17], while if the uncertainty is represented trough

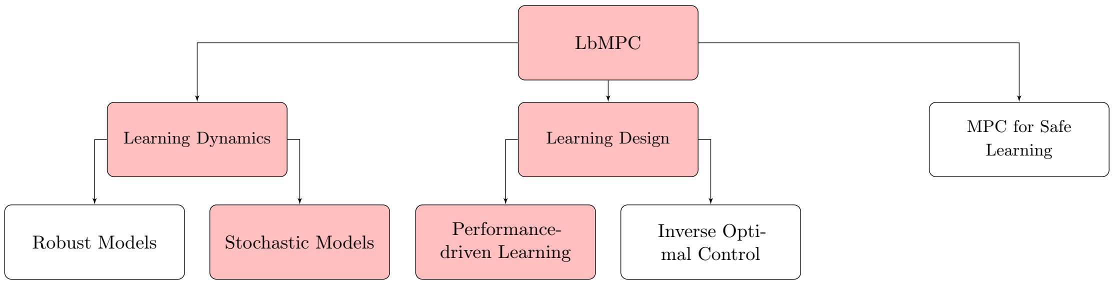


Figure 1.1. State of the art representation of the Learning-based MPC framework [15].

distributional information, they fall in the stochastic models one, e.g., as in [18–20]. Both robust and stochastic models can be characterized as *grey-box* or *black-box*. The former relies on a mathematical description of the system known *a priori*, which is complemented by a learning-based modeling of the model mismatch, defined by exploitation of the data. Different formulations have been proposed, e.g., in [21] an arbitrary continuous and bounded *oracle* function is used to obtain the learning-based part of the model, Gaussian Processes (GP) have been used in [22], and in [23] a parametric model derived by combining physical principles, causal relations and experimental data has been considered. The black-box approach, instead, completely bases the modeling on the data, and different strategies have been studied, e.g., in [24] the model derivation is obtained by nonlinear set membership methodology, in [25] the use of neural networks has been proposed, in [26] input-output modeling trough kinky inference has been developed, in [27] a Takagi-Sugeno adaptive neuro-fuzzy inference system has been considered, while in [28] non-parametric kernel regression has been adopted. In the learning dynamics context, Gaussian Process Regression (GPR) [29] has demonstrated remarkably effective thanks to offline and runtime reduction techniques that maintain the GP characteristics, e.g. continuity and smoothness of the function, favoring its usage in strictly real-time scenarios. Moreover, GPR exploits a solid probabilistic framework that favors the inclusion of a priori knowledge to define input-output models or to compensate modeling inaccuracies by employing data of the real system, and its implementation in Learning-based Model Predictive Control has been studied in different applicative scenarios, e.g., automotive, robotics [30–34].

The learning design branch addresses the optimization of a *true* closed-loop cost for the whole task, as a mapping from a parametrized form of the MPC cost function, model or constraints to high-level control objectives. It can be divided in two subcategories. On one side, the inverse optimal control strategy deals with the automatic design of a controller that imitate a chosen desired system behaviour. On the other side, performance-driven learning adapts the parametric problem to enhance controller performance, stability and/or robustness in an episodic setting. Within the latter framework, Bayesian Optimization has been used to tune the model and controller in iterative tasks [35], and the demonstration that MPC can be tuned to obtain the optimal policy even with a wrong model description led to its adoption as a function approximator in the Reinforcement Learning scheme [36]. Moreover, different MPC auto-tuning strategies, i.e. which specifically considers the cost function weights as tuning parameters, have been implemented based on data-driven techniques [37], e.g. Reinforcement Learning [38], Bayesian Optimization [39], Particle Swarm Optimization [40] and Genetic Algorithm [41]. Due to the possibility of solving multi-objective optimization problems with constraints, Genetic Algorithms (GAs) result to be a common approach for tuning complex NMPC controllers [41–44]. Using GA, in fact, it is possible to define an optimal tuning problem that considers and balances different aspects of the desired control performance.

Recently, the application of LbNMPC strategies to four-wheel vehicles control is gaining academic and industrial interest, and it has been proposed for different tasks, e.g., powertrain and vehicle dynamics control. A comprehensive summary of the Learning-based NMPC publications in the automotive sector can be found in [45]. Specifically, vehicle dynamics control comprehends driving assistance, stability control and autonomous driving. In this context, learning dynamics approaches have been proposed to obtain the model, exploiting different data-driven representations, e.g. Set Membership [24], Fuzzy Modeling [27] and GPs [31–34]. In particular, GP-based implementations resulted very effective in enhancing the control performance and they have been recently applied in autonomous driving. A driverless race car controller has been developed in [31], where a dynamic bicycle model is improved using a sparse GP formulation obtained by Fully Independent Training Conditional to reduce the GP dimension, the path following problem for off-road mobile robots has been studied in [32], where a kinematic unicycle model is enhanced by a learned disturbance model representing both unmodelled robot dynamics and systematic environmental disturbances, a control design for aggressive vehicle maneuvers is described in [33], where Gaussian processeses on polynominal basis are introduced to improve the accuracy of vehicle and tire dynamics, and an high performance control method for remote-controlled race cars is presented in [34], where a bicycle model considering Pacejka tire forces is enriched by GP-based velocity prediction error estimates.

As shown in the literature, to obtain effective real-time LbNMPC controllers different methodologies must be considered, e.g., feature selection procedures to reduce the GP dimension, sparse GP formulations to further reduce the model complexity, fast NMPC approaches to accelerate the optimal control problem solution. In this sense, the most suitable GP characterization, its approximation and their correct integration within the NMPC controller is still under investigation. Notably, for two-wheel vehicles the integration of learning-based strategies and NMPC is still an open research topic, and the promising results achieved for four-wheel vehicles suggest interesting possibilities in further developments for autonomous riders.

1.2 Thesis Contribution

The main contribution of the thesis consists in the application of LbNMPC methodologies in automotive scenarios. The doctoral scholarship has been funded by an agreement of the Department of Information Engineering of University of Padova with the company VI-Grade GmbH, with the specific topic "*Desing of control systems for virtual vehicles and driving simulators*", hence most results regard virtual drivers and riders, i.e., controllers for virtual vehicles, while a concluding application on a real go-kart platform has been implemented. In theses cases, the task is to obtain real-time capable controllers by balancing prediction accuracy, horizon length, and computational cost. Both two-wheel and four-wheel vehicles have been considered, focusing mainly on learning dynamics strategies to enhance or derive the control model, and a data-driven tuning procedure for a virtual rider has been developed.

1.2.1 Two-wheel vehicles

Virtual riders design based on LbNMPC methodology is an innovative research topic, and it has been addressed in this thesis both in the context of learning dynamics and learning design.

Firstly, a tailored LbNMPC that relies on a continuous grey-box model based on GPs has been implemented. The formulation leverages the data acquired in specific training runs to obtain a more accurate characterization of the accelerations. To cope with the computational burden induced by the GP, the following features have been explored: (i) the dimension of the GP input is reduced through a systematic feature selection strategy; (ii) an off-line sparse GP approximation based on Variational Free Energy approach is exploited; (iii) an online sparse GP approximation based on Transduction Learning technique is applied; (iv) Non-Uniform Grid integration has been employed taking advantage of the continuous-time formulation, i.e., modeling the acceleration mismatch rather then the velocity prediction one. The results show the proficiency of the proposed approach in remarkably enhancing the tracking performances of the controller while maintaining real-time capabilities.

Secondly, a data-driven tuning procedure based on a Genetic Algorithm for an NMPC-based virtual rider has been developed. The considered optimization objectives of the GA are both the high-level performance of the control task, i.e. the lap-time on the track, and the sensitivity of the solution to variations of controller parameters, as an indicator of robustness with respect to the configuration changes. In particular, to enforce robustness an additional criterion has been added by casting the single-objective optimization problem into a multi-objective one. Moreover, the possibility to fail the task, due to the motorcycle falling down or numerical issues related to NMPC fast/approximate solutions, is explicitly considered in the problem formulation. The proposed method results suitable for tuning a nonlinear MPC-based virtual motorcycle rider, obtaining better results with respect to previously published manual tuning, and for evaluating the trade-off between performance and robustness of the controller.

1.2.2 Four-wheel vehicles

LbNMPC methodologies for four-wheel vehicle control have been recently studied, and they have been addressed in this thesis both in the context of virtual models and experimental scenarios.

Firstly, a LbNMPC for a high-performance virtual driver running real-time has been developed. The target is to enhance performance through the development of a discrete grey-box model, improving physically-derived dynamics by training GPs to compensate for velocity prediction errors. In particular, the definition of the most appropriate GP has been explicitly addressed by: (i) implementing a specific feature selection procedure, (ii) considering different stationary kernel functions, and (iii) analyzing the trade-off between sparse reduction and prediction accuracy. Moreover, the task is formulated in a space-domain rather than time-domain, and it has been provided a specific data acquisition method to deal with this case. The obtained behavior clearly outperforms the purely physical models, both in open-loop predictions and in closed-loop performance, still maintaining real-time feasibility.

Secondly, a LbNMPC controller for a real go-kart based on a pure black-box model obtained by GPs has been designed and tested. The experimental environment is a very challenging scenario for the proposed methodology, due to low computational power on embedded computers, sensor noise, communication flaws, etc. In this sense, the implementation and validation of an LbNMPC controller on a physical system is the main contribution of the thesis, demonstrating the effectiveness of the approach in real contexts. Specifically, by conveniently adopting a continuous model of accelerations, it has been possible to directly use a space-based contouring problem formulation, computing the optimal trajectory and velocity profiles directly within the online optimization. To limit the computational burden, an offline GP model reduction strategy based on Subset of Data is implemented, and an online Nearest Neighbours method relying on the state prediction of the LbNMPC is used, leading to a sequence of low-dimensional local GP models. Validation of the approach has been accomplished by autonomously driving a real go-kart vehicle on a 180m long indoor track. The proposed method allowed for accomplishing the driving task, properly performing the lap while keeping the go-kart within the track bounds, demonstrating that the black-box dynamic model can be a viable approach even in such a complex scenario.

Publications

Through the thesis, some of the author's recently published and submitted papers are presented. In particular, Part I contains the continuous learning dynamics characterization published in [46]; Part II includes the NMPC controller for virtual motorcycles published in [47], the data-driven tuning of the aforementioned NMPC controller presented in [48] and its grey-box modeling LbNMPC implementation submitted as [49]; Part III contains the NMPC controller for a virtual car published in [50], its grey-box modeling LbNMPC implementation presented in [51] and the black-box modelling LbNMPC for a real go-kart published in [52]. Finally, the Appendix contains the toolbox `LbMATMPC` presented in [53].

1.3 Outline of the Thesis

The thesis is organized as follows. Part I comprehends the methodological framework, describing in Ch. 2 the Learning-based Nonlinear Model Predictive Control scheme, and in Ch. 3 the Gaussian Process Regression methodology with specific reference to the relevant features in the applicative case of interest. Part II concerns the virtual motorcycles control, and it includes in Ch. 4 the NMPC controller underlying the following works, in Ch. 5 the data-driven tuning procedure implemented to optimize the controller behaviour, and in Ch. 6 the LbNMPC controller based on a grey-box dynamics model. Part III deals with four-wheel vehicles control, describing in Ch. 7 the NMPC controller underlying the following learning-based implementation, in Ch. 8 the LbNMPC controller based on a grey-box dynamics model developed in a virtual environment, and in Ch. 9 the LbNMPC implementation for an experimental go-kart platform based on a black-box dynamics model. Finally, the developed MATLAB toolbox that allows to integrate data-driven components in the NMPC prediction model, namely `LbMATMPC`, is presented in Appendix, together with a comparison between the continuous and discrete formulations for dynamics learning.

Part I

Background: Methodological Framework

In this part, the background on the methodological framework and the most relevant algorithms applied through the thesis are presented. Chapter 2 describes the Learning-based Nonlinear Model Predictive Control scheme, detailing the Stochastic Optimal Control Problem formulation and its approximation and solution through Nonlinear Model Predictive Control technique. Moreover, it illustrates the Learning Dynamics and Learning Design techniques, together with the related algorithms. In chapter 3, the Gaussian Process Regression is outlined, specifying the relevant reduction techniques used for real-time feasibility of the LbNMPC problem solution.

2 Learning-based Nonlinear Model Predictive Control

The Learning-based Nonlinear Model Predictive Control strategy deals with the control of nonlinear dynamical systems in presence of uncertainty, where the task is to create an optimal controller for the unknown/disturbed system. In this sense, it can be interpreted as an ideal Stochastic Optimal Control Problem (SOCP), that is approximated by a specifically designed Nonlinear Model Predictive Control problem. Different variants can be derived, relying on data-driven techniques either to constitute or enhance the prediction model within the NMPC, i.e. learning dynamics branch, or to optimize the closed-loop behaviour, i.e. learning design one.

2.1 Stochastic Optimal Control Problem

In general, the system uncertainty can be represented both as process noise $\boldsymbol{w}$, i.e. a sequence of random variables assumed to be independent and identically distributed, or a parametric uncertainty $\boldsymbol{\theta} \sim \mathcal{Q}$, i.e. a random variable constant over time, the performance is related to a cost function J and the controller should fulfill the constraints on states and inputs with a given probability, namely, *chance* constraints. Hence, a generic SOCP in discrete time domain may be defined similarly to [15] as

$$\begin{aligned}
\min_{\{\pi_k\}} \quad & J = \mathbb{E}\left(\sum_{k=0}^{\infty} l(\boldsymbol{\xi}(k), \boldsymbol{u}(k), k)\right) && \text{(2.1a)}\\
s.t. \quad & \boldsymbol{\xi}(k+1) = \boldsymbol{\phi}(\boldsymbol{\xi}(k), \boldsymbol{u}(k), \boldsymbol{w}(k), \boldsymbol{\theta}, k), && \text{(2.1b)}\\
& \boldsymbol{u}(k) = \pi_k(\boldsymbol{\xi}(0), \dots, \boldsymbol{\xi}](k)), && \text{(2.1c)}\\
& Pr(\Xi \in \bar{\Xi}) \geq p^{\xi} && \text{(2.1d)}\\
& Pr(\Upsilon \in \bar{\Upsilon}) \geq p^{u} && \text{(2.1e)}
\end{aligned}$$

where the minimization is taken with respect to the control policy at k-th time instant π_k, the cost is defined as the expected value with respect to the random variables of the sum of stage costs in the future, the system evolution is characterized by $\boldsymbol{\phi}$, the control inputs are defined by the control policy π_k, and the states $\Xi = [\xi_0, \xi_1, ...]$ and inputs $\Upsilon = [u_0, u_1, ...]$ must lie within their constraint sets $\bar{\Xi}$ and $\bar{\Upsilon}$ with probability $p^{\xi,u}$, respectively. In particular, the system dynamics in Eq. (2.1b), i.e.

$$\boldsymbol{\xi}(k+1) = \boldsymbol{\phi}(\boldsymbol{\xi}(k), \boldsymbol{u}(k), \boldsymbol{w}(k), \boldsymbol{\theta}, k),$$

where $\boldsymbol{\xi} \in n_\xi$ is the state, $\boldsymbol{u} \in n_u$ is the input, $\boldsymbol{w} \in n_w$ is the process noise, and $\boldsymbol{\theta} \in n_\theta$ is an (uncertain) parameter vector, represents the (partially unknown) discrete-time evolution of the system.

The obtained minimization problem can hardly be solved, and becomes intractable for nonlinear system. In fact, a direct optimization over general feedback laws π_k is not feasible, even since the cost function may be non-convex. In this sense, Nonlinear Model Predictive Control can be used to approximate the SOCP iteratively at runtime, to obtain a relatively optimal policy for the system. Specifically, the LbNMPC framework aims at defining a deterministic NMPC problem that achieves the desired performance, either through the exploitation of a probabilistic model of the system or by a data-driven adaptation of the problem parameters (e.g. cost weights, prediction horizon length, etc.).

2.2 Nonlinear Model Predictive Control

Model Predictive Control is a an optimization-based control technique that leverages explicitly system models and constraints. It computes a control sequence by minimizing a cost function in the prediction horizon $[t_k, t_{k+T}]$, while satisfying the dynamics of the system. Then, only the first control action of the sequence is applied to the plant, the current state measurements (or estimates) are obtained and the procedure is repeated, as depicted in Fig. 2.1, [54, 55].

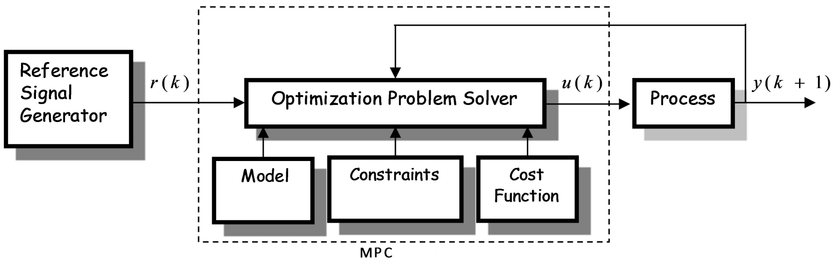


Figure 2.1. MPC scheme [55].

When leveraging nonlinear dynamics, the technique is referred to as Nonlinear MPC (NMPC). A precise characterization of the system can highly enhance the control performance, but, at the same time, the nonlinearities could yield to difficulties in solving the optimization problem accurately and in real-time. However, fast solution algorithms and high computational power even in embedded control units led to the adoption of NMPC in different industrial scenarios [56, 57].

2.2.1 Optimal Control Problem

The NMPC scheme requires, at each time step, to solve an Optimal Control Problem (OCP) that, given the current state measurement $\check{\boldsymbol{\xi}}_k$, allows to find the optimal control trajectory in the prediction horizon $[t_k, t_{k+T}]$. The OCP can be stated as

$$\min_{\boldsymbol{\xi}(\cdot),\boldsymbol{u}(\cdot)} \quad J = \int_{t_k}^{t_{k+T}} (l(\boldsymbol{\xi}(k), \boldsymbol{u}(k), k)dt) + L(\boldsymbol{\xi}(t_{k+T})) \tag{2.2a}$$

$$s.t. \quad \boldsymbol{\xi}(t_k) = \check{\boldsymbol{\xi}}_k \tag{2.2b}$$

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}(t), \boldsymbol{u}(t)),\ \forall t \in [t_k, t_{k+T}], \tag{2.2c}$$

$$\boldsymbol{r}(\boldsymbol{\xi}(t), \boldsymbol{u}(t)) \leq \boldsymbol{0},\ \forall t \in [t_k, t_{k+T}] \tag{2.2d}$$

$$\boldsymbol{r}_T(\boldsymbol{\xi}(t_{k+T})) \leq \boldsymbol{0}, \tag{2.2e}$$

where T is the prediction horizon length, $l(\cdot)$ represent the control objective, $L(\cdot)$ is the terminal cost, $\boldsymbol{r}(\cdot)$ defines the states and input constraints, and $\boldsymbol{r}_T(\cdot)$ is the boundary condition, i.e. the terminal constraint. The definition of those quantities is particularly critical, as the prediction horizon length, the terminal costs and terminal constraints can be used to guarantee *stability* of the system under NMPC control law, i.e. ensuring that, for a constant reference, the system will steer to a stable equilibrium point. Moreover, the existence of a solution depends on the constraints definition, hence their characterization should be well-posed. Specifically, *feasibility* of the OCP means that it exist a solution trajectory, in terms of $(\boldsymbol{\xi}, \boldsymbol{u})$, for which all the constraints are satisfied. Eq. (2.2c), i.e.

$$\dot{\boldsymbol{\xi}}(t) = \boldsymbol{f}(\boldsymbol{\xi}(t), \boldsymbol{u}(t)),$$

where $\boldsymbol{\xi} \in \mathbb{R}^{n_\xi}$ is the state and $\boldsymbol{u} \in \mathbb{R}^{n_u}$ is the input, is a system of Ordinary Differential Equations (ODE) and describes the behaviour of the plant under the control input. Given an initial condition $\boldsymbol{\xi}(t_k) = \check{\boldsymbol{\xi}}_k$ and a control input $\boldsymbol{u}(t)$, it defines an Initial Value Problem. Throughout the thesis, the assumption that $\boldsymbol{\phi}$ is uniformly Lipschitz continuous in $\boldsymbol{\xi}$ and $\boldsymbol{u}$ and continuous in t will be assumed, to apply the Picard's existence theorem that assures there is a unique solution $\boldsymbol{\xi}(t)$ on a given interval $t \in [t_k, t_{k+T}]$.

In LbNMPC framework, the OCP formulation can represent an approximation of the SOCP stated in Eq. (2.1) by embedding a deterministic approximation of both the probabilistic dynamics in the ODE by, e.g., considering the mean of the random variables or perturbations, and of the chance constraint in the states and input constraints function by, e.g., considering the variance or the domain of the random variables or perturbations. In this context, *robustness* of the NMPC scheme is a desiderable property, as it determines the ability of the control algorithm to maintain certain characteristics, e.g. stability, feasibility, and/or performance, in the presence of unmodelled or uncertain dynamics.

2.2.2 Nonlinear Programming Problem

Different methods are available to solve the OCP in Eq. (2.2): (i) dynamic programming, (ii) indirect methods and (iii) direct methods [58, 59]. The third one is the most used in fast NMPC algorithms, thanks to the possibility to exploit numerical optimization solvers. Direct methods rely on a finite dimensional parametrization of the OCP, obtaining a discrete Nonlinear Programming Problem (NLP) that approximates the OCP solution [60]. In the thesis setup, multiple shooting method is used, due to its efficiency in practical applications. Indeed, the possibility to incorporate information about the behaviour of the state trajectory into the initial guess for the iterative solution procedure can damp the influence of poor initial guesses for the controls (which are usually much less known). In the context of NMPC, where a sequence of neighbouring optimization problems is treated, solution information of the previous problem can be effectively exploited [61].

In the multiple shooting method, the prediction horizon is divided in N *shooting intervals* $[t_{0|k}, t_{1|k}, \dots, t_{N|k}]$, the input trajectory is parametrized as piece-wise constant, the system dynamics is integrated to obtain a discrete state evolution and the constraints are imposed *only* at the shooting points. Specifically, at every time instant k, the OCP is converted in a NLP over the prediction horizon as follows:

$$\min_{\boldsymbol{\xi},\boldsymbol{u}} \sum_{j=0}^{N-1} \frac{1}{2} \left\| \boldsymbol{h}_j(\boldsymbol{\xi}_{j|k}, \boldsymbol{u}_{j|k}) \right\|_W^2 + \frac{1}{2} \left\| \boldsymbol{h}_N(\boldsymbol{\xi}_{N|k}) \right\|_{W_N}^2 \tag{2.3a}$$

$$s.t.\, 0 = \boldsymbol{\xi}_{0|k} - \check{\boldsymbol{\xi}}_{0|k}, \tag{2.3b}$$

$$0 = \boldsymbol{\xi}_{j+1|k} - \boldsymbol{\phi}_k(\boldsymbol{\xi}_{j|k}, \boldsymbol{u}_{j|k}),\ j = 0, 1, \dots, N-1, \tag{2.3c}$$

$$\underline{\boldsymbol{r}}_{j|k} \le \boldsymbol{r}(\boldsymbol{\xi}_{j|k}, \boldsymbol{u}_{j|k}) \le \overline{\boldsymbol{r}}_{j|k},\ j = 0, 1, \dots, N-1, \tag{2.3d}$$

$$\underline{\boldsymbol{r}}_{N|k} \le \boldsymbol{r}_N(\boldsymbol{\xi}_{N|k}) \le \overline{\boldsymbol{r}}_{N|k}, \tag{2.3e}$$

$$\tag{2.3f}$$

where $\check{\boldsymbol{\xi}}_{0|k}$ is the state at the current time instant k. At the discrete time point $t_{j|k}$ for $j = 0, \dots, N$ the system states $\boldsymbol{\xi}_{j|k} \in \mathbb{R}^{n_x}$ are defined while the control inputs $\boldsymbol{u}_{j|k} \in \mathbb{R}^{n_u}$ for $j = 0, \dots, N-1$ are piece-wise constant. Eq. (2.3a) refers to the objective function, where $\boldsymbol{h}$ is the inner objective and W is the weight matrix, Eq. (2.3b) refers to the initial value embedding, and the constraint functions (2.3d)-(2.3e) are defined as $\boldsymbol{r}(\boldsymbol{\xi}_{j|k}, \boldsymbol{u}_{j|k}) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_r}$ and $\boldsymbol{r}_N(\boldsymbol{\xi}_{N|k}) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_{r_N}}$ with upper and lower bound $\overline{\boldsymbol{r}}_{j|k}, \underline{\boldsymbol{r}}_{j|k}$. The system dynamics, Eq. (2.2c), is enforced by the *continuity constraint*, Eq. (2.3c), where $\boldsymbol{\phi}_k(\boldsymbol{\xi}_k, \boldsymbol{u}_k)$ is a numerical integration operator that solves the following IVP and returns the solution at t_{k+1}, i.e.

$$0 = \boldsymbol{\phi}(\dot{\boldsymbol{\xi}}(t), \boldsymbol{\xi}(t), \boldsymbol{u}(t), t), \quad \boldsymbol{\xi}(0) = \boldsymbol{\xi}_k. \tag{2.4}$$

2.2.3 Sequential Quadratic Programming

To solve the NLP problem, Sequential Quadratic Programming (SQP) is used in the presented works. It is an iterative procedure which reformulates the NLP at a given iterate in a Quadratic Programming (QP) problem [62, 63]. Specifically, the objective functions (2.3a) are replaced by their local quadratic approximation and the constraint functions (2.3b)-(2.3e) by their local affine approximations. Thus, at SQP iterate l, a QP problem is formulated as follows (the subscript $_{\cdot|k}$ is omitted for clarity):

$$\min_{\Delta\xi,\Delta\mathbf{u}} \quad \sum_{j=0}^{N-1}\left(\frac{1}{2}\begin{bmatrix}\Delta\boldsymbol{\xi}_j\\ \Delta\boldsymbol{u}_j\end{bmatrix}^\top H_j^l \begin{bmatrix}\Delta\boldsymbol{\xi}_j\\ \Delta\boldsymbol{u}_j\end{bmatrix} + g_j^{i\top}\begin{bmatrix}\Delta\boldsymbol{\xi}_j\\ \Delta\boldsymbol{u}_j\end{bmatrix}\right) \tag{2.5a}$$

$$+\frac{1}{2}\Delta\boldsymbol{\xi}_N^\top H_N^l \Delta\boldsymbol{\xi}_N + g_N^{l\top}\Delta\boldsymbol{\xi}_N \tag{2.5b}$$

$$s.t. \quad \Delta\boldsymbol{\xi}_0 = \check{\boldsymbol{\xi}}_0 - \boldsymbol{\xi}_0, \tag{2.5c}$$

$$\Delta\boldsymbol{\xi}_{j+1} = A_j^l\Delta\boldsymbol{\xi}_j + B_j\Delta\boldsymbol{u}_j + \boldsymbol{a}_j^l, \tag{2.5d}$$

$$\underline{\boldsymbol{c}}_j^l \leq C_j^l\Delta\boldsymbol{\xi}_j + D_j^l\Delta\boldsymbol{u}_j \leq \overline{\boldsymbol{c}}_j^l, \tag{2.5e}$$

$$\underline{\boldsymbol{c}}_N^l \leq C_N^l\Delta\boldsymbol{\xi}_N \leq \overline{\boldsymbol{c}}_N^l, \tag{2.5f}$$

where $\Delta\xi = \boldsymbol{\Xi} - \boldsymbol{\Xi}^l$, $\Delta\mathbf{u} = \mathcal{U} - \mathcal{U}^l$, using the compact notation $\boldsymbol{\Xi} = \left[\boldsymbol{\xi}_0^\top, \boldsymbol{\xi}_1^\top, \dots, \boldsymbol{\xi}_N^\top\right]^\top$, $\mathcal{U} = \left[\boldsymbol{u}_0^\top, \boldsymbol{u}_1^\top, \dots, \boldsymbol{u}_{N-1}^\top\right]^\top$ for the discrete state and control variables, and $\boldsymbol{\Xi}^l$ and $\mathcal{U}^l$ are the previous guess for state and control trajectories. The linearized matrices are given by

$$\begin{aligned}
&A_j^l = \frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{\xi}_j}, \quad B_j^l = \frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{u}_j},\\
&\boldsymbol{a}_j^l = \boldsymbol{\phi}(\boldsymbol{\xi}_j^l, \boldsymbol{u}_j^l) - \boldsymbol{\xi}_{j+1}^l,\\
&C_j^l = \frac{\partial\boldsymbol{r}_j}{\partial\boldsymbol{\xi}_j}, \quad D_j^l = \frac{\partial\boldsymbol{r}_j}{\partial\boldsymbol{u}_j}, \quad C_N^l = \frac{\partial\boldsymbol{r}_N}{\partial\boldsymbol{\xi}_N},\\
&\overline{\boldsymbol{c}}_j^l = \overline{\boldsymbol{r}}_j - \boldsymbol{r}_j(\boldsymbol{\xi}_j^l, \boldsymbol{u}_j^l), \quad \underline{\boldsymbol{c}}_j^l = \underline{\boldsymbol{r}}_j - \boldsymbol{r}_j(\boldsymbol{\xi}_j^l, \boldsymbol{u}_j^l),\\
&\overline{\boldsymbol{c}}_N^l = \overline{\boldsymbol{r}}_N - \boldsymbol{r}_N(\boldsymbol{\xi}_N^l), \quad \underline{\boldsymbol{c}}_N^l = \underline{\boldsymbol{r}}_N - \boldsymbol{r}_N(\boldsymbol{\xi}_N^l),
\end{aligned} \tag{2.6}$$

while the Hessian matrices H_j^l, H_N^l are computed through the Gauss-Newton method.

The solution of the QP problem can be obtained by different methods [64, 65], and the most suitable ones in this scenario are *active-set* and *interior point*. Active-set techniques are based on the detection of the *active* inequality constraints, i.e. the ones that are needed to correctly obtain the QP solution [66], while interior point ones remove the inequality constraints by adding slack variables with a penalty term [67]. Both methods can be used within the proposed methodology, and the most appropriate should be chosen depending on the specific application.

Thanks to the similarity of the NLPs at two consecutive sampling instants in this framework, different methodologies have been developed to speed up the solution time of the problem. In particular, Real-Time Iteration (RTI) scheme is a strategy that performs only one SQP iteration to solve the NLP, providing a sub-optimal solution that is a tangential predictor of the exact one [68]. If the initial trajectory is close enough to the optimal one, local convergence of the algorithm is ensured by RTI guarantees on contractivity and boundness of the loss of optimality compared to optimal feedback control [69].

The complete solution of (2.3) is obtained by using solution of (2.5) through

$$\begin{aligned}
\boldsymbol{\Xi}^{l+1} &= \boldsymbol{\Xi}^l + \beta^l\Delta\xi^l,\\
\mathcal{U}^{l+1} &= \mathcal{U}^l + \beta^l\Delta\mathbf{u}^l,
\end{aligned} \tag{2.7}$$

where β^l can be computed at each step by globalization strategies to apply the optimal step size.

2.3 Learning Design

The cost function and constraints have a great impact on the closed-loop performance of a predictive controller, and their tuning through specific automatic algorithms has been recently studied. Two categories can be defined: (i) Inverse Optimal Control, where the controller is tuned in order to imitate a given closed-loop behaviour, and (ii) Performance-driven Learning, where the aim is to improve the performance by consecutive adjustments of the parametrization.

The Performance-driven Learning is a very active research field, and different auto-tuning strategies have been implemented based on data-driven techniques. In this episodic framework, a further optimization problem that minimizes a *true* closed-loop cost of the whole task $J_t(\Theta)$ is defined, considering the tuning parameters Θ as optimization variables. Hence, by consecutive runs of the task using a parametrized NMPC controller, the strategies aim at minimize high-level control objectives such as the controller performance, stability and/or robustness [37]. The high-level problem can be defined as

$$\arg\min_{\Theta} J_t(\Theta) \tag{2.8}$$

and the NLP to be solved by the NMPC controller is expressed in general form as

$$\min_{\boldsymbol{\xi},\boldsymbol{u}} \sum_{j=0}^{N-1} \frac{1}{2} \left\| \boldsymbol{h}_j(\boldsymbol{\xi}_{j|k}, \boldsymbol{u}_{j|k}; \boldsymbol{\theta}_l) \right\|_W^2 + \frac{1}{2} \left\| \boldsymbol{h}_N(\boldsymbol{\xi}_{N|k}; \boldsymbol{\theta}_l) \right\|_{W_N}^2 \tag{2.9a}$$

$$s.t.\, 0 = \boldsymbol{\xi}_{0|k} - \bar{\boldsymbol{\xi}}_{0|k}, \tag{2.9b}$$

$$0 = \boldsymbol{\xi}_{j+1|k} - \boldsymbol{\phi}(\boldsymbol{\xi}_{j|k}, \boldsymbol{u}_{j|k}; \boldsymbol{\theta}_\phi),\ j = 0, 1, \dots, N-1, \tag{2.9c}$$

$$\underline{\boldsymbol{r}}_{j|k} \leq \boldsymbol{r}(\boldsymbol{\xi}_{j|k}, \boldsymbol{u}_{j|k}; \boldsymbol{\theta}_r) \leq \overline{\boldsymbol{r}}_{j|k},\ j = 0, 1, \dots, N-1, \tag{2.9d}$$

$$\underline{\boldsymbol{r}}_{N|k} \leq \boldsymbol{r}_N(\boldsymbol{\xi}_{N|k}; \boldsymbol{\theta}_r) \leq \overline{\boldsymbol{r}}_{N|k}, \tag{2.9e}$$

$$\tag{2.9f}$$

where the vector Θ comprehends the whole parametrization, i.e., $\boldsymbol{\theta}_l \in \Theta$ are the cost function parameters, $\boldsymbol{\theta}_\phi \in \Theta$ the system dynamics one, and $\boldsymbol{\theta}_r \in \Theta$ the constraints one. Note that the parametrization comprehends also the system dynamics, hence the method can be used to learn the *best* control model that allows to obtain the desired performance. Contrary to the Learning Dynamics approach, that seeks to characterize the system itself, in the Learning Design characterization there is no identification process but a direct definition of the most *suitable* model parameters.

Most solution methods are derived from the machine learning framework, e.g. Reinforcement Learning [38, 70], Bayesian Optimization [39, 71], Particle Swarm Optimization [40] and Genetic Algorithm [41]. Due to the possibility of solving multi-objective optimization problems with constraints, Genetic Algorithms (GAs) result to be a common approach for tuning complex NMPC controllers [41–44]. Using GA, in fact, it is possible to define an optimal tuning problem that considers and balances different aspects of the desired

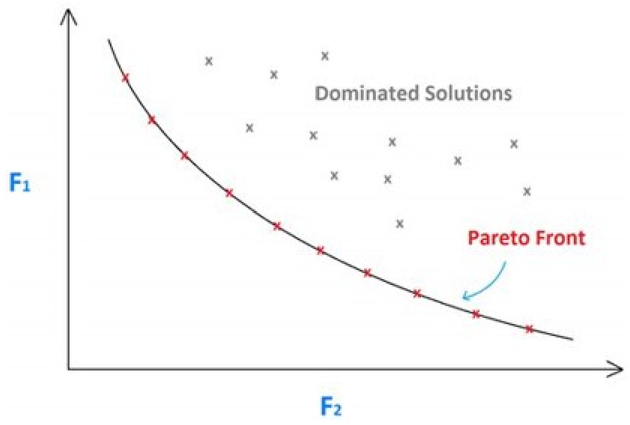


Figure 2.2. Example of a 2-dimensional Pareto front.

control performance. In particular, the robustness of the control system to parameters changes is a desiderable characteristic that should be considered alongside the control performance. In fact, in practical problems, if the optimal solution is sensitive to small changes of the design variables, the *implemented* solution, e.g. realized with limited accuracy, may result in a highly different objective values by affecting the real performance of the controller. Indeed, in the case the environment changes slightly or the parameters are affected by uncertainty, the solution on a high plateau of the performance objective is expected to yield more consistent performance than the solution on the peak [72]. A solution proposed in literature is to enforce robustness during optimization, which can be done either by correcting the original cost function with a robustness-related element or by inserting an additional optimization criterion assessing robustness [73]. The latter technique consists in casting a single-objective optimization problem into a multi-objective one where a second robustness objective is added [74, 75].

In the following of the thesis, Multi-Objective Genetic Algorithms will be used to achieve the desired data-driven tuning of the NMPC Controller, and hence their definition is reported here for completeness.

2.3.1 Multi-Objective Genetic Algorithm

Multi-objective optimization (MOO) is the problem of finding a vector of decision variables (or parameters) which satisfies constraints and optimizes a vector field, whose elements represent the conflicting objective functions [76]. In this framework, no single global solution is obtained, and a whole set of points can be defined as optima. Specifically, the multi-objective optimality is defined as follows [76]

Definition 2.3.1 (Pareto optimality)**.** A point, $\mathbf{p}^\star \in P \subset \mathbb{R}^l$, is *Pareto optimal* with respect to P *iff* there does not exist another point, $\mathbf{p} \in P$, such that $J(\mathbf{p}) \leq J(\mathbf{p}^\star)$ and $J_i(\mathbf{p}) < J_i(\mathbf{p}^\star)$ for at least one objective functions.

The points that satisfy Def. (2.3.1) are called Pareto-optimal solutions (or *non-dominated* solutions), and they constitute the *Pareto front*. An example of a 2-dimensional Pareto front is depicted in Fig. 2.2.

To solve MOO, GAs are widely used because they are derivatives-free, can handle non-

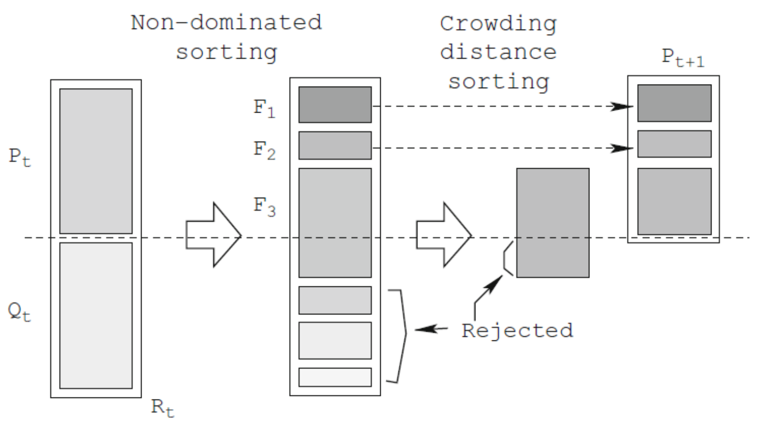


Figure 2.3. NSGA-II procedure summary.

convex problems and can be implemented by parallelization to reduce the computational burden. Specifically, GA is a stochastic optimization technique inspired by the process of natural selection the involves four steps depicted in Fig. 2.3:

1. initialization of population with a set of individuals that represent a possible solution within the constrained search space;
2. evaluation of the fitness functions, to determine the performance of each individual with respect to the optimizations objectives;
3. selection of the individuals with the higher performances for the next generation;
4. crossover and mutation of the selected individuals to produce a new generation.

Cycling on steps (2)-(4) continues until a termination condition is met, e.g., the maximum number of generations is reached or a specific threshold has been exceeded.

NSGA-II

The Non-dominated Sorting Genetic Algorithm (NSGA-II) [77] is a very popular algorithm to solve constrained multi-objective optimization problems. It is a global multi-objective optimization algorithm proposed as an improvement of NSGA [78] by introducing the elite strategy based on the crowding distance and rank operator of fast non-dominated sorting algorithm. With this approach, the number of computations to complete a generation is $O(MN^2)$, where M is the number of objectives and N is the population size. A crowded-comparison approach is implemented to preserve diversity in the solution. This method is based on a density-estimation metric and a crowded-comparison operator, and it does not require any user-defined parameter for maintaining diversity among population members.

Constraint-Handling Approach

In the presence of constrained problem, each founded optimal solution can be either feasible or infeasible. In this case, the NSGA-II adopted a constraint handling approach based on the following definition.

Definition 2.3.2. A solution i is said to constrained-dominate a solution j, if any of the following conditions is true.

1. Solution i is feasible and solution j is not.
2. Solutions i and j are both infeasible, but solution i has a smaller overall constraint violation.
3. Solutions i and j are feasible and solution i dominates solution j.

The effect of using this constrained-domination principle is that any feasible solution has a better non-domination rank than any infeasible solution.

2.4 Learning Dynamics

The model of the system is a fundamental element of the NMPC algorithm, hence its correct definition is crucial to obtain satisfactory closed-loop performance. In this sense, learning dynamics strategies aim at defining or refining the system model to be used within the NMPC.

In the framework of learning dynamics, the learning-based modeling is usually considered in the discrete formulation, obtaining an expression of the system state at the next discrete time instants. NMPC modeling, on the other hand, is usually derived starting from the continuous-time model, i.e. acceleration definition, that is then integrated through a numerical integrator (e.g. Euler, Runge-Kutta, etc.). Incorporating the learning-based modeling within the accelerations ensures that both the regression process and the obtained model are independent on the time step and hence regression can be accomplished with available data at any frequency. Then, using this characterization, it is possible to arbitrarily change the NMPC integration step. This feature is particularly valuable in the design phase, or if any change in the control system task is needed. Moreover, non-uniform integration grids can be adopted, achieving high reductions in computational burden. Therefore, all the results presented in the thesis, except for Ch. 8 and the comparison in Appendix B, are defined as continuous-time model.

In the following, a description of the discrete-time dynamics is reported, and the definition of continuous-time acceleration is introduced. The learning-based definition assumes to employ Gaussian Process Regression, which is detailed in Ch. 3, but different methodologies could similarly apply. Both black- and grey-box models are characterized, either exploiting a nominal model and model-mismatch data or directly input-output system measurements, respectively. Explicit definition of the GP-based model and its characterization are given. A thorough comparison of the approaches in an experimental scenario is reported in Appendix B.

2.4.1 System Dynamics

Consider the following *real* system

$$\dot{\boldsymbol{\xi}}(t) = \boldsymbol{f}(\boldsymbol{\xi}(t), \boldsymbol{u}(t)), \tag{2.10}$$

where $\boldsymbol{\xi} \in \mathbb{R}^{n_\xi}$ is the system state and $\boldsymbol{u} \in \mathbb{R}^{n_u}$ is the system input, and denote by

$$\dot{\tilde{\boldsymbol{\xi}}}(t) = \tilde{\boldsymbol{f}}(\boldsymbol{\xi}(t), \boldsymbol{u}(t)) \tag{2.11}$$

the *nominal dynamics*, i.e. a model of the system evolution.

Since the model is usually derived from a physical description, the state (both actual and nominal) can be divided as

$$\boldsymbol{\xi} = [\boldsymbol{q}, \dot{\boldsymbol{q}}]^\top, \tag{2.12}$$

where $\boldsymbol{q} \in \mathbb{R}^{n_q}$ and $\dot{\boldsymbol{q}} \in \mathbb{R}^{n_q}$ are the position and velocities of the physical system, respectively[1].

For T time instants, the current system state $\boldsymbol{\xi}_k$ and inputs $\boldsymbol{u}_k$, the actual velocities at the following time instant $\dot{\boldsymbol{q}}_{k+1}$, and the real accelerations of the system $\ddot{\boldsymbol{q}}_k$ can be grouped as

$$\begin{aligned} \Xi &= \{\boldsymbol{\xi}_0, \boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_{T-1}\} \\ \Upsilon &= \{\boldsymbol{u}_0, \boldsymbol{u}_1, \dots, \boldsymbol{u}_{T-1}\} \\ \dot{\mathcal{Q}} &= \{\dot{\boldsymbol{q}}_1, \dot{\boldsymbol{q}}_2, \dots, \dot{\boldsymbol{q}}_T\} \\ \ddot{\mathcal{Q}} &= \{\ddot{\boldsymbol{q}}_0, \ddot{\boldsymbol{q}}_1, \dots, \ddot{\boldsymbol{q}}_{T-1}\} \end{aligned} \tag{2.13}$$

where $\dot{\mathcal{Q}} \subset \Xi$ has been explicitly defined for notation convenience, and the *real* accelerations $\ddot{\boldsymbol{q}}$ can be obtained by IMU systems or discrete differentiation of the system velocities $\dot{\boldsymbol{q}}$.

2.4.2 Learning-based Discrete Dynamics

Define a numerical integration operator $\tilde{\boldsymbol{\phi}}$, e.g. 4th order Explicit Runge-Kutta, that returns the state at t_{k+1} with $\boldsymbol{x}(0) = \boldsymbol{x}_k$, i.e.

$$\tilde{\boldsymbol{\xi}}_{k+1} = \tilde{\boldsymbol{\phi}}(\boldsymbol{\xi}_k, \boldsymbol{u}_k), \tag{2.14}$$

where $\tilde{\boldsymbol{\xi}}_{k+1}$ is the discrete nominal dynamics evolution. Using the integrator and the quantities in (2.13), it is possible to compute the nominal velocities

$$\dot{\tilde{\mathcal{Q}}} = \{\dot{\tilde{\boldsymbol{q}}}_1, \dot{\tilde{\boldsymbol{q}}}_2, \dots, \dot{\tilde{\boldsymbol{q}}}_T\}. \tag{2.15}$$

Those quantities can then be used to train a GP that compensates for the unknown dynamics of the system, and estimates the *velocity prediction errors* $\dot{\boldsymbol{q}}_k - \dot{\tilde{\boldsymbol{q}}}_k$. In particular, for each component of the velocity vector $\dot{\boldsymbol{q}}$, the prediction error has been modeled with a distinct and independent GP. The GPs input vector at time t_k is the collection $\boldsymbol{\xi}_{k-1}$ and $\boldsymbol{u}_{k-1}$. The output of the i-th GP is $y_k^i = \dot{q}_k^i - \dot{\tilde{q}}_k^i$, where $\dot{q}_k^i$ and $\dot{\tilde{q}}_k^i$ are, respectively, values of the measured and predicted i-th component of $\dot{\boldsymbol{q}}$. The GPR methodology considered is detailed in Ch. 3.

[1]This state definition is common for physical systems, but the framework generalizes to any system definition.

The overall discrete *grey-box* model of the system dynamics is expressed by

$$\begin{aligned} \boldsymbol{\xi}_{k+1} &= \tilde{\boldsymbol{\phi}}(\boldsymbol{\xi}_k, \boldsymbol{u}_k) &&+ \quad \bar{\boldsymbol{\phi}}(\boldsymbol{\xi}_k, \boldsymbol{u}_k) \\ \begin{bmatrix} \boldsymbol{q}_{k+1} \\ \dot{\boldsymbol{q}}_{k+1} \end{bmatrix} &= \begin{bmatrix} \tilde{\boldsymbol{\phi}}_q(\boldsymbol{\xi}_k, \boldsymbol{u}_k) \\ \tilde{\boldsymbol{\phi}}_{\dot{q}}(\boldsymbol{\xi}_k, \boldsymbol{u}_k) \end{bmatrix} &&+ \quad \begin{bmatrix} \bar{\boldsymbol{\phi}}_{\dot{q}}(\boldsymbol{\xi}_k, \boldsymbol{u}_k) \cdot \frac{T_s}{2} \\ \bar{\boldsymbol{\phi}}_{\dot{q}}(\boldsymbol{\xi}_k, \boldsymbol{u}_k) \end{bmatrix} \end{aligned} \tag{2.16}$$

where $\bar{\boldsymbol{\phi}}_{\dot{\boldsymbol{q}}}$ is the estimate of the velocity prediction errors, T_s is the sampling interval, and the acceleration has been considered constant within the sampling interval. In this formulation, a *black-box* model, meaning that it is characterized without any nominal description of the accelerations, can be easily obtained by defining $\tilde{\boldsymbol{\phi}}_{\dot{q}}(\boldsymbol{\xi}_k, \boldsymbol{u}_k) = \mathbb{I}_{n_q}$. Note that, in this case, the velocity prediction of the nominal model is $\tilde{\dot{\boldsymbol{q}}}_k = \dot{\boldsymbol{q}}_{k-1}$, hence the model mismatch estimated through the GP is the *velocity increment* $\dot{\boldsymbol{q}}_{\boldsymbol{k}} - \dot{\boldsymbol{q}}_{\boldsymbol{k-1}}$.

LbNMPC formulation

To include the learning-based dynamics, the continuity constraint in the NLP (2.3c) is modified as

$$0 = \boldsymbol{\xi}_{j+1|k} - \left[\tilde{\boldsymbol{\phi}}(\boldsymbol{\xi}_{j|k}, \boldsymbol{u}_{j|k}) + \bar{\boldsymbol{\phi}}(\boldsymbol{\xi}_{j|k}, \boldsymbol{u}_{j|k})\right], \quad j = 0, 1, \dots, N-1, \tag{2.17}$$

and consequently the QP approximation (2.5) is reformulated substituting (2.5d) as

$$\Delta\boldsymbol{\xi}_{j+1} = (A^l_{j,ODE} + A^l_{j,GP})\Delta\boldsymbol{\xi}_j + (B^l_{j,ODE} + B^l_{j,GP})\Delta\boldsymbol{u}_j + \boldsymbol{a}^l_{j,ODE-GP}, \tag{2.18}$$

where

$$\begin{aligned} &A^l_{j,ODE} = \frac{\partial\tilde{\boldsymbol{\phi}}}{\partial\boldsymbol{x}_j}, \quad B^l_{j,ODE} = \frac{\partial\tilde{\boldsymbol{\phi}}}{\partial\boldsymbol{u}_j}, \\ &A^l_{j,GP} = \frac{\partial\bar{\boldsymbol{\psi}}}{\partial\boldsymbol{x}_j}, \quad B^l_{j,GP} = \frac{\partial\bar{\boldsymbol{\psi}}}{\partial\boldsymbol{u}_j}, \\ &\boldsymbol{a}^l_{j,ODE-GP} = \tilde{\boldsymbol{\phi}}(\boldsymbol{x}^l_j, \boldsymbol{u}^l_j) + \bar{\boldsymbol{\psi}}(\boldsymbol{x}^l_j, \boldsymbol{u}^l_j) - \boldsymbol{x}^l_{j+1}. \end{aligned} \tag{2.19}$$

2.4.3 Learning-based Continuous Dynamics

Using the quantities in (2.13), it is possible to compute the *nominal* accelerations $\tilde{\ddot{\boldsymbol{q}}}$ applying Eq. (2.11) to get

$$\tilde{\ddot{\mathcal{Q}}} = \{\tilde{\ddot{\boldsymbol{q}}}_1, \tilde{\ddot{\boldsymbol{q}}}_2, \dots, \tilde{\ddot{\boldsymbol{q}}}_{T-1}\}. \tag{2.20}$$

Those quantities can be used for GPR, compensating for the unknown dynamics of the system, and estimating the *acceleration prediction errors* $\ddot{\boldsymbol{q}}_k - \tilde{\ddot{\boldsymbol{q}}}_k$. In particular, for each component of the acceleration vector $\ddot{\boldsymbol{q}}$, we model the estimation error with a distinct and independent GP. The GPs input vector at time t_k contains both $\boldsymbol{\xi}_k$ and $\boldsymbol{u}_k$. The corresponding output of the i-th GP is $y^i_k = \ddot{q}^i_k - \tilde{\ddot{q}}^i_k$, where $\ddot{q}^i_k$ and $\tilde{\ddot{q}}^i_k$ are, respectively, the measured and predicted i-th component of $\ddot{\boldsymbol{q}}$. The GPR methodology considered is detailed in Ch. 3.

The overall continuous *grey-box* model of the system dynamics is defined by

$$\begin{aligned} \hat{\dot{\boldsymbol{\xi}}}(t) &= \tilde{\dot{\boldsymbol{\xi}}}(t) \quad &+& \quad \bar{\dot{\boldsymbol{\xi}}}(t) \\ \begin{bmatrix} \hat{\dot{\boldsymbol{q}}}(t) \\ \hat{\ddot{\boldsymbol{q}}}(t) \end{bmatrix} &= \begin{bmatrix} \tilde{\dot{\boldsymbol{q}}}(t) \\ \tilde{\ddot{\boldsymbol{q}}}(t) \end{bmatrix} \quad &+& \quad \begin{bmatrix} \boldsymbol{0} \\ \bar{\ddot{\boldsymbol{q}}}(t) \end{bmatrix} \end{aligned} \tag{2.21}$$

where $\bar{\ddot{\boldsymbol{q}}}$ is the estimate of the acceleration prediction errors. Using this formulation, a *black-box* model, i.e. with no nominal characterization of the accelerations, can be easily obtained by defining $\tilde{\ddot{\boldsymbol{q}}}(t) = 0$.

Finally, define a numerical integrator operator

$$\hat{\boldsymbol{\xi}}_{k+1} = \hat{\boldsymbol{\phi}}(\boldsymbol{\xi}_k, \boldsymbol{u}_k), \tag{2.22}$$

that integrates the dynamics $\hat{\dot{\boldsymbol{\xi}}}(t)$ and returns the solution at t_{k+1} with $\boldsymbol{\xi}(0) = \boldsymbol{\xi}_k$, e.g. 4th order Explicit Runge-Kutta.

LbNMPC formulation

To include the learning-based dynamics, the continuity constraint in the NLP (2.3c) is modified as

$$0 = \boldsymbol{\xi}_{j+1|k} - \hat{\boldsymbol{\phi}}(\boldsymbol{\xi}_{j|k}, \boldsymbol{u}_{j|k}),\ j = 0, 1, \ldots, N-1 \tag{2.23}$$

and consequently the QP approximation (2.5) is reformulated substituting (2.5d) as

$$\Delta\boldsymbol{\xi}_{j+1} = \hat{A}^l_j \Delta\boldsymbol{\xi}_j + \hat{B}_j \Delta\boldsymbol{u}_j + \boldsymbol{a}^l_j, \tag{2.24}$$

where

$$\begin{gathered} \hat{A}^l_j = \frac{\partial\hat{\boldsymbol{\phi}}}{\partial\boldsymbol{\xi}_j}, \quad \hat{B}^l_j = \frac{\partial\hat{\boldsymbol{\phi}}}{\partial\boldsymbol{u}_j}, \\ \boldsymbol{a}^l_j = \hat{\boldsymbol{\phi}}(\boldsymbol{x}^l_j, \boldsymbol{u}^l_j) - \boldsymbol{x}^l_{j+1}. \end{gathered} \tag{2.25}$$

Note that the continuous formulation does not modifies the NLP and QP definition with respect to the integrator function $\hat{\boldsymbol{\phi}}$, but just modifies the integrator definition itself. This allows to easily change the integration step of the NMPC and to apply Non-Uniform Grids for dynamics integration.

3 Gaussian Process Regression for Dynamics Learning

Gaussian Process Regression (GPR) has proven to be a favored approach to address the problem of learning dynamics in the context of LbMPC, since it is a flexible nonparametric stochastic approach [15]. Moreover, it is apt for active learning strategies [79], it generally avoids overfitting, and it is suitable also for advanced frameworks in which stochastic state distributions are propagated over the prediction horizon, e.g. stochastic MPC.

In the following, the basics of GPR are described, and the relevant Gaussian Process model approximations to be used in real-time LbNMPC applications are reported.

3.1 Gaussian Process Regression

A Gaussian Process (GP) can be seen as a generalization of the Gaussian probability distribution, that describes random variables, to the distribution over function. A formal definition is [29]

Definition 3.1.1 (Gaussian Process)**.** A Gaussian Process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

In this context, the random variables represent the value of a function $f(x)$ at input location x, and a generic GP can be expressed as

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')), \tag{3.1}$$

where mean $m(x)$ and covariance functions $k(x, x')$ completely characterize the GP $f(x)$, and are defined as

$$m(x) = \mathbb{E}[f(x)], \tag{3.2a}$$
$$k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))]. \tag{3.2b}$$

In the context of learning dynamics framework, GPs have been applied considering a dataset $\mathcal{X}$ containing T noisy observations y of the system and a multidimensional input $\boldsymbol{x}$ containing the systems states, inputs and possibly derived quantities. The regression task assumes the following probabilistic model

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_T \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{x}_1) \\ \vdots \\ f(\boldsymbol{x}_T) \end{bmatrix} + \begin{bmatrix} e_1 \\ \vdots \\ e_T \end{bmatrix} = \boldsymbol{f} + \boldsymbol{e},$$

where $\mathbf{y}$ is a vector containing the n noisy observations, $\boldsymbol{e}$ is zero mean Gaussian noise[1] with standard deviation σ_n, and $\boldsymbol{f} \sim N(0, K_{\boldsymbol{xx}})$ is a zero mean Gaussian process with covariance $K_{\boldsymbol{xx}}$ defined through a kernel function $k(\cdot,\cdot)$. More precisely, the covariance between the output at time t_k and t_j, i.e. the element of $K_{\boldsymbol{xx}}$ at row k and column j, is $E[y_k y_j] = k(\boldsymbol{x}_k, \boldsymbol{x}_j)$. As detailed in [29], the posterior distribution of $\boldsymbol{f}$ in a target input location $\boldsymbol{x}_*$ is Gaussian, and its maximum a posteriori estimator is the posterior mean $\mu(\boldsymbol{x}_*)$, given by

$$\mu(\boldsymbol{x}_*) = \mathbf{k}_{\boldsymbol{x}_*\boldsymbol{x}}^\top (K_{\boldsymbol{xx}} + \sigma_n^2 \mathbb{I})^{-1} \mathbf{y}, \tag{3.3}$$

with variance

$$\Sigma(\boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \mathbf{k}_{\boldsymbol{x}_*\boldsymbol{x}}^\top (K_{\boldsymbol{xx}} + \sigma_n^2 \mathbb{I})^{-1} \mathbf{k}_{\boldsymbol{x}_*\boldsymbol{x}}, \tag{3.4}$$

where $\mathbf{k}_{\boldsymbol{x}_*\boldsymbol{x}}$ is the vector of covariances between the test point and the training points, and $\mathbb{I} \in \mathbb{R}^{n\times n}$ is the identity matrix.

A key aspect of GPR is the choice of the kernel, which encodes the prior assumptions on the unknown function. In particular, stationary kernels define the similarity of two points based on their weighted distance

$$d(\boldsymbol{x}_k, \boldsymbol{x}_j)^2 = (\boldsymbol{x}_k - \boldsymbol{x}_j)^\top \mathcal{L}^{-2} (\boldsymbol{x}_k - \boldsymbol{x}_j), \tag{3.5}$$

where $\mathcal{L}$ is a positive diagonal matrix, with the diagonal elements named lengthscales. Stationary kernels are widely used thanks to their invariance with respect to time instants, making them well-suited for dynamics learning. In the presented works, different well-studied stationary kernels have been considered:

- the Squared Exponential kernel, defined by

$$k_{\mathrm{SQ}}(\boldsymbol{x}_k, \boldsymbol{x}_j) = \sigma_n \exp(-d(\boldsymbol{x}_k, \boldsymbol{x}_j)^2)\,, \tag{3.6}$$

- the Rational Quadratic kernel, defined by

$$k_{\mathrm{RQ}}(\boldsymbol{x}_k, \boldsymbol{x}_j) = \sigma_n \left(1 + \frac{d(\boldsymbol{x}_k, \boldsymbol{x}_j)^2}{2\alpha_{\mathrm{rq}}}\right)^{-\alpha_{\mathrm{rq}}}, \quad \alpha_{\mathrm{rq}} > 0\,, \tag{3.7}$$

- the Matern 3/2 kernel, defined by

$$k_{\mathrm{M32}}(\boldsymbol{x}_k, \boldsymbol{x}_j) = \sigma_n \left(1 + \sqrt{3} d(\boldsymbol{x}_k, \boldsymbol{x}_j)\right) \exp\left(-\sqrt{3} d(\boldsymbol{x}_k, \boldsymbol{x}_j)\right)\,, \tag{3.8}$$

- the Matern 5/2 kernel, defined by

$$k_{\mathrm{M52}}(\boldsymbol{x}_k, \boldsymbol{x}_j) = \sigma_n \left(1 + \sqrt{5} d(\boldsymbol{x}_k, \boldsymbol{x}_j) + \frac{5}{3} d(\boldsymbol{x}_k, \boldsymbol{x}_j)^2\right) \exp\left(-\sqrt{5} d(\boldsymbol{x}_k, \boldsymbol{x}_j)\right). \tag{3.9}$$

The regression task is then the selection of the best hyperparameters, that are composed of the kernel function parameters (e.g., lengthscales) and the noise variance σ_n, and they can be tuned relying on empirical methods, such as cross validation, or by maximization

[1] $\boldsymbol{e}$ usually represents the measurement noise, but, in the simulation frameworks presented in the thesis, it will represent a regularization parameter that allows to obtain smoother GP characterizations.

of the training data Marginal Likelihood (ML) [29]. Importantly, the hyperparameter σ_n defines the trade-off between adherence to the training data and regularization. High values of σ_n promote regularization, i.e., the smoothness of $\boldsymbol{f}(\boldsymbol{x})$, but could compromise accuracy. On the other hand, too small values of σ_n could lead to non-smooth functions, which might be an issue for the LbNMPC solution.

3.2 GP model approximation for LbNMPC

Even though GPs provide a powerful Bayesian Regression framework, that is nonparametric and interpretable, they suffer from scalability problems. Hence, to apply them in the LbNMPC framework, sparse approximations are usually required, see e.g. [30, 31]. Approximations can be both global, which distillate the entire original dataset $\mathcal{X}$ into a lower dimensional dataset maintaining the information on the whole input space, and local, which divide data for subspace learning [80]. In this section, a description of the employed global approximators, namely, an heuristic procedure named Subset of Data (SoD), Variational Free Energy (VFE) [81], and Fully Independent Training Conditional (FITC) [82, 83], is given. Moreover, the Nearest Neighbor (NN) approach [84] and the application of FITC to online local approximation, namely Transduction Learning (TL) [85], are described. Finally, two feature selection procedures are illustrated, one bottom-up iterative incremental approach [86] and one top-down permutation-based feature set reduction [87].

3.2.1 Global Approximations

Global approximations rely on the hypothesis that the information stored in the original dataset $\mathcal{X}$ can be synthesized through a set of inducing points, obtaining a reduced dataset $\mathcal{X}_u$. They can be used *offline*, i.e. in the data gathering and modeling phases, to reduce the dimensionality of the whole GP model and hence the computational burden of the LbNMPC problem.

Subset of Data

This heuristic procedure aims at reducing the original dataset $\mathcal{X}$ through an iterative algorithm that exploits information provided by the posterior variance in Eq. (3.4). First, the reduced dataset $\mathcal{X}_u$ is initialized with the first data point of $\mathcal{X}$. Then, the algorithm iterates over the remaining points in $\mathcal{X}$. For each point, the algorithm computes Eq. (3.4) using as training data the current points in $\mathcal{X}_u$, and includes the point in $\mathcal{X}_u$ if the variance is higher than a certain threshold provided by the user. The basic idea behind this approach is that if the model is confident in the current input location the point can be neglected, while if the variance is high we have to add the point to $\mathcal{X}_u$ to improve prediction accuracy. Once the reduced dataset has been obtained, the same formula defined for *standard* inference is used, i.e. Eq. (3.3).

Variational Free Energy

VFE inference is based on the idea that the data contained in the original dataset $\mathcal{X}$ can be reproduced through a group of inducing points. Hence, a collection of $T_u < T$ inducing points $\mathcal{X}_u$ can be used to compute each regression target mean with a significant lower computational effort. The method seeks to reduce the discrepancy between the full posterior GP and its approximation, i.e. the training of VFE minimizes the following upper bound on the Negative Log-Likelihood [81]

$$\min_{\boldsymbol{\Theta},\mathcal{X}_u} = \left\{ \frac{1}{2}\log\left(\det\left(Q_{\boldsymbol{xx}} + \sigma_n^2\mathbb{I}\right)\right) + \frac{1}{2}\mathbf{y}^\top\left(Q_{\boldsymbol{xx}} + \sigma_y^2\mathbb{I}\right)^{-1}\mathbf{y} + \frac{1}{2\sigma_y^2}\mathrm{tr}\left(K_{\boldsymbol{xx}} - Q_{\boldsymbol{xx}}\right) + \frac{n}{2}\log(2\pi) \right\}, \tag{3.10}$$

where $Q_{\boldsymbol{xx}} = K_{\boldsymbol{xu}}K_{\boldsymbol{uu}}^{-1}K_{\boldsymbol{ux}}$. Once the resulting optimal values are obtained, the predictions at the test point $\boldsymbol{x}_*$ can be computed as

$$\mu(\boldsymbol{x}_*) = \mathbf{k}_{\boldsymbol{x}_*\boldsymbol{u}}\left(\sigma_n^2 K_{\boldsymbol{uu}} + K_{\boldsymbol{ux}}K_{\boldsymbol{xu}}\right)^{-1} K_{\boldsymbol{ux}}\mathbf{y}. \tag{3.11}$$

Fully Independent Training Conditional

FITC seeks to find a set of m inducing points of the original dataset $\mathcal{X}$ that can condense the information with a significantly lower number of data. In particular, the method minimizes the following term [88]

$$\min_{\boldsymbol{\Theta},\mathcal{X}_u} = \left\{ \frac{1}{2}\log\left(\det\left(Q_{\boldsymbol{xx}} + T + \sigma_n^2\mathbb{I}\right)\right) + \frac{1}{2}\mathbf{y}^\top\left(Q_{\boldsymbol{xx}} + T + \sigma_y^2\mathbb{I}\right)^{-1}\mathbf{y} + \frac{n}{2}\log(2\pi) \right\}, \tag{3.12}$$

where $Q_{\boldsymbol{xx}} = K_{\boldsymbol{xu}}K_{\boldsymbol{uu}}^{-1}K_{\boldsymbol{ux}}$ and $T = \mathrm{diag}(K_{\boldsymbol{xx}} - Q_{\boldsymbol{xx}})$. In this case, the posterior results

$$\mu(\boldsymbol{x}_*) = \mathbf{k}_{\boldsymbol{x}_*\boldsymbol{u}}\Sigma K_{\boldsymbol{ux}}\Lambda^{-1}\mathbf{y}, \tag{3.13}$$

where

$$\Sigma = (K_{\boldsymbol{uu}} + K_{\boldsymbol{ux}}\Lambda^{-1}K_{\boldsymbol{xu}})^{-1}, \tag{3.14}$$

$$\Lambda = \mathrm{diag}(K_{\boldsymbol{xx}} - K_{\boldsymbol{xu}}K_{\boldsymbol{uu}}^{-1}K_{\boldsymbol{ux}} + \sigma_n^2\mathbb{I}). \tag{3.15}$$

3.2.2 Local Approximations

Local approximations provide GP models valid only in the neighborhood of the current input, and they should be used *online*, i.e. while the NMPC controller is running, and they are particularly useful to further reduce the computational burden of the control algorithm. The methods rely on the knowledge of the last computed predicted trajectory in terms of states and inputs of the system $\{\boldsymbol{\xi}_{\cdot|i-1}, \boldsymbol{u}_{\cdot|i-1}\}$ and compute the predicted GP inputs $\boldsymbol{x}_{\cdot|i-1}$ associated. These procedures can be accomplished at runtime before the LbNMPC call, updating the GP-related parameters within the Nonlinear Programming Problem (see Sec. 2.4). The main strength of these approaches is that they exploit the structure of the NMPC problem in order to simplify the GP-based predictive model. Indeed, inducing variables may be placed directly along the predicted NMPC trajectory, as it is highly probable that the system at the next instant will be in that region. More formally, the evolution of the GP inputs to be used for regression based on the future trajectory prediction, i.e. steps $l = 0 \dots N$ of the previous NMPC iteration $i-1$, may be summarized as

$$\boldsymbol{x}_{\cdot|i-1} = \left[\boldsymbol{x}_{0|i-1}{}^{\top}, \boldsymbol{x}_{1|i-1}{}^{\top}, \dots, \boldsymbol{x}_{N|i-1}{}^{\top}\right]^{\top}. \tag{3.16}$$

Nearest Neighbor

This approach is based on the key idea that the closest points to the target are the most informative for a local prediction of the GP [84]. In particular, the closest are defined by the distance considering the trained lengthscales L using a weighted norm, as defined in Eq. 3.5. The search for the *nearest points* can be done on a previously obtained inducing set $\mathcal{X}_u$, further reducing the number of inducing points $T_p < T_u < T$. Once the points have been selected, the quantities to obtain the posteriori mean of the GP $\mu(\boldsymbol{x}_*)$ have to be recomputed, substituting in Eq. (3.3) the entire training measures $\mathbf{y}$ with the measure subset relative to the chosen points.

Transduction Learning

The transduction learning relies on an approximation adjusted according to the available information on the desired test points of the FITC method [85]. To reduce the computational burden, a subset of the previous trajectory may be employed for transduction. More precisely, given a *transduction ratio* $h \in \mathbb{N}$ and $r = \lfloor N/h \rfloor$, points may be sampled from the previous trajectory as

$$\mathcal{X}_s = \left[\boldsymbol{x}_{0|i-1}{}^{\top}, \boldsymbol{x}_{h|i-1}{}^{\top}, \boldsymbol{x}_{2h|i-1}{}^{\top}, \dots, \boldsymbol{x}_{(r-1)\cdot h|i-1}{}^{\top}\right]^{\top}. \tag{3.17}$$

The sampled points $\mathcal{X}_s$ can be used to provide a local GP approximation to be employed for the current LbNMPC control action. This is done using a general (possibly already sparse) GP model based on dataset $\mathcal{X}_u$ and the FITC approximating formula of the mean of the predictive distribution, Eq. (3.13).

3.2.3 Feature Selection

While the previously described approximation aims at reducing the number of points used for the GP prediction, in the feature analysis the objective is to select the most significant quantities to be included within the GP input $\boldsymbol{x}$. This procedure allows both to reduce the dimensionality of the GP and to avoid deceptive correlations within the GP. In particular, starting from a complete feature set $\bar{\boldsymbol{x}} \in \mathbb{R}^D$, only the most informative quantities may be selected as GP inputs $\boldsymbol{x} \subseteq \bar{\boldsymbol{x}}$. An evaluation procedure on all the 2^D possible feature subsets would be unfeasible, hence we propose an iterative incremental approach, similarly to [86], and a permutation-based feature set reduction [87].

Both the method apply the R^2 index across the whole dataset to measure the fit quality. The R^2 index is computed comparing the real target evolution y and the

Listing 3.1 Iterative incremental approach.

```
for acceleration_target
  good_features = [];
  for iter = 1:N
    for feature not in good_features
      simulate addition of feature to good_features;
      evaluate R^2;
    end
    add feature leading to max(R^2) to good_features;
    for feature in good_features
      simulate removal of feature from good_features;
      evaluate R^2;
      if R^2 increases
        remove feature;
      end
    end
  end
end
```

predicted evolution $\hat{y}$ as

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}, \tag{3.18}$$

where $SS_{res} = \sum_k (y_k - \hat{y}_k)^2 = \sum_k e_k^2$ is the residual sum of squares, $\hat{y}$ is the GP estimate, $SS_{tot} = \sum_k (y_k - \bar{y})^2$ is the total sum of squares and $\bar{y} = \frac{1}{T}\sum_{k=1}^{T} y_k$ is the mean value.

Iterative Incremental Approach

This approach is based on the idea of incrementally adding the feature that improves the most the prediction capability. We adopt a fitting quality to measure the feature importance, where full GP models should be trained on subsets of the complete training set, by performing exact inference.

Starting from and empty set of features, at each iteration we add the feature that, if added, would lead to the highest R^2 index, Eq. (3.18). Moreover, a backtracking step has been applied, where, at each iteration, a fitting analysis is performed to verify whether the removal of previously added features leads to an improvement of the fit index. This backward step allows to remove features that become redundant or harmful along the iterative procedure [86]. The general procedure is schematized in Lis. 3.1.

Permutation-Based Feature Set Reduction

This approach is based on the idea of removing the features that affect the least the prediction capability. Consider a modified dataset $(\pi^l(\mathcal{X}), \mathrm{y})$ where the l-th component of the GP inputs $\boldsymbol{x}$ in the whole dataset has been shuffled through a random permutation $\pi^l(\cdot)$. By training a GP on the *standard* dataset and another on the *modified* one, it is possible to define the permutation sensitivity as

$$S(l) = \frac{R^2(\mathcal{X}, \mathrm{y}) - R^2(\pi^l(\mathcal{X}), \mathrm{y})}{R^2(\mathcal{X}, \mathrm{y})}, \tag{3.19}$$

where the R^2-score is defined as in Eq. (3.18). The sensitivity evaluates the effect on the GP prediction capability of permuting a feature $\boldsymbol{x}^{(l)}$, hence a low sensitivity $S(l)$ implies that the l-th feature has a small impact on the prediction performance and can be omitted. By iteratively omitting the least important feature and then training a reduced model, the set of features can be reduced to the most important ones.

Part II

Two-wheel Vehicles

In this part, the applications related to two-wheel vehicles are presented. Chapter 4 describes an NMPC controller for a virtual motorcycle, that is the foundation of the LbNMPC presented in the next chapters. In particular, Ch. 5 reports the data-driven tuning of the controller in the learning design framework, while Ch. 6 illustrates a learning dynamics strategy applied to the virtual rider.

4

A NMPC for a Virtual Motorcycle in Real-Time

In this Chapter, the definition of a virtual rider, i.e. a controller for motorcycles in virtual environment (whose simulation model is depicted in 4.1), based on a real-time capable NMPC controller is described. The goal is to track a desired trajectory, both in terms of path and velocity profile, to ride the vehicle along a high-performance virtual circuit. Sec. 4.1 details the dynamical model underlying the controller, that includes the main out-of-plane dynamics and the rider movement. Performance of the controller is analyzed in Sec. 4.2, showing the cosimulation with an accurate real-time multibody simulation software for two challenging maneuvers, i.e., a chicane and a lap of a virtual track.

Figure 4.1. Representation of the virtual motorcycle in the simulation framework [89].

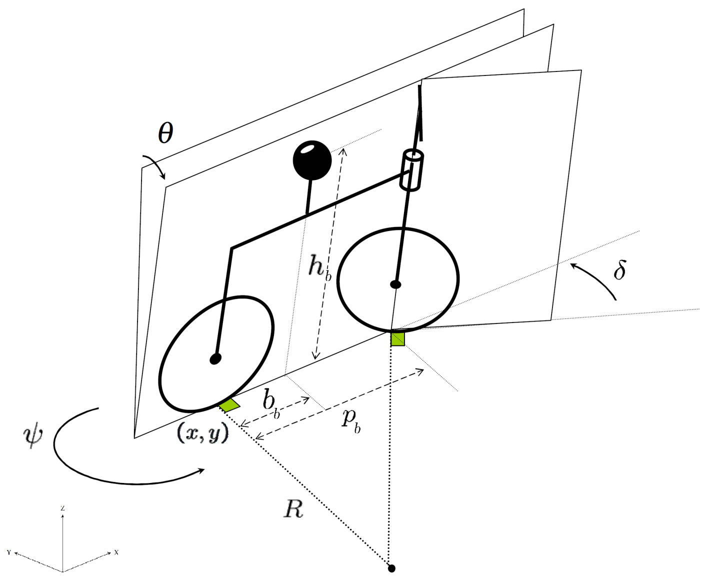


Figure 4.2. Scheme of the motorcycle model. The generalized coordinates for the model derivation and the most relevant parameters are shown.

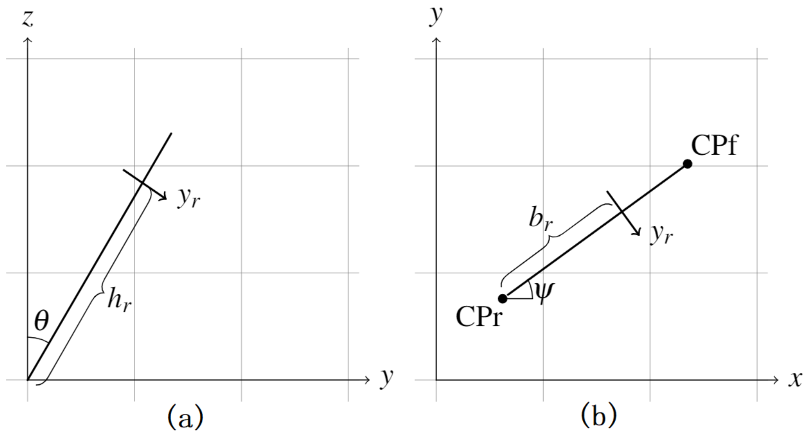


Figure 4.3. The point mass rider movement model. In (a), the movement direction and the sign convention from behind is shown, while in (b) the movement direction from upside (with motorcycle in vertical position) is shown.

4.1 Model for Control Synthesis

The non-linear dynamics model used in the NMPC algorithm is based on the sliding plane representation [90]. In particular, in the formulation proposed the plane can roll and slide both in x and y directions (see Fig. 4.2) and a moving point mass with lateral DoF, representative of the rider, is attached to the plane. The system is driven by drag and tire forces: the lateral tire forces are generated by using the *Pacejka Magic Formula*, while longitudinal forces are linearly dependent on throttle and brake. Specific contributions for the motorcycle modelling as the effect of caster and roll on the steering angle and the gyroscopic effects are also considered. Moreover, the maximum transmission torque is changed on-line to introduce the gearbox effect. A list of the model parameters is

Table 4.1. Main model quantities and parameters description.

Symbol	Description
x	longitudinal position
y	lateral position
ψ	yaw angle
θ	roll angle of the vehicle
β	sideslip angle of the vehicle
δ_f	steering angle of the front wheel
y_r	rider lateral position
m_b	vehicle mass
m_r	rider mass
h_b	height of the vehicle' CM
h_r	height of the rider' CM
I	inertia of the vehicle
b_b	rear wheels to CM vehicle longitudinal distance
b_r	rear wheels to CM rider longitudinal distance
$F_{\{x,y\}i}$	wheels lateral/longitudinal forces in the tire frames
F_d	longitudinal drag force in the vehicle frame
F_{z_i}	normal forces on the wheels
s	curvilinear abscissa
e_y	lateral tracking error
e_ψ	angular tracking error

presented in Tab. 4.1.

4.1.1 Dynamics

The dynamics have been derived through the Euler-Lagrangian method [90] by using the generalized coordinates

$$\boldsymbol{q} = [x, y, \psi, \theta]^\top \,, \tag{4.1}$$

where (x, y) are the rear wheel contact patch position coordinates, ψ is the yaw angle, and θ is the roll angle, as shown in Fig. 4.2. The rider is modelled as a point mass that can move in the direction perpendicular to the motorcycle vertical axis, as shown in Fig. 4.3.

The Lagrangian equation is $L = K_b + K_r - U_b - U_r$, where the kinetic and potential energy are[1]

$$\begin{aligned}
K_b &= \frac{1}{2} m_b (\dot{x}_{cm,b}^2 + \dot{y}_{cm,b}^2 + \dot{z}_{cm,b}^2) + \frac{1}{2}\omega_b^\top I \omega_b, \\
K_r &= \frac{1}{2} m_r (\dot{x}_{cm,r}^2 + \dot{y}_{cm,r}^2 + \dot{z}_{cm,r}^2), \\
U_b &= m_b\, g\, h_b\, c_\theta, \;\; U_r = m_r\, g\, h_r\, c_\theta.
\end{aligned} \tag{4.2}$$

[1]The subscript b stands for the motorcycle quantities and r for rider quantities and the notation $c_\alpha = cos(\alpha)$ and $s_\alpha = sin(\alpha)$ is used.

The center of mass (CM) positions and the angular velocity of the motorcycle are derived as

$$\begin{aligned} x_{cm,b} &= x + b_b\, c_\psi + h_b\, s_\theta s_\psi, & z_{cm,b} &= h_b\, c_\theta, \\ y_{cm,b} &= y + b_b\, s_\psi - h_b\, s_\theta c_\psi, & \omega_b &= [\dot\theta, \dot\psi s_\theta, \dot\psi c_\theta]^\top, \end{aligned} \tag{4.3}$$

the rider CM position as

$$\begin{aligned} x_{cm,r} &= x + b_r\, c_\psi + (h_r\, s_\theta + y_r\, c_\theta)\, s_\psi, \\ y_{cm,r} &= y + b_r\, s_\psi - (h_r\, s_\theta + y_r\, c_\theta)\, c_\psi, \\ z_{cm,r} &= h_r\, c_\theta - y_r\, s_\theta, \end{aligned} \tag{4.4}$$

and the parameters are: $m_{b/r}$, mass of the motorcycle/rider; $b_{b/r}$, rear contact patch to motorcycle/rider CM longitudinal distance; $h_{b/r}$, height of the motorcycle/rider CM; p_b, rear to front contact patch distance; $I = diag([I_{xx}, I_{yy}, I_{zz}])$, inertia matrix of the motorcycle. The effect of the forces on the system is derived using the generalized forces formulation [91]

$$Q_j = \sum_{k=1}^{N} F_k \cdot \frac{\partial r_k}{\partial q_j},\ j \in \{1..4\},\ k \in \{1..5\}, \tag{4.5}$$

where F_k is the force vector at point k and r_k is the position vector to point k, measured in the inertial reference frame. The resulting Q_j, $j \in \{1..4\}$ can be stacked and expressed as $B(\boldsymbol{q}) \cdot \boldsymbol{w}$ where $\boldsymbol{w}$ are the forces applied to the system, i.e.

$$\boldsymbol{w} = [F_{xf},\, F_{yf},\, F_{xr},\, F_{yr},\, F_d]^\top \tag{4.6}$$

where $F_{l,i}$, $l \in \{x, y\}$, $i \in \{f, r\}$ are the longitudinal/lateral front and rear wheel forces in the respective tire frame, F_d is the longitudinal drag force in the motorcycle frame. Note that, the angle used to characterize the direction of the forces on the front tire w.r.t. the motorcycle direction, namely the steering angle on the ground δ_G, is related to the handlebar angle δ as [92, p. 315-324]

$$\delta_G = atan\left(\frac{c_\epsilon s_\delta}{c_\theta c_\delta - s_\theta s_\epsilon s_\delta}\right), \tag{4.7}$$

where ϵ is the caster angle[2].

The equation of motion then results

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}}) + G(\boldsymbol{q}) = B(\boldsymbol{q})w. \tag{4.8}$$

The equation has been reformulated in a frame rotating jointly with the motorcycle around the global Z-axis, through the definition of a velocity transformation T and new velocity coordinates [93]

$$\dot{\boldsymbol{q}}' = \begin{bmatrix} v_x \\ v_y \\ \dot\psi \\ \dot\theta \end{bmatrix} = \begin{bmatrix} c_\psi & s_\psi & 0 & 0 \\ -s_\psi & c_\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot x \\ \dot y \\ \dot\psi \\ \dot\theta \end{bmatrix} = T^\top \dot{\boldsymbol{q}} \tag{4.9}$$

[2]The caster angle is the angular displacement of the steering axis from the vertical axis of the front wheel.

where v_x and v_y are longitudinal and lateral velocity in the new frame. Inverting and differentiating (4.9) yields

$$\ddot{\boldsymbol{q}} = T\ddot{\boldsymbol{q}}' + \dot{T}\dot{\boldsymbol{q}}' \tag{4.10}$$

that, substituting in (4.8) and premultiplying by $T^\top$, gives

$$T^\top M(\boldsymbol{q})T\ddot{\boldsymbol{q}}' + T^\top[M(\boldsymbol{q})\dot{T}\dot{\boldsymbol{q}}' + C(\boldsymbol{q}, T\dot{\boldsymbol{q}}')] + T^\top G(\boldsymbol{q}) = T^\top B(\boldsymbol{q})\boldsymbol{w}. \tag{4.11}$$

The resulting dynamic model is

$$\tilde{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}' + \tilde{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}') + \tilde{G}(\boldsymbol{q}) = \tilde{B}\boldsymbol{w}, \tag{4.12}$$

where

$$\begin{aligned} \tilde{M} &= T^\top M T, & \tilde{G} &= T^\top G, \\ \tilde{C} &= T^\top \left(M\dot{T}\dot{\boldsymbol{q}}' + C(\boldsymbol{q},\ T\dot{\boldsymbol{q}}')\right), & \tilde{B} &= T^\top B. \end{aligned} \tag{4.13}$$

4.1.2 Forces

The longitudinal tire forces are computed as

$$\begin{aligned} F_{xf} &= -\gamma_b \lambda \frac{\tau_b^f}{r_{wf}}, \\ F_{xr} &= -\gamma_b(1-\lambda)\frac{\tau_b^r}{r_{wr}} + \gamma_t \frac{\tau_t^M}{r_{wr}} + (1-\gamma_t)\frac{\tau_t^m}{r_{wr}} \end{aligned} \tag{4.14}$$

where $\gamma_t \in [0,1]$ is the normalized throttle input, $\gamma_b \in [0,1]$ is the normalized brake input, $\lambda \in [0,1]$ is the front-rear brake bias, $\tau_b^{f/r}$ is the maximum front/rear brake torque, τ_t^M is the maximum positive transmission torque, τ_t^m is the maximum negative torque given by engine braking, $r_{wf/r}$ are front and rear wheel radii. The maximum positive and negative transmission torques are parameters changed on-line depending on the current gear in order to model the effect of the gearshift.

The lateral tire forces are computed by using the *Pacejka Magic Formula* for motorcycles [94, p. 580, eq. 11.E19], i.e.

$$\begin{aligned} F_{y,i} = D_y^i\, sin\big[&C_y^i atan\left(B_y^i\alpha_i - E_y^i\left(B_y^i\alpha_i - atan(B_y^i\alpha_i)\right)\right) + \\ &+ C_\theta^i atan\left(B_\theta^i\theta - E_\theta^i\left(B_\theta^i\theta - atan(B_\theta^i\theta)\right)\right)\big], \end{aligned} \tag{4.15}$$

where α_i are the side slip angles of the tires

$$\alpha_r = atan(\frac{v_y}{v_x}), \quad \alpha_f = atan(\frac{v_y + p_b\dot{\psi}}{v_x}) - \delta_G. \tag{4.16}$$

The normal forces $F_{z\{f,r\}}$ are computed including an estimation of the load transfer torque as

$$F_{zf} = \frac{m_T\, b_T\, g - \tau_{LT}}{p_b}, \quad F_{zr} = \frac{m_T\,(p_b - b_T)\, g + \tau_{LT}}{p_b}. \tag{4.17}$$

where $\tau_{LT} = (F_{xf} + F_{xr})\, h_T\, cos(\theta)$ and the total inertial and geometric prameters are

$$m_T = m_b + m_r, \quad b_T = \frac{b_b\, m_b + b_r\, m_r}{m_T}, \quad h_T = \frac{h_b\, m_b + h_r\, m_r}{m_T}." \tag{4.18}$$

The longitudinal drag force is modelled using the formula $F_d = \frac{1}{2}\, \rho\, C_d\, A\, v_x^2$ where ρ is the air density, C_d is the drag coefficient, and A is frontal section area [95, p. 92].

4.1.3 Model enhancement

To enhance the model accuracy, the gyroscopic torques generated by the motion of the wheels [95] have been added to the yaw and roll equations in the damping matrix as

$$\bar{C} = \tilde{C} + \left[0,\, 0,\, C_{gyro}^{yaw},\, C_{gyro}^{roll}\right]^\top, \tag{4.19}$$

where

$$\begin{aligned} C_{gyro}^{yaw} &= (I_{wf}\omega_{wf} + I_{wr}\omega_{wr})\dot{\theta}, \\ C_{gyro}^{roll} &= -(I_{wf}\omega_{wf} + I_{wr}\omega_{wr})\dot{\psi}c_\theta, \end{aligned} \tag{4.20}$$

and $\omega_{wi} = \frac{v_x}{r_{wi}}$, $i \in \{f, r\}$ are the front and rear wheels rotational velocities, I_{wi}, $i \in \{f, r\}$ are the front and rear wheels rotational inertias.

The final dynamics is $\tilde{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}' + \bar{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}') + \tilde{G}(\boldsymbol{q}) = \bar{B}\boldsymbol{w}$ and right-hand side equations are then

$$\ddot{\boldsymbol{q}}' = \tilde{M}^{-1}\left(\bar{B}\boldsymbol{w} - \bar{C} - \tilde{G}\right). \tag{4.21}$$

4.1.4 Spatial Reformulation

The system dynamics has been reformulated in the Frenet frame, i.e. using spatial coordinates w.r.t. s, the arc length along the track [96, 97], in order to eliminate the dependency on the velocity in the position reference given to the MPC. The reference trajectory σ is then parameterized on s and the *curvature* $\zeta = \frac{1}{\rho}$, where ρ is the instantaneous curvature radius, is computed as

$$\zeta = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}}, \tag{4.22}$$

where $\dot{f} = \frac{df(s)}{ds}$ and $\ddot{f} = \frac{d^2 f(s)}{ds^2}$.

The *tracking errors* are geometrically derived as

$$e_\psi = \psi - \psi_s, \;\; e_y = c_{\psi_s}(Y_s - Y) + s_{\psi_s}(X_s - X), \tag{4.23}$$

where ψ_s is the reference yaw angle, (X_s, Y_s) is the absolute position reference at the current spatial coordinate and (X, Y) is the current absolute position.

The tracking errors dynamics are derived as [98]

$$\dot{e}_\psi = \dot{\psi} - \zeta\, \dot{s}, \;\; \dot{e}_y = v_x s_{e_\psi} + v_y c_{e_\psi}. \tag{4.24}$$

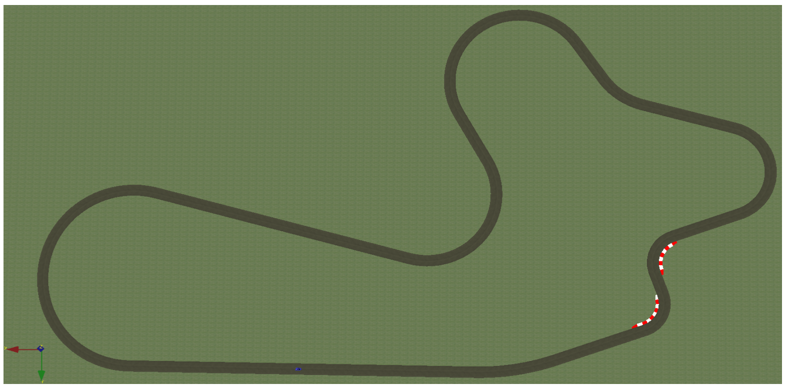


Figure 4.4. Simulation track. The track is available with a standard installation of `VI-BRT` and allows to evaluate performance on a high velocity testcase.

Moreover, this allows to differentiate a generic time dependent function $\Psi(t)$ w.r.t the curvilinear abscissa s using the *chain rule* as

$$\Psi' = \frac{d\Psi}{ds} = \frac{d\Psi}{dt}\frac{dt}{ds} = \frac{d\Psi}{dt}\frac{1}{\dot{s}} = \frac{\dot{\Psi}}{\dot{s}}, \tag{4.25}$$

where $\dot{s} = \frac{1}{1-\zeta\, e_y}\left(v_x\, c_{e_\psi} - v_y\, s_{e_\psi}\right) \neq 0\, \forall t$.

4.1.5 Complete Model

The complete state vector is

$$\boldsymbol{\xi} = [e_\psi,\, e_y,\, \theta,\, v_x,\, v_y,\, \dot{\psi},\, \dot{\theta},\, \delta,\, \gamma_t,\, \gamma_b,\, y_r]^\top, \tag{4.26}$$

where these states are assumed to be fully measured or computed by measurable quantities without noise, and the input vector is

$$\boldsymbol{u} = [\dot{\delta},\, \dot{\gamma}_t,\, \dot{\gamma}_b, \dot{y}_r]^\top, \tag{4.27}$$

i.e. the derivatives of the actual input to the vehicle. This formulation allows to have a smoother action of the controller and to avoid too aggressive non-realistic behaviours.

The dynamics equation of the model used in the NMPC algorithm can be compactly written as

$$\boldsymbol{\xi}' = \boldsymbol{f}(\boldsymbol{\xi}(s), \boldsymbol{u}(s); \sigma(s)). \tag{4.28}$$

4.2 Results

The controller has been tested on two different scenarios in order to evaluate its capability:

- a chicane maneuver with a double curve of radius $r = 40m$ (Fig. 4.5) to be performed with a limit constant velocity, used to assess flexibility of the controller and handling capabilities;

- a race track scenario (Fig. 4.4), available with a standard installation of `VI-BRT`, that allows to highlight the effectiveness of the rider movement in travelling curves at high velocities.

4.2.1 Cosimulation environment

The cosimulation relies on `VI-BikeRealTime` (`VI-BRT`), a high fidelity simulation software specifically designed to reproduce in real time motorcycle behaviour for high performance driving [89]. Its simulation model has 11 degree of freedom (DoF), 6 for the sprung mass, 2 for each wheel-suspension and 1 for the rotation between pinion and chassis. It includes comprehensive dynamics of tires, suspensions, brakes, driveline and gyroscopic effects. Moreover, a moving mass with lateral DoF simulates the rider.

The cosimulation is performed in `Simulink`, by connecting a `VI-BRT` simulation block with the VR. In particular, the controls computed by the `MATMPC`-based controller are integrated and fed to the simulation block. `VI-BRT` is used to simulate at $f_{sim} = 1000Hz$ the dynamics of the vehicle while the control action is updated by `MATMPC` at $f_c = 100Hz$, i.e. updating the control every $10ms$.

The simulations have been made on a WINDOWS 10 PC, with Intel(R) Core(TM) i7-7700 CPU running at 3.60GHz.

The motorcycle used for the simulation is a vehicle available with a standard installation of `VI-BRT`.

4.2.2 Controller setup

The NMPC controller relies on the NLP definition in Eq. (2.3). To effectively track the reference trajectory, the cost function for the virtual rider is defined as

$$\begin{aligned} \boldsymbol{h}_k(\boldsymbol{\xi}_k, \boldsymbol{u}_k) &= [e_\psi + \alpha, e_y, \dot{e}_\psi, \dot{e}_y, e_v, \alpha, y_r, \gamma_t \cdot \gamma_b, \dot{\delta}, \dot{\gamma}_t, \dot{\gamma}_b, \dot{y}_r]^\top, \\ \boldsymbol{h}_N(\boldsymbol{\xi}_k) &= [e_\psi + \alpha, e_y, \dot{e}_\psi, \dot{e}_y, e_v, \alpha, y_r, \gamma_t \cdot \gamma_b]^\top \end{aligned} \tag{4.29}$$

where $e_v = v - v_{ref}$, $v = \sqrt{v_x^2 + v_y^2}$, is the vehicle velocity error and $\alpha = \text{atan}(\frac{v_y + b\dot{\psi}}{v_x})$ is the motorcycle side slip angle. The terms related to errors e_ψ, e_y, e_v are needed to make the controller follow a path, while those related to the derivatives of errors have been introduced to have smoother actions for tracking. The penalty on the sideslip α is used to make the controller choose a restrained behaviour of the motorcycle. Note that the heading error e_ψ does not penalize the sideslip, while a dedicated term has been used to have more flexibility in the cost function tuning. The penalty on y_r is used to make the rider return to vertical position and those on $\gamma_t \cdot \gamma_b$ are used to avoid contemporary throttling and braking. Finally, the terms on the inputs ensure a smooth control action.

The constraints functions are defined as

$$\begin{aligned} \boldsymbol{r}_k(\boldsymbol{\xi}_k, \boldsymbol{u}_k) &= [\delta, \gamma_t, \gamma_b, y_r, \dot{\delta}, \dot{\gamma}_t, \dot{\gamma}_b, \dot{y}_r]^\top, \\ \boldsymbol{r}_N(\boldsymbol{\xi}_k) &= [\delta, \gamma_t, \gamma_b, y_r]^\top, \end{aligned} \tag{4.30}$$

where the constraints on $\delta, \gamma_t, \gamma_b$ and y_r are intrinsic bounds of the motorcycle controls,

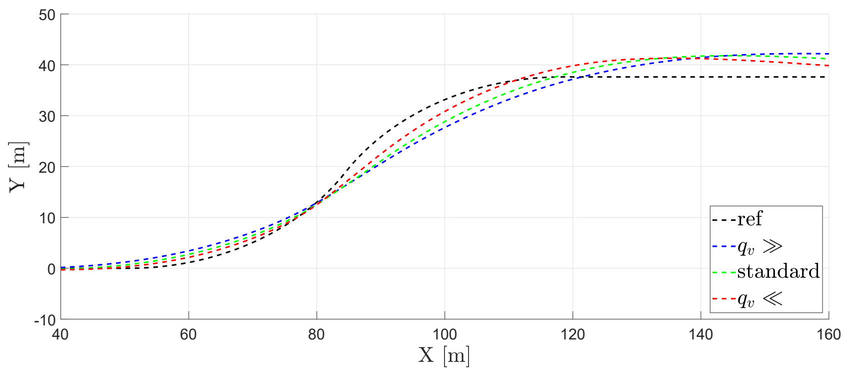


Figure 4.5. Reference path and trajectory of the motorcycle on the chicane. The different behaviours in terms of tracking performance are shown: the higher is the weight on velocity error q_v, the more the trajectory cuts the corners.

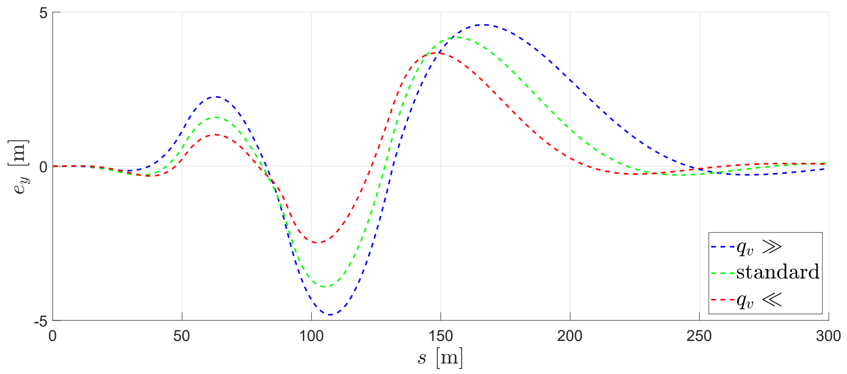


Figure 4.6. Trajectory lateral error along the chicane using different weight configurations: the higher is the weight on the velocity error q_v, the higher is the tracking error.

while those on $\dot{\delta}, \dot{\gamma}_t, \dot{\gamma}_b$ and $\dot{y}_r$ are added in order to improve the smoothness of the inputs. The bounds have been set as

$$\begin{aligned} \underline{\boldsymbol{r}}_k &= [-\frac{\pi}{20}, 0, 0, -0.15, -1, -10, -10, -1]^\top, \\ \overline{\boldsymbol{r}}_k &= [+\frac{\pi}{20}, 1, 1, +0.15, +1, +10, +10, +1]^\top, \\ \underline{\boldsymbol{r}}_N &= [-\frac{\pi}{20}, 0, 0, -0.15]^\top, \\ \overline{\boldsymbol{r}}_N &= [+\frac{\pi}{20}, 1, 1, +0.15]^\top, \end{aligned} \tag{4.31}$$

where the bounds on δ, γ_t, and γ_b are the limits of the vehicle controls and the one on y_r is an estimate of the maximum lateral displacement of the rider's CM from the vertical axis of the motorcycle. The bounds on the control derivatives are set as to have a smooth action of the controller.

The integrator used in `MATMPC` for the simulations is Explicit Runge Kutta 4 and the QP Solver is `HPIPM` [99]. For the integrator, two integration steps per shooting interval of length $T_s = 2$ meter are used. A total number of $N = 50$ shooting intervals are used, enabling a prediction length of 100 meters on track.

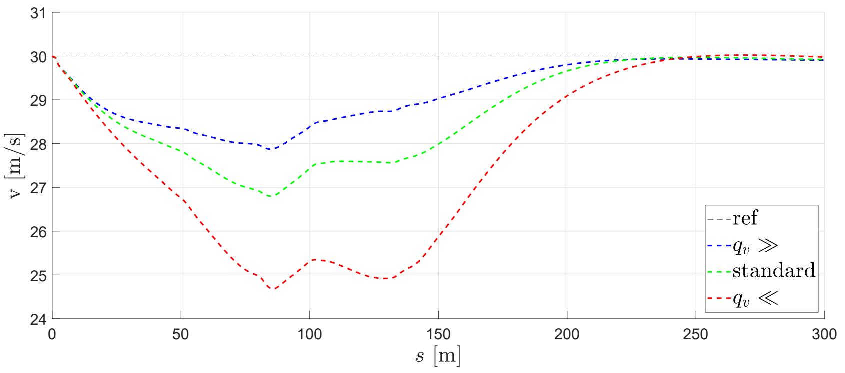


Figure 4.7. Velocity of the motorcycle along the chicane using different weight configurations: the higher is the weight on the velocity error q_v, the lower is the velocity error.

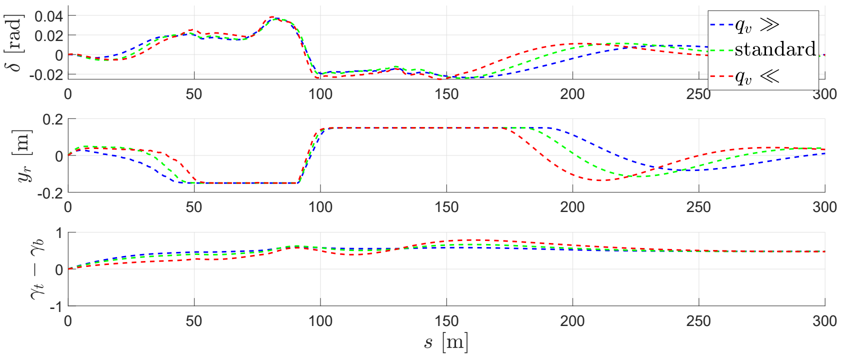


Figure 4.8. Steering angle, rider position and throttle/braking fed to the motorcycle. The configuration with highest weight on the velocity error anticipates the action for both steering angle and rider movement in order to cut the corners.

4.2.3 Simulation Results on Chicane

The chicane maneuver requires roll angles over $45deg$ to be accomplished at the velocity reference of $30m/s$. The comparison of the controller behaviour has been made using different weights for velocity and trajectory tracking to demonstrate the flexibility of the controller and the possibility to choose different behaviours. The adopted *standard* configuration of weights is

$$\begin{aligned}W &= \text{diag}([q_{e_\psi}, q_{e_y}, q_{\dot{e}_\psi}, q_{\dot{e}_y}, q_{e_v}, q_\alpha, q_{y_r}, q_\gamma, q_{\dot{\delta}}, q_{\dot{\gamma}_t}, q_{\dot{\gamma}_b}, q_{\dot{y}_r}]) \\ &= \text{diag}([5 \cdot 10^{-1}, 10^{-2}, 10^{0}, 10^{-4}, 10^{-2}, 2 \cdot 10^{2}, 10^{-3}, \\ &\qquad 10^{5}, 2 \cdot 10^{0}, 2 \cdot 10^{-1}, 5 \cdot 10^{-1}, 5 \cdot 10^{-2}]).\end{aligned}$$

As is common in MPC implementations, a more conservative set of weights is used for the terminal cost [54] to ensure a stable dynamical behavior. An empirical analysis has been conducted, resulting in improved stability of the closed-loop system when the terminal weights on the heading error and its velocity are increased. The terminal weights are then set as follows

$$\begin{aligned}W_N &= \text{diag}([q^N_{e_\psi}, q^N_{e_y}, q^N_{\dot{e}_\psi}, q^N_{\dot{e}_y}, q^N_{e_v}, q^N_\alpha, q^N_{y_r}, q^N_\gamma]) \\ &= \text{diag}([10^{2}, 10^{-2}, 10^{1}, 10^{-4}, 10^{-2}, 2 \cdot 10^{2}, 10^{-3}, 10^{5}]).\end{aligned}$$

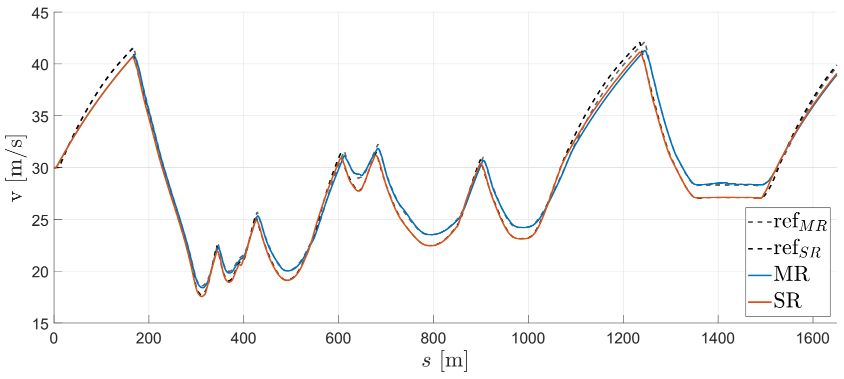


Figure 4.9. Actual and reference velocity along the track. The reference profiles have been computed as to have the maximum performance in each case. The MR allows higher cornering velocities in the same boundary conditions.

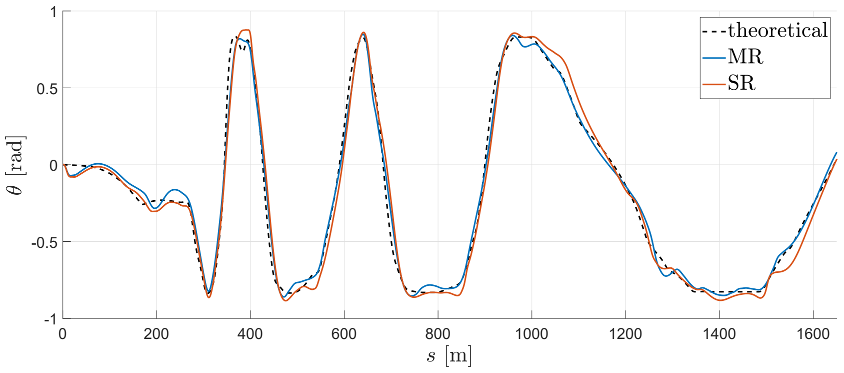


Figure 4.10. Roll angle of the motorcycle along the track. The actual roll is close to the theoretical one (computed for SR case, since the expression does not apply to the MR case), and reaches spikes around $50deg$.

The weights on the velocity and lateral errors are modified in order to define the "$q_v \gg$" configuration (larger weight on velocity, smaller weight on lateral error), i.e.

$$q_{e_v} = 5 \cdot 10^{-2}, \; q_{e_y} = 10^{-2},$$

and "$q_v \ll$" configuration (smaller weight on velocity, larger weight on lateral error), i.e.

$$q_{e_v} = 5 \cdot 10^{-3}, \; q_{e_y} = 5 \cdot 10^{-2}.$$

The different behaviours in terms of trajectory tracking of the controller can be seen in Figg. 4.5 and 4.6. In particular, Fig. 4.6 shows the value of the lateral error with respect to the reference trajectory. The results have to be compared to Fig. 4.7, where the velocity profiles are shown. As expected, using the weight configuration "$q_v \gg$" the motorcycle heavily cuts the corner to maintain a higher velocity, while using the "$q_v \ll$" configuration significantly decreases velocity and lateral error.

Finally, the computed controls for the motorcycle are shown in Fig. 4.8. The rider movement, in accordance with the steering angle, is anticipated by using "$q_v \gg$" configuration before entering the first curve and postponed at the end of the second curve in order to cut the corners and maintain velocity.

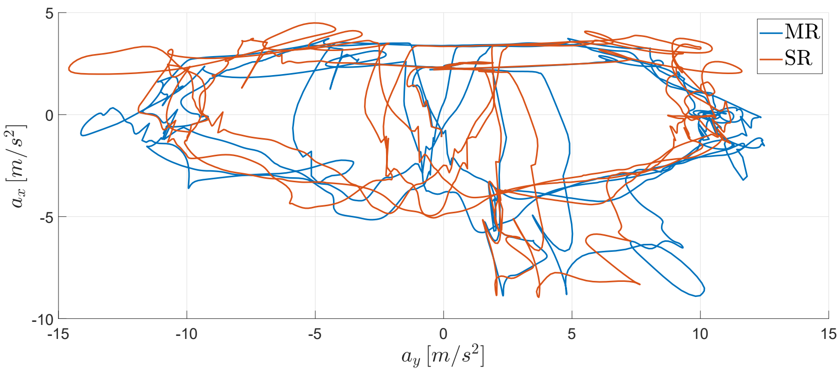


Figure 4.11. Longitudinal on lateral acceleration of the motorcycle along the track: the maximum steady-state turning lateral acceleration is around $10m/s^2$, hence the limits of the motorcycle are actually reached.

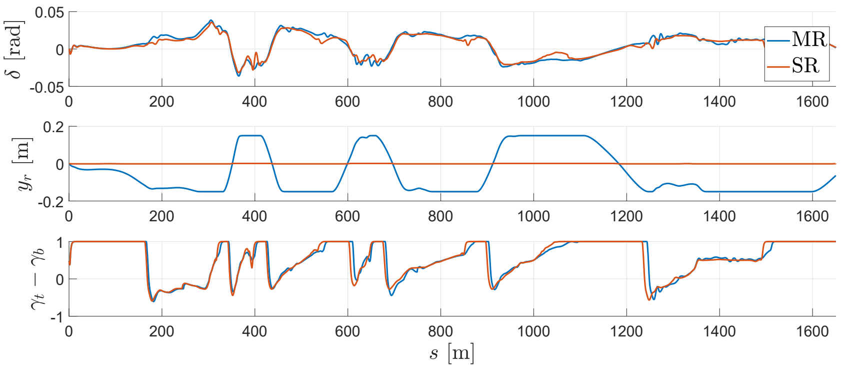


Figure 4.12. Controls fed to the simulator along the track. Note that the lateral displacement of the rider is in accordance to the steering angle and the roll angle of the motorcycle, hence actually supporting the maneuver.

4.2.4 Simulation Results on Track

The simulation on track has been done with both moving (MR) and stationary rider (SR). In order to compare the two cases with the exact same configuration of weights, the following set of values that gives good performance for both MR and SR has been chosen:

$$\begin{aligned} W = \text{diag}([&5 \cdot 10^{-1}, 5 \cdot 10^{-2}, 10^0, 10^{-4}, 3 \cdot 10^{-1}, 4 \cdot 10^2, 10^{-5}, 10^5, \\ &5 \cdot 10^{-1}, 1 \cdot 10^{-4}, 8 \cdot 10^{-4}, 6 \cdot 10^{-1}]), \\ W_N = \text{diag}([&5 \cdot 10^{-1}, 5 \cdot 10^{-2}, 10^0, 10^{-4}, 3 \cdot 10^{-1}, 4 \cdot 10^2, 10^{-5}, 10^5]). \end{aligned}$$

To prove the effectiveness of the controller in handling aggressive maneuvers, the velocity reference is generated to specifically maximize the sustained lateral acceleration during steady-state turning. The velocity profile is computed by means of a commercial static lap time planner based on tire characteristics, engine and brake power, and motorcycle geometrical properties. Since the planner only approximates the dynamics behaviour, the resulting velocity profile is usually conservative. To make the maneuver

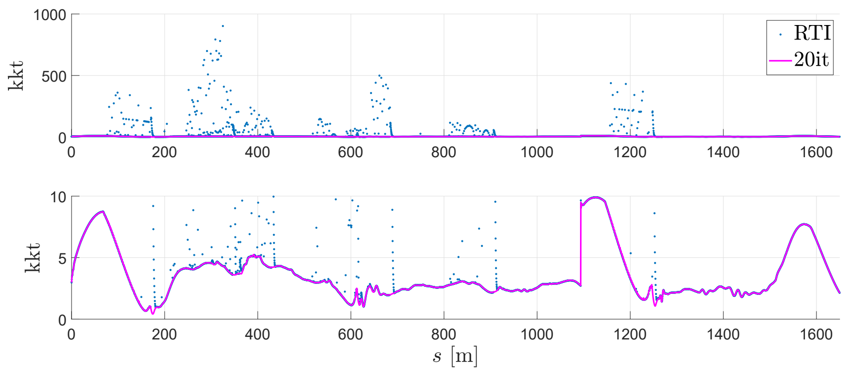


Figure 4.13. KKT values along the simulation using RTI scheme and 20 iterations for solving SQP. The zoomed plot (bottom) allows to recognize the similarity in the KKT trend along the path for the different solutions.

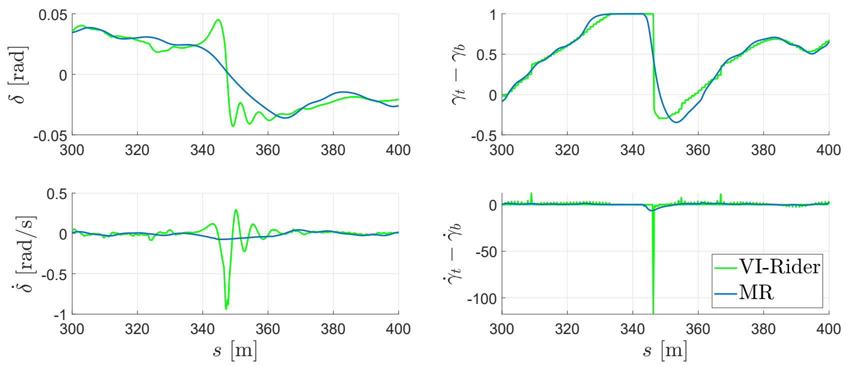


Figure 4.14. Comparison with *VI-Rider*. Detail of the steering angle and longitudinal command. A smoother behaviour can be noticed, particularly clear in the commands derivatives.

more aggressive, i.e. increase the velocity profile, the tire grip has been used as scaling factor and progressively augmented. The actual velocity profiles in simulation are shown in Fig. 4.9. The motorcycle with MR control is able to travel all the curves at least $1m/s$ faster then the SR case, while maintaining similar (or smaller) values of roll angle.

The roll angle and the theoretical roll angle are shown in Fig. 4.10. The controller autonomously follow a roll trajectory similar to the theoretical one, computed as $\theta_{th} = atan\left((v_{ref}^2 \cdot \zeta)/g\right)$, where v_{ref} is the SR velocity reference.

In Fig. 4.11, the lateral-longitudinal accelerations plot is shown: the steady-state turning lateral acceleration in both cases is around $10m/s^2$ ($1g$ of lateral acceleration), hence the limits of the motorcycle are reached.

The computed controls for the motorcycle are shown in Fig. 4.12. The rider movement is in accordance to the steering angle and the roll angle of the motorcycle, actually contributing to the improvement of performance.

A useful index provided by the optimization is the KKT value, i.e., the norm of the gradient of the Lagrangian function associated to the NLP problem [100], that is 0 in the optimal solution. The KKT values for the problem at hand are shown in Fig. 4.13, compared with the ones obtained by using 20 iterations for solving the SQP, to achieve a more accurate solution. As expected, with 20 SQP iterations the KKT values are smaller and more regular w.r.t. the less accurate RTI case. The isolated relatively large KKT

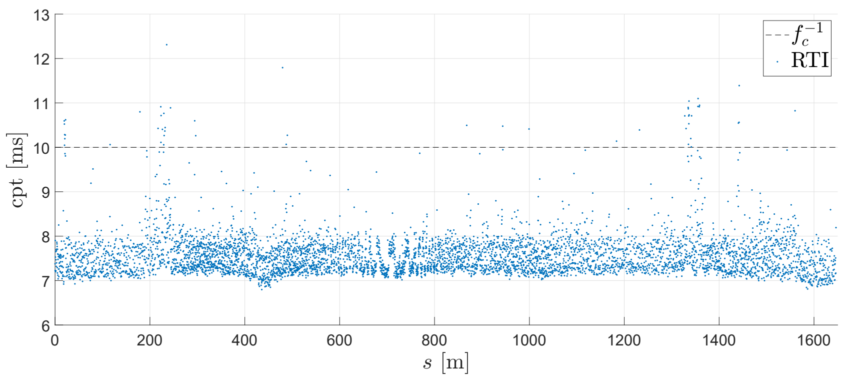


Figure 4.15. Computational times along the track for RTI and control period. The mean time is 7.56ms and the maximum is $T_{solver}^{max} = 12.31ms$.

	δ_f	y_r	$\gamma_t - \gamma_b$
Mean relative difference	0.17%	0.10%	0.16%
Max relative difference	2.93%	0.76%	3.63%

Table 4.2. Relative difference $d = |(x - y)/x| \cdot 100$ for the commands using RTI or SQP.

values of RTI scheme visible in the plot are essentially due to the absence of a line search strategy, which is used instead while solving the problem with SQP. However, the closed loop performance of the RTI and SQP controllers are nearly identical (see Table 4.2), proving the effectiveness of the RTI scheme in this scenario.

Moreover, a comparison with the commercial virtual rider distributed with `VI-BRT`, namely `VI-Rider`, has been carried out on the same scenario. `VI-Rider` is based on the work presented in [10, 101] and references therein. The velocity tracking performance are similar, but smoother dynamics can be observed both on the steering angle and the longitudinal command $\gamma_t - \gamma_b$ (see Fig. 4.14). This characteristic better matches a human-like behaviour, hence making the virtual prototyping tool more effective.

Finally, the computational times at each iteration for the controller in the MR case are shown in Fig 4.15. The mean value is $T_{solver}^{mean} = 7.56ms$ and the maximum one is $T_{solver}^{max} = 12.31ms$. Values exceeding $10ms$ are mainly due to the Windows OS, that does not allow to give priority to a single process. In a RT environment, actual real-time performance would be achieved.

5

Data-driven Tuning of a Real-Time NMPC Virtual Rider

In this Chapter, the data-driven tuning based on a Genetic Algorithm (GA) of the NMPC-based Virtual Rider (VR) presented in Ch. 4 is described. The goal is to obtain the *most suitable* weight configuration to track a desired trajectory, both in terms of path and velocity profile. The considered optimization objectives of the GA, i.e. the overall performance and the sensitivity of the solution to variations of controller parameters, are characterized in Sec. 5.1, while the validation of the procedure and its effectiveness in a realistic VR scenario (whose simulation model is depicted in 5.1) is reported in Sec. 5.2.

Figure 5.1. Representation of the virtual motorcycle in the simulation framework [89].

5.1 Data-driven Tuning Strategy

The aim of the strategy is to find the weighting matrices W and W_N of the cost function within the optimal control problem, as illustrated in the previous chapter in Eq. (4.29). Each individual of the population in the NSGA-II algorithm, then, is composed by the values of those matrices (*weights*). The n-tuple of decision variables, i.e. the weights, is called *configuration*. For comparison purposes, each configuration is normalized inside each domain interval.

The single optimization problem of finding the best performing simulation is cast into a multi-objective one, introducing a sensitivity function as a second objective. For the problem at hand, in fact, numerical issues are frequent, and it is very valuable to know if a configuration is *robust* to small parameters variation. The multi-objective optimization problem is hence characterized by two objectives and a constraint. In particular, the two objectives are $\mathbf{J}_1$, i.e. the time to complete a lap of the track, and $\mathbf{J}_2$, i.e. the sensitivity of the performance with respect to the weights, and the constraint $\mathbf{C}$ represents the exceeding of the track limits. The GA is then expected to reveal a Pareto front composed by a set of configurations that explore the trade-off between the lap-time and the robustness with respect to weight modifications, while excluding configurations that overcome the track bounds.

Note that, in case the motorcycle falls down or the fast NMPC solver has numerical issues during the task, the lap will not be concluded. This fact can heavily impact on the tuning procedure results, hence in the following tailored strategies are defined to account for it. However, as the NMPC constraints in Eq. (4.30) are defined only on the control actions and their integration, the feasibility of the NLP is not a fundamental subject, hence it is not explicitly considered.

5.1.1 Performance function

To assess the performance of a configuration, the track lap-time has been considered. Moreover, in order to consider task failures, the following cost function has been defined

$$\mathbf{J}_1 = \begin{cases} t_{end} & \text{if simulation does conclude the lap} \\ t_{max} - t_{end} & \text{if simulation does \textit{not} conclude the lap} \end{cases} \tag{5.1}$$

where t_{end} is the simulation time (i.e. time passed within the simulation) and t_{max} is a user-defined value higher than any possible concluding lap-time. In this way, even if no simulation is concluding, the ones that run closer to the end are preferred over the others in the GA algorithm.

5.1.2 Sensitivity function

The sensitivity function $\mathbf{J}_2$ considers, for each tested weights configuration, the variation of the closed loop performance - in terms of $\mathbf{J}_1$ - with respect to the variation of the decision variables values, weighted with respect to the distance in the configuration space

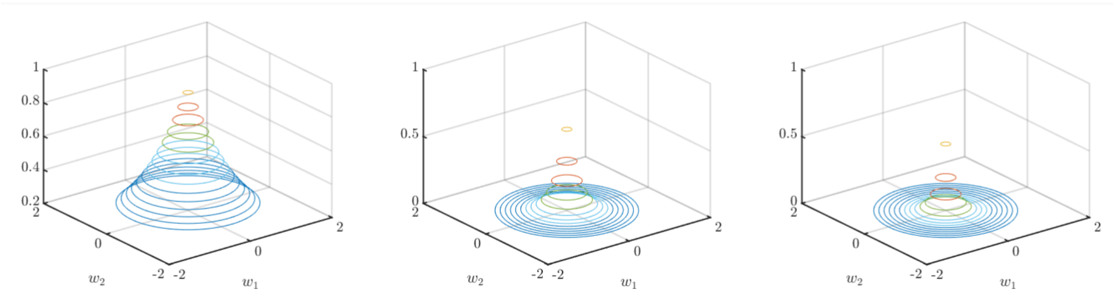


Figure 5.2. Values of w_i for configurations centered in $(0, 0)$ in a 2-dimension decision variable space, with (left) $\beta = 1$, (center) $\beta = 5$, (right) $\beta = 7$.

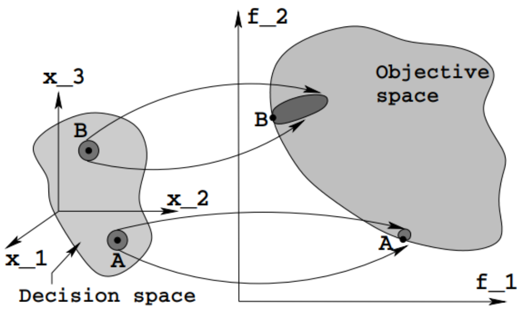


Figure 5.3. Point A is less sensitive to variable perturbation than point B in a multi-objective optimization scenario [75].

(as represented in Fig. 5.3).

To define the sensitivity, a comparison of the lap-time of every configuration $\mathbf{w}_j$ of a new generation with respect to all the previously run simulations (in all the generations) $\mathbf{w}_i, i \in \{1, \ldots, G \cdot M\}$, where G is the generation number and M is the number of individuals in each generation, is obtained through the following sensitivity function

$$S_j = \frac{\sum_i q_i \cdot C_{j,i}}{\sum_i q_i} \tag{5.2}$$

where $C_{j,i}$ defines the *performance similarity index* between i-th and j-th simulations and q_i weights the distance in the decision variable space as

$$q_i = \frac{1}{e^{\beta \cdot d_i}}, \tag{5.3}$$

where $d_i = \|\mathbf{w}_j - \mathbf{w}_i\| \in \left[0, \sqrt{n}\right]$ is the Euclidean distance between the $\mathbf{w}_j$ and $\mathbf{w}_i$, n is the configuration dimension, and β is the shape factor. In Fig. 5.2, q_i for a representative 2 dimensional scenario is shown. If both the simulation j and the compared one i finish the lap, the index $C_{j,i}$ is defined as

$$C_{j,i} = \begin{cases} -1 & \text{if } V_{j,i} \leq V_m \\ \left(\dfrac{V_{j,i} - V_m}{V_M - V_m}\right)(\alpha + 1) \; - 1 & \text{if } V_m \leq V_{j,i} \leq V_M \\ \alpha & \text{if } V_{j,i} \geq V_M \end{cases} \tag{5.4}$$

where $V_{j,i} = \dfrac{P_i - P_j}{P_j} \times 100$ is the percentual lap-time difference between i-th and j-th simulations, and P_j and P_i are the lap-time of i-th and j-th simulations, respectively. V_M and V_m are the maximum and minimum threshold, respectively, to consider a match or a mismatch with respect to the performance, while $\alpha \geq 1$ is a user-defined parameter that defines the importance of a mismatch in the neighborhood. Using this definition, the comparison results in a *perfect match* (i.e. $C_{j,i} = -1$) if the lap-time differs less then V_m%, increases linearly to α until V_M% difference, and it is a *mismatch* (i.e. $C_{j,i} = \alpha$) if the lap-time differs more then V_M%. If the simulation j finishes the lap but i does *not*, the index is defined as a mismatch, i.e. $C_{j,i} = \alpha$.

Besides, the sensitivity function for a simulation that does *not* end the lap lose its relevance. In this case, the sensitivity function is defined as to support the exploration of decision variable space where more simulations conclude the lap. To obtain this behavior, when the simulation j does *not* conclude the lap

$$C_{j,i} = \begin{cases} -1 & \text{if simulation } i \text{ does conclude the lap} \\ \alpha & \text{if simulation } i \text{ does } \textit{not} \text{ conclude the lap} \end{cases}.$$

In this way, better values (i.e., lower ones) are given to simulations that do not conclude the lap but, in their neighborhood, there are more configurations that conclude it.

Finally, the obtained sensitivity value, that lays in the interval $S \in [-1, \alpha]$, is shifted in the range [0, 100], and hence the actual objective function for each configuration j is given by

$$\mathbf{J}_2 = \left(\frac{100}{\alpha + 1}\right)(S + 1). \tag{5.5}$$

In particular, a configuration that presents 0% of sensitivity means that little variation on weights of the cost function produces the same closed-loop simulation results. On the contrary, 100% of sensitivity means that any slight weights change leads the simulation to end with totally different behavior.

5.1.3 Constraint

In NSGA-II, the constraints are meant as *soft constraints*, meaning that they can be exceeded but, in that case, the solution cannot be in the first Pareto front. In this way, as soon as one simulation satisfies the constraint, it will be preferred over the others and, by the end of the procedure, only the simulations that satisfy the constraint will remain. To do so, a function that measures *how much* the constraint is violated is recommended.

For this reason, a function that evaluates the percentage of simulation that is traveled outside the track bounds has been defined as

$$\mathbf{C} = \frac{c_{out}}{c_{tot}}, \tag{5.6}$$

where c_{out} is the part of track that has been traveled outside the bounds and c_{tot} is the total length of the track.

5.2 Results

In this section, the results obtained with the proposed tuning strategy are presented. First, the tuning strategy implementation is described, then an analysis of the tuning algorithm procedure is shown, a closed-loop performance comparison between the fastest and the most robust configuration is described, and specific evaluations on the sensitivity function are presented.

5.2.1 Tuning Strategy Implementation

The tuning strategy described in Sec. 2.3.1 and defined in Sec. 5.1 has been implemented for the task at hand. The different implementation aspects, i.e. parameters choice, initial population and algorithm usage are specified in the following.

The automatic tuning procedure presented in this manuscript is developed in MATLAB. The constrained multi-objective optimization problem, properly defined for tuning purposes, is solved by applying the NSGA-II algorithm and its implementation relies on the MATLAB-based toolbox proposed by Tamilselvi et al. in [102]. To run the algorithm, each configuration of weights is set in the NMPC-based virtual rider and a simulation of the track lap is performed as described in Sec. 4.2.1. At the end of each simulation, the lap time, the out-of-track percentage, and the sensitivity value are evaluated to determine the overall performance of each individual.

To speed up the procedure, a parallelization technique is taken into account providing the possibility of performing different co-simulation of the virtual rider simultaneously.

Finally, in order to further reduce the computational time of the procedure, only the weights that have proven to be highly significant in improving driving performance have been manipulated. In particular, these weights are the heading error weight q_{e_ψ}, the lateral error weight q_{e_y}, the velocity error weight q_{e_v} and the sideslip weight q_{e_α}. In addition, a multiplier of the reference velocity profile is considered. For each weight, a search range has been defined that varies by two orders of magnitude higher and lower with respect to the manually tuned configuration

$$\begin{aligned} \mathrm{W} &= \mathrm{diag}([q_{e_\psi}, q_{e_y}, q_{\dot{e}_\psi}, q_{\dot{e}_y}, q_{e_v}, q_\alpha, q_{y_r}, q_\gamma, q_{\dot{\delta}}, q_{\dot{\gamma}_t}, q_{\dot{\gamma}_b}, q_{\dot{y}_r}]) \\ &= \mathrm{diag}([5 \cdot 10^{-1}, 5 \cdot 10^{-2}, 10^{0}, 10^{-4}, 3 \cdot 10^{-1}, 4 \cdot 10^{2}, \\ &\qquad 10^{-5}, 10^{5}, 5 \cdot 10^{-1}, 1 \cdot 10^{-4}, 8 \cdot 10^{-4}, 6 \cdot 10^{-1}]) \end{aligned} \tag{5.7}$$

presented in Ch. 4. Regarding the velocity multiplier, its searching interval is defined between 1 and 1.1.

Hyper-parameters Tuning

Concerning the NSGA-II algorithm parameters, the choice of the population size (M) and the number of generations (G_M) control the trade-off between the convergence to a significant final Pareto front and the computational burden of the algorithm. Satisfactory performance have been obtained setting $M = 50$ and $G_M = 50$. To reduce the risk of local minima the distribution index for crossover and mutation have been both set equal

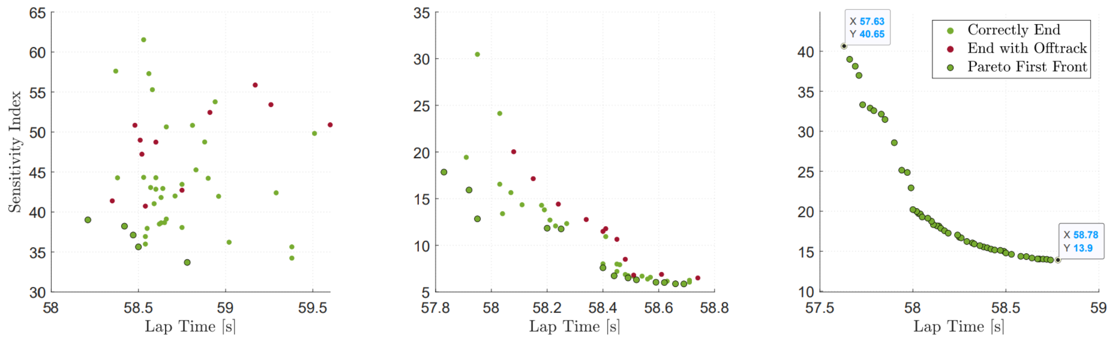


Figure 5.4. Evolution of the Pareto front at (left) generation 5, (center) generation 10, (right) generation 50.

to 1 and the mutation probability to 0.9, according to [102].

Regarding the sensitivity function, i.e. Eq. (5.2), the values of the hyperparameters α and β influence the sensitivity computation. The parameter α defines the importance of a performance mismatch (or a failing simulation) with respect to a performance match in the neighborhood, and it has been set to $\alpha = 1.2$ in order to further penalize the mismatch in the context of sensitivity. Setting $\beta = 5$ resulted in a suitable weighting of the contribution of neighboring configurations, and hence an appropriate neighborhood dimension. Indeed, too high values of β would lead to a too small neighborhood, hence to ignore the adjacent results, while too low values would yield to assign the same importance to all the simulation results. Finally, $V_m = 1\%$ and $V_M = 5\%$ resulted suitable values for equivalence classes of the performance.

Concerning the initial population, a correct distribution in the search space help the algorithm to converge rapidly. In this work, each decision variable presents a choice interval that spans several orders of magnitude. For this reason, a randomized choice is made within each magnitude order of each variable. In particular, let $\tilde{\mathbf{w}}_u$ and $\tilde{\mathbf{w}}_l$ be the upper and lower bounds of a generic component $\tilde{\mathbf{w}}$ of the configuration $\mathbf{w}$, we first consider the magnitude order applying

$$l_U = \log_{10}(\tilde{\mathbf{w}}_u), \quad l_l = \log_{10}(\tilde{\mathbf{w}}_l). \tag{5.8}$$

Then, for each decision variable, we define M random values of its magnitude order through

$$r = (l_U - l_l) \cdot rand(M) + l_l, \tag{5.9}$$

where the function $rand(M) \in [0, 1]$ returns M normalized random values drawn from a uniform distribution. To obtain the actual initial value of each component, the exponential is used as $\tilde{\mathbf{w}}_0 = 10^r$. Finally, condensing and reshaping the M values for the n components, M configurations are defined. The result is an initial population with M weights configurations randomly distributed in the search space with respect to the magnitude order.

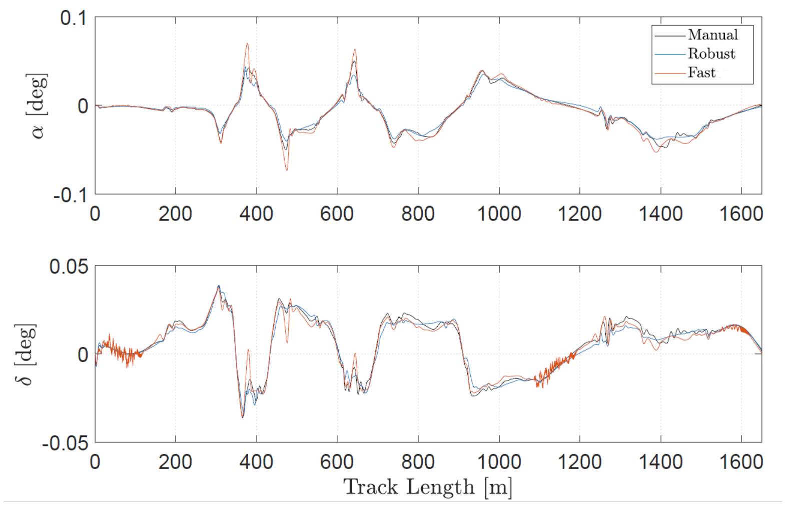


Figure 5.5. Comparison between manually tuned (black), most robust (blue) and fastest (red) configurations. (Top) sideslip angle, (bottom) steering angle command.

5.2.2 Tuning Algorithm Analysis

The tuning procedure described has been applied to the NMPC virtual rider. At the 1st generation, only 2 of the 50 configurations of weights, randomly chosen in the search intervals, complete the lap. Moreover, with both configurations, the rider overcomes the track limits. As depicted in Fig. 5.4, at the 5th generation all the configurations finish the lap even if some still exceed the limits of the track. At the 10th generation, all the simulations correctly complete the lap and the sensitivity index start to decrease and a Pareto front is emerging. After 50 generations, all the configurations belong to the first Pareto front, and, as expected, the conflict between robustness and lap-time performance in the problem at hand can be clearly seen.

5.2.3 Closed-loop Performance Comparison

To better analyze the meaning of the sensitivity index, a comparison of the two configurations at the extremes of Pareto - the faster and more sensitive and the more robust and slower - has been made. In particular, the dynamics of the bike-rider system during a lap simulation has been studied, specifically considering the steering angle signal and the side-slip angle during the simulations (Fig. 5.5). The fastest configuration presents an aggressive and unstable riding behavior, clearly shown by the ripple on the steering angle and the peaks on the side-slip angle. On the contrary, the robust configuration leads to a smoother and more stable behavior of the system.

Notably, the fastest simulation ends the lap in 57.63, which is 0.91 seconds less than the manually tuned configuration presented in Ch. 4 [103].

5.2.4 Sensitivity Analysis

An explicit sensitivity analysis has been conducted by evaluating the simulation results of new configurations, generated around both the most robust and the less robust

Table 5.1. Simulation results for 100 configurations chosen in a 5-dimensional sphere around the robust and fast configurations.

	Around Robust	Around Fast
Simulations that correctly ends	98	36
Simulations that overcome limits	1	27
Simulations that fails	1	37

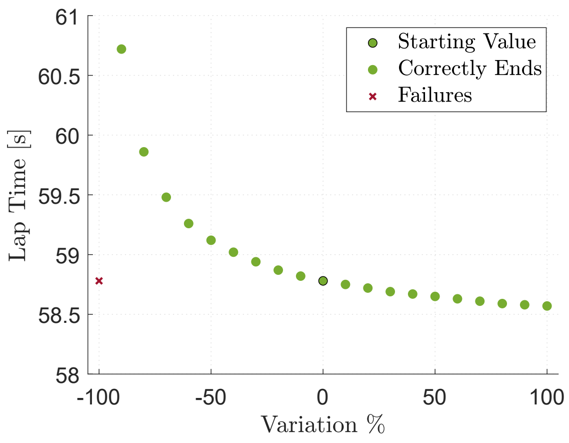


Figure 5.6. Effect of velocity error weight q_{e_v} modifications in terms of lap-time.

ones. Specifically, the two new sets of configurations were generated by considering 100 random configurations normally distributed in a 5-dimensional sphere around 30% of each value of the 5 weights of both configurations. The simulations results, reported in Tab. 5.1, illustrate that the almost every configuration generated around the robust one correctly end the lap inside the track limits. On the other hand, with the more sensitive configuration, only the 36% of the configurations correctly ends the lap and the other ones fail or overcome the track limits, confirming the sensitivity with respect to weights changes.

Additionally, an evaluation of the effect on the performance of varying a single weight has been accomplished. In particular, starting from the robust configuration, the weight on the velocity error q_{e_v} was modified up to $\pm 100\%$ of its value while all the others were kept fixed. As expected, the robustness of this configuration allows to highly modify the weight and obtain consistent simulation results, as shown in Fig. 5.6. This is an important feature for possible manual post-tuning by the user. Note that, on the contrary, most of the configurations around the non-robust one would fail (as revealed by the previous analysis).

6

A LbNMPC for a Virtual Motorcycle in Real-Time

In this Chapter, the definition of a Learning-based NMPC controller for a virtual motorcycle (depicted in 6.1) is described. The goal is to enhance the tracking performance of the controller on a desired trajectory, both in terms of path and velocity profile. Relying on the model described in Ch. 4, a grey-box prediction model is adopted, characterizing the residual acceleration error by Gaussian Processes, as detailed in Sec. 6.1. The results show the proficiency of the proposed approach in remarkably enhancing the tracking performances of the controller while maintaining real-time capabilities, as reported in Sec. 6.2.

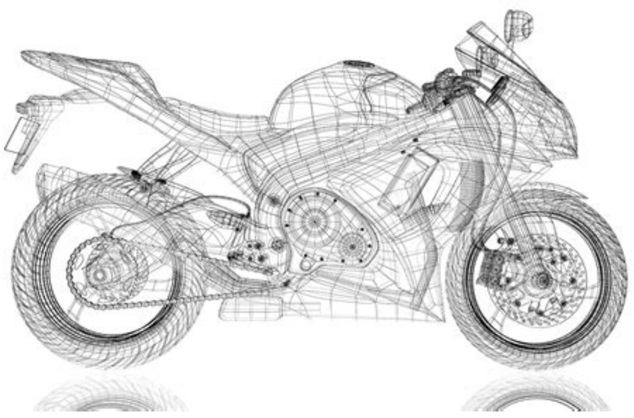


Figure 6.1. Representation of the virtual motorcycle in the simulation framework [89].

6.1 Model for Control Synthesis

The employed model is based on a nominal description of the system obtained by Lagrangian formulation, as described in Ch. 4, and integrated by Gaussian Processes to compensate for the residual acceleration errors. In the following, the main characteristics of the nominal model are reported for completeness, and the characterization of the GP-based modeling and the feature selection strategy are described, showing the obtained acceleration estimates.

The aim is to define the grey-box dynamics equation for the accelerations $\hat{\ddot{\boldsymbol{q}}}$ as described in (2.21) in Ch. 2, i.e.

$$\hat{\ddot{\boldsymbol{q}}} = \tilde{\ddot{\boldsymbol{q}}} + \bar{\ddot{\boldsymbol{q}}}, \tag{6.1}$$

where $\tilde{\ddot{\boldsymbol{q}}}$ is the nominal characterization, and $\bar{\ddot{\boldsymbol{q}}}$ is the GP-based acceleration errors model.

The nominal non-linear dynamics model is based on a sliding plane representation [103], as described in Ch. 4.1, and reported here for completeness. Its derivation relies on the Lagrangian framework, where the plane is allowed to roll and slide in both x and y directions. In addition, a characterization of the rider body is obtained by attaching a moving point mass with lateral DoF to the plane. Tire and drag forces are the input to the dynamics, where longitudinal tire forces are linear in throttle and brake commands, and the lateral tire forces are computed through the *Pacejka Magic Formula* for motorcycles. The gyroscopic effects are considered in the formulation, as well as the effect of caster and roll on the steering angle on the ground. The system dynamics equation is in the form $M(\boldsymbol{q})\tilde{\ddot{\boldsymbol{q}}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}}) + G(\boldsymbol{q}) = B\boldsymbol{w}$, where $\boldsymbol{w}$ are the tire and drag forces, and the resulting right-hand side equation is in the form

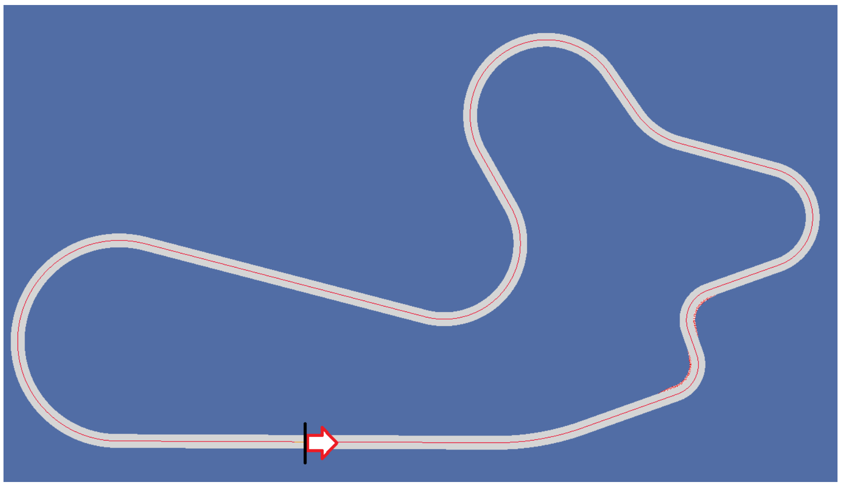
$$\tilde{\ddot{\boldsymbol{q}}} = M^{-1}\left(B\boldsymbol{w} - C - G\right), \tag{6.2}$$

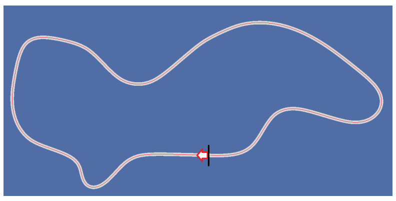
where $\boldsymbol{q} = [x, y, \varphi, \theta]^\top$, i.e. the coordinates (x, y) of the rear wheel contact patch position, the yaw angle φ, and the roll angle θ. Overall, the nominal model comprehends most of the important out-of-plane dynamics components for the motorcycle, but still lacks the vertical and pitch movements given by the suspensions as well as the tire combined behaviour.

6.1.1 Grey-box Dynamics

Given the nominal dynamics, the acceleration error characterization has been obtained through GPR, as described in Ch. 3. To this aim, a dataset was retrieved riding the motorcycle using the nominal NMPC controller (i.e. the NMPC controller based on the nominal dynamics) along different test tracks (shown in Fig. 6.2) in the simulation framework `VI-BRT`. Data collection was performed at $100Hz$, gathering the *actual*[1] accelerations, the nominal estimates, and the complete feature set at each time-step. This procedure led to a dataset of 51180 points, to be used for both training and testing of grey-box regression models. The obtained dataset is composed by the complete features

[1]quantities computed by the physical engine within the `VI-BRT` simulation will be referred to as *actual*.


(a) *VI-Track*


(b) Additional training track #1

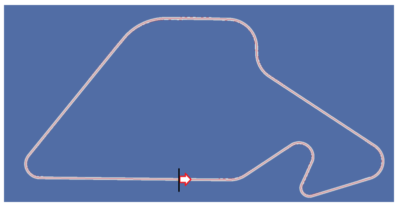

(c) Additional training track #2

Figure 6.2. Tracks used for data generation.

$\bar{\boldsymbol{x}}_k$ and the corresponding targets $y_k^i = \ddot{q}_k^i - \tilde{\ddot{q}}_k^i$, collected at the same time-step, that can be expressed as

$$\boldsymbol{z}_k^i = (\bar{\boldsymbol{x}}_k, y_k^i), \tag{6.3}$$

where $\boldsymbol{z}_k^i$ are the whole datapoints, comprising the complete features $\bar{\boldsymbol{x}}_k^{gp} \in \mathbb{R}^D$ and the targets $y_k^i \in \mathbb{R}$; index $k \in \{1, \dots, T\}$ refers to the time instant, $T = 51180$; index $i \in \{y, \varphi, \theta\}$ refers to the 3 different targets of interest. Note that the target accelerations are the direct lateral acceleration $\dot{v}_y$, the yaw acceleration $\ddot{\varphi}$ and the roll acceleration $\ddot{\theta}$, while the longitudinal one $\dot{v}_x$ has not been included since it obtained satisfactory results using just the nominal dynamics description. Thus, the GP-based estimate of the acceleration prediction error in (6.1) is defined as

$$\bar{\ddot{\boldsymbol{q}}} = [0, \, \bar{\ddot{q}}^y, \, \bar{\ddot{q}}^\varphi, \, \bar{\ddot{q}}^\theta]^\top. \tag{6.4}$$

To effectively apply GPR, the iterative incremental approach described in Sec. 3.2.3 has been employed for each component separately. Indeed, for each acceleration target, the subset of features that are most important to correctly predict the residual acceleration error $\ddot{q} - \tilde{\ddot{q}}$ has been obtained. The complete feature set considered in this work is

$$\bar{\boldsymbol{x}} = [\theta, v_x, v_y, \dot{\varphi}, \dot{\theta}, \delta, \gamma_t, \gamma_b, y_r, \dot{\delta}, \dot{\gamma}_t, \dot{\gamma}_b, \dot{y}_r, \alpha_r, \alpha_f, cos(\theta), sin(\theta)], \tag{6.5}$$

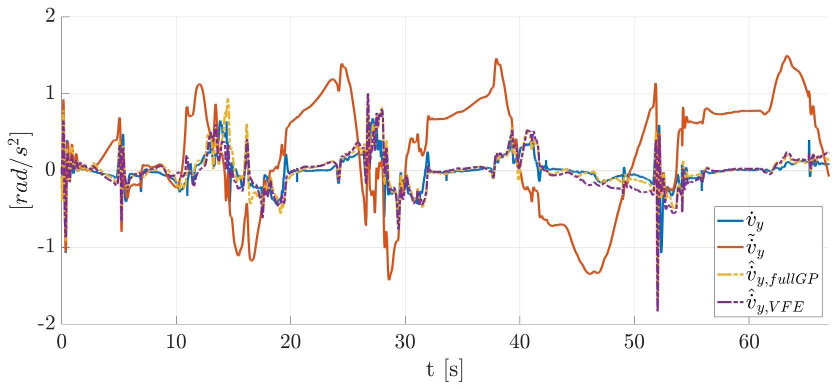
$\bar{\boldsymbol{x}} \in \mathbb{R}^{17}$, that includes (i) part of the system state, i.e. θ, v_x, v_y, $\dot{\varphi}$, $\dot{\theta}$, δ, γ_t, γ_b, y_r, (ii) the control inputs, i.e. $\dot{\delta}$, $\dot{\gamma}_t$, $\dot{\gamma}_b$, $\dot{y}_r$, and (iii) additional features, i.e. the sideslip angles α_r, α_f and composite functions $cos(\theta)$, $sin(\theta)$. In particular, the composite features

# of features	$\bar{\bar{q}}^y$	$\bar{\bar{q}}^\varphi$	$\bar{\bar{q}}^\theta$
2 Feat	[17,8] 0.3489	[17,7] 0.2352	[17,8] 0.4631
3 Feat	[17,8,7] 0.4799 (+38%)	[17,7,12] 0.3686 (+57%)	[17,8,5] 0.5324 (+15%)
4 Feat	[17,8,7,4] 0.5591 (+17%)	[17,7,12,8] 0.4265 (+16%)	[17,8,5,12] 0.5932 (+12%)
5 Feat	[17,8,7,4,6] 0.5749 (+3%)	[17,7,12,8,9] 0.4850 (+14%)	[17,8,5,12,4] 0.6588 (+11%)
6 Feat	/	[17,7,12,8,9,2] 0.5360 (+11%)	[17,8,5,12,4,2] 0.7301 (+11%)
7 Feat	/	[7,12,8,9,2,1] 0.5411 (+1%)	[17,8,5,12,4,2,15] 0.7985 (+7%)

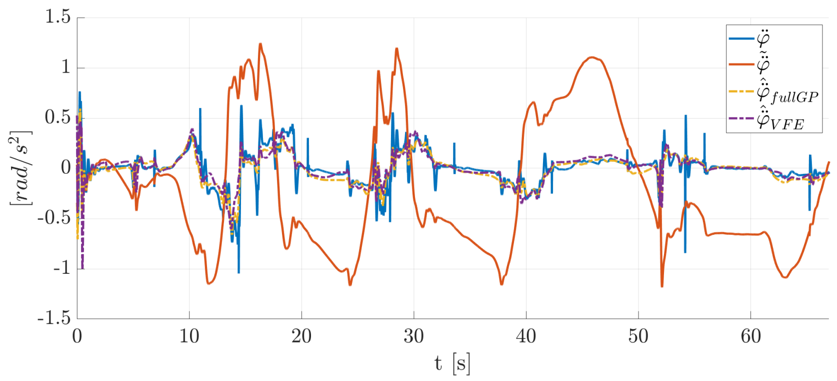
Table 6.1. R^2 index of different feature combinations for each grey-box acceleration target. The index has been evaluated on the whole dataset after training on a subset of 2000 points. The numbers represent the chosen features as ordered in Eq. (6.5).

have been used because they can be very expressive and correlate better to the desired target than the system state itself. See Sec. 6.1.2 for further details on each quantity. The results obtained by the algorithm in Lis. 3.1 on the grey-box modeling task are summarized in Tab. 6.1, showing, for each regression target, the features chosen by the algorithm at each iteration and the resulting R^2 index. As expected, the R^2 index tends to grow as new regression features are added. However, in order to maintain the GP modeling as light as possible, the configuration to be used in the test scenario was selected according to a parsimony principle. In particular, the chosen feature set is the one that, if a feature was added, it would lead to an increment of less then 10% of R^2. This resulted in 4 features for $\bar{\bar{q}}^y$, namely $\{sin(\theta), \gamma_b, \gamma_t, \dot{\varphi}\}$, and 6 features for both $\bar{\bar{q}}^\varphi$, namely $\{sin(\theta), \gamma_t, \dot{\gamma}_b, \gamma_b, y_r, v_x\}$, and $\bar{\bar{q}}^\theta$, namely $\{sin(\theta), \gamma_b, \dot{\theta}, \dot{\gamma}_b, \dot{\varphi}, v_x\}$.

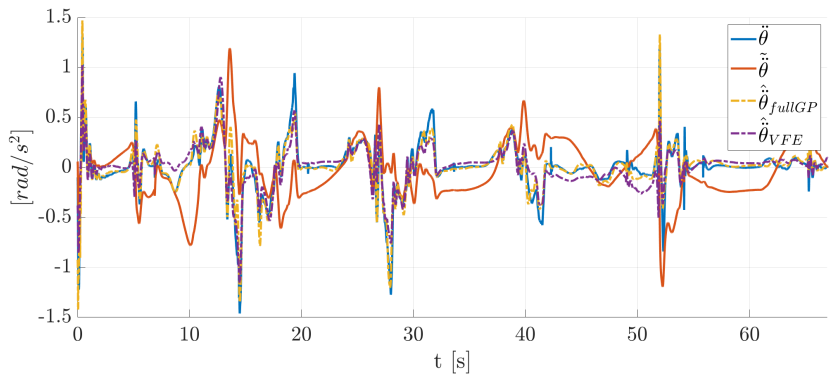
Once the feature analysis has been completed, the VFE sparse GP model approximation described in Sec. 3.2.1 has been employed for offline GP reduction. The accuracy of the models have been compared to the nominal one, assessing the acceleration estimate precision with respect to the actual ones. The grey-box acceleration estimates are evaluated both in their full version (without any reduction) and the sparse one (with VFE reduction), as depicted in Fig. 6.3. The GPs are remarkably effective in modeling the acceleration errors, obtaining precise fitting results for all the considered accelerations. Moreover, a very slight difference is present between the full and sparse implementation. In particular, note that the nominal accelerations are quite inaccurate, because of the high impact of unmodeled dynamics, e.g. suspensions and tire combined behaviour. Indeed, the suspension system is fundamental for the computation of the vertical load on the tires, that deeply influence the actual lateral and longitudinal forces acting on the motorcycle body. Moreover, the combined behaviour of the tires is particularly important in the deceleration phase before a turn and in the acceleration phase after a turn, with a high impact on yaw and direct lateral acceleration.



(a) Comparison of the direct lateral acceleration estimate.



(b) Comparison of the yaw acceleration estimate.



(c) Comparison of the roll acceleration estimate.

Figure 6.3. Acceleration estimate comparison for the considered models, i.e. nominal (red), full GP (yellow), sparse GP (purple), with respect to the actual acceleration (blue).

6.1.2 Complete Model

The dynamics model previously described has been used within the NMPC, implemented in velocity form and reformulated in spatial coordinates with respect to the arc length along the track s [96]. In this way, it is possible to define the *tracking errors* with respect to the reference path e_y and e_φ and to eliminate the dependency on the velocity in the position reference given to the NMPC.

The state of the NMPC prediction model is then

$$\boldsymbol{\xi} = [\delta,\ \gamma_t,\ \gamma_b,\ y_r, e_\varphi,\ e_y,\ \theta,\ v_x,\ v_y,\ \dot{\varphi},\ \dot{\theta}]^\top, \tag{6.6}$$

where $\delta, \gamma_t, \gamma_b, y_r$ are the actual commands to the motorcycle, i.e. steering angle, normalized throttle and brake and lateral rider body position, e_y and e_φ are the lateral and angular errors, θ is the roll angle, and $v_x, v_y, \dot{\varphi}, \dot{\theta}$ are the longitudinal, lateral, yaw and roll velocities. These states are assumed to be fully measured or computed by measurable quantities without noise. Since the NMPC is implemented in velocity form and its inputs are the derivatives of the actual commands, the NMPC input vector is $\boldsymbol{u} = [\dot{\delta},\ \dot{\gamma}_t,\ \dot{\gamma}_b, \dot{y}_r]^\top$. The resulting Ordinary Differential Equation (ODE) used in the NMPC algorithm is expressed by

$$\hat{\boldsymbol{\xi}}' = \boldsymbol{f}(\boldsymbol{\xi}(s), \boldsymbol{u}(s); \zeta(s)), \tag{6.7}$$

where $\zeta(s) = \frac{1}{\rho}$ is the trajectory curvature, with ρ instantaneous radius of the trajectory χ, and the ODE is integrated with respect to the arc length along the track s, by applying the *chain rule* $\Psi' = \frac{d\Psi}{ds} = \frac{d\Psi}{dt}\frac{dt}{ds} = \frac{d\Psi}{dt}\frac{1}{\dot{s}} = \frac{\dot{\Psi}}{\dot{s}}$. An ERK4 integrator is then applied to obtain the needed discrete evolution $\hat{\boldsymbol{\phi}}(\boldsymbol{\xi}_{j|k}, \boldsymbol{u}_{j|k})$.

6.2 Results

The LbNMPC controller has been implemented and run in co-simulation with the high-fidelity simulation engine `VI-BikeRealTime` [89]. The software allows to accurately reproduce the vehicle dynamics, in order to easily test motorcycle virtual prototypes. The main analyses have been conducted on a `VI-Track` lap (depicted in Fig. 6.2a).

Both offline and online sparse approximations have been tested within the grey-box modeling to assess the strategy capabilities. Applying only offline reduction, the *minimal* effective GP model obtained by VFE reduction resulted to be composed of 20 points. On the other hand, the application of online local approximation by Transduction Learning (TL) resulted effective with a minimum number of 10 points. To efficiently obtain the TL model online, a prior offline reduction with VFE has been accomplished, getting a 40 points-reduced dataset. Summarizing, in this section we compare the following models:

1. a grey-box model obtained by offline VFE reduction with 20 points, named *grey-box VFE*,
2. a grey-box model obtained by online Transduction Learning with 10 points applied on top of a reduced dataset of 40 points obtained by offline VFE, named *grey-box TL*,
3. the nominal physics-based model.

6.2.1 Cosimulation environment

The cosimulation relies on `VI-BRT`, that is specifically devised to emulate the behaviour of a motorcycle during high performance driving [89]. It models the system with 11 degrees of freedom (DoF), 6 for the main body, 2 for each wheel-suspension system and 1 rotation between the pinion and the main body. It comprises realistic dynamics of driveline, brakes, tires, suspensions and gyroscopic effects. Moreover, the movement of the rider is simulated as a moving mass with lateral DoF. `VI-BRT` allows to implement a cosimulation in `Simulink`, providing a simulation block that receives in inputs the vehicle commands. The implemented LbNMPC controller is then used to compute the commands derivatives $\boldsymbol{u}$ at $f_c = 100Hz$, that are integrated and sent to `VI-BRT`. The simulation software internally reproduces the motorcycle dynamics at $f_{sim} = 1000Hz$ and updates the state to be fed to the LbNMPC controller at f_c. The cosimulation has been run on a Windows 10 PC, with a CPU 11th Gen Intel(R) Core(TM) i7-11800H@2.30GHz.

6.2.2 Controller Setup

The NMPC controller relies on the NLP definition in Eq. (2.3), with the learning-based continuous dynamics constraint described in Sec. 2.4.3 and specifically defined in Eq. (2.23). The task is reference tracking on `VI-Track`, and, in order to obtain effective results, the cost function has been defined as

$$\begin{aligned} \boldsymbol{h}_j(\boldsymbol{\xi}_{j|k}, \boldsymbol{u}_{j|k}) &= [e_\varphi + \alpha, e_y, \dot{e}_\varphi, \dot{e}_y, e_v, \alpha, y_r, \gamma_t \cdot \gamma_b, \\ &\qquad \dot{\delta}, \dot{\gamma}_t, \dot{\gamma}_b, \dot{y}_r]^\top, \\ \boldsymbol{h}_N(\boldsymbol{\xi}_{N|k}, \boldsymbol{u}_{N|k}) &= [e_\varphi + \alpha, e_y, \dot{e}_\varphi, \dot{e}_y, e_v, \alpha, y_r, \gamma_t \cdot \gamma_b]^\top \end{aligned} \tag{6.8}$$

comprehensive of the path tracking errors e_ψ and e_y, to follow the given path; their derivatives, to promote smooth behaviours; the velocity error $e_v = v - v_{ref}$ where $v = \sqrt{v_x^2 + v_y^2}$, to follow the given velocity profile; the rear sideslip angle $\alpha = \text{atan}(\frac{v_y + b\dot{\psi}}{v_x})$, to maintain a sufficiently restrained behaviour of the motorcycle; the rider lateral position y_r, to favor the vertical position of the rider; the product between throttle and brake $\gamma_t \cdot \gamma_b$, to avoid their contemporary usage; and the control inputs $\dot{\delta}, \dot{\gamma}_t, \dot{\gamma}_b, \dot{y}_r$, to regularize the optimal control problem. The same quantities are used at the final prediction step, excluding the inputs.

To allow a fair comparison of the different methodologies, the NMPC controller has been set up in the same way for every trial, except for the desired change in the model. In particular, to find the common weight configuration, the genetic tuning procedure described in [48] has been applied to the nominal implementation. The procedure optimizes lap-time versus sensitivity of the lap-time itself w.r.t. weights change, generating a Pareto front that reveals their trade-off. Therefore, a configuration was selected from the Pareto front that balances the two indicators just introduced. This configuration allows the riding task to be completed with all the different models and, at the same time, guarantees sufficiently high performance to stress the differences. The employed weight matrices are then

$$\begin{aligned} W = [&0.951, 0.130, 0.900, 10^{-4}, 0.103, 2.084, 10^{-3}, 10^5, \\ &0.500, 10^{-4}, 8 \cdot 10^{-4}, 0.600] \end{aligned} \tag{6.9}$$

$$W_N = [0.951, 0.130, 0.900, 10^{-4}, 0.103, 2.084, 10^{-3}, 10^5]. \tag{6.10}$$

	nominal	grey-box VFE		grey-box TL	
e_y $[m]$	0.40	0.09	(-77.7%)	0.22	(-45.9%)
e_φ $[deg]$	1.45	0.53	(-63.0%)	0.96	(-33.5%)
e_v $[m/s]$	0.41	0.16	(-60.5%)	0.25	(-39.3%)
$T_{solver}[ms]$	3.74	6.63	(+77.2%)	4.99	(+33.4%)

Table 6.2. Root Mean Square Errors of trajectory tracking indicators, their variations w.r.t. the nominal NMPC and (Lb)NMPC solution time using the presented models on VI-Track.

The constraint functions have been defined as

$$\begin{aligned} \boldsymbol{r}_{j|k} &= [\delta, \gamma_t, \gamma_b, y_r, \dot{\delta}, \dot{\gamma}_t, \dot{\gamma}_b, \dot{y}_r]^\top, \\ \boldsymbol{r}_{N|k} &= [\delta, \gamma_t, \gamma_b, y_r]^\top, \end{aligned} \tag{6.11}$$

comprehensive of the states $\delta, \gamma_t, \gamma_b$ and y_r, that represent the actual limits of the motorcycle commands, and of the inputs $\dot{\delta}, \dot{\gamma}_t, \dot{\gamma}_b$ and $\dot{y}_r$ that support the realization of a smooth control action. Indeed, their bounds have been set as

$$\begin{aligned} \underline{\boldsymbol{r}}_k &= [-\frac{\pi}{20}, 0, 0, -0.15, -1, -10, -10, -1]^\top, \\ \overline{\boldsymbol{r}}_k &= [+\frac{\pi}{20}, 1, 1, +0.15, +1, +10, +10, +1]^\top, \\ \underline{\boldsymbol{r}}_N &= [-\frac{\pi}{20}, 0, 0, -0.15]^\top, \\ \overline{\boldsymbol{r}}_N &= [+\frac{\pi}{20}, 1, 1, +0.15]^\top. \end{aligned} \tag{6.12}$$

Note that the adopted setup of the controller guarantees the recursive feasibility of the problem since the only constrained quantities are either the actual NMPC input or their integration over time, that are exactly predicted by the NMPC controller. On the other side, as it is common for this kind of application [5, 12], formal proofs of stability and robustness are not explicitly considered, while the local convergence of the algorithm is ensured by RTI guarantees on contractivity and boundness of the loss of optimality with respect to optimal feedback control [68, 69].

Within the LbMATMPC toolbox, the ERK4 integrator has been used, with sampling step $T_s = 2m$. The sampling step is expressed in meters, thanks to the spatial reformulation detailed in Sec. 6.1.2. The QP solution is obtained through HPIPM sparse solver [67], while adopting Real-Time Iteration scheme [61] and Non Uniform integration Grid (NUG) [104] with grid defined as $G = [0:1:10, 12:2:34, 36:1:40]$, where $a:b:c$ is a vector from a to c at step b expressed in T_s unit. As a result, the number of shooting points is $N = 27$, allowing a prediction horizon of $80m$.

6.2.3 Results on Track

The controllers relying on the three models previously stated have been tested and the closed-loop results have been compared. Both grey-box models clearly outperform the nominal model in terms of tracking performance, with a great reduction of the lateral, angular, and velocity errors, as shown in Fig. 6.6. In particular, the grey-box VFE model achieves the best behavior, with 60% to nearly 80% reduction of the tracking errors,

	nominal	grey-box VFE		grey-box TL	
e_y $[m]$	0.41	0.13	(-68.2%)	0.17	(-57.8%)
e_φ $[deg]$	1.32	0.46	(-64.7%)	0.59	(-55.2%)
e_v $[m/s]$	0.30	0.05	(-85.5%)	0.07	(-75.8%)

Table 6.3. Root Mean Square Errors of trajectory tracking indicators and their variations w.r.t. the nominal NMPC using the presented models on the test track.

while the grey-box TL model obtains between 30% to 45% reduction. On the other hand, the grey-box VFE is the most computationally demanding solution, with an increment of nearly 80% for the solution time, while the grey-box TL leads to only around 30% of increment. However, the inputs computed by the grey-box controllers are quite different, as the online adaptation of the TL model introduces relevant high-frequency vibrations, as can be clearly seen in Fig. 6.7(c). Yet, the high-frequency components are mostly filtered by the actual motorcycle system, and their effect is significantly weakened in the actual velocity and roll profiles, Fig. 6.7(a), 6.7(b). Moreover, possible solutions for the input vibrations could be realized in different setups, e.g., increasing the NMPC weights on the input derivatives, employing a greater TL model, forcing temporal correlations in the chosen points. Finally, a detail of the trajectory traveled on the first chicane is depicted in Fig. 6.5, where the similarity of the two LbNMPC controllers is displayed, while Tab. 6.2 quantifies the performance, reporting the numerical results in terms of errors and solution time, together with their variation with respect to the nominal case, confirming previous observations.

Summarizing, both the proposed models highly improve the closed-loop performance, highlighting the trade-off between computational burden and input consistency. Indeed, the proposed TL model, based on only 10 points, has been chosen to obtain a significant solver time reduction with respect to the offline VFE approximation, while both strategies highlight the potential of the approach to enhance the controller action and emphasize the possible weaknesses.

Generalization Capability

In order to verify the generalization capability of the proposed LbNMPC controller, an analysis on a test track that was not used in the training phase (shown in Fig. 6.4) has

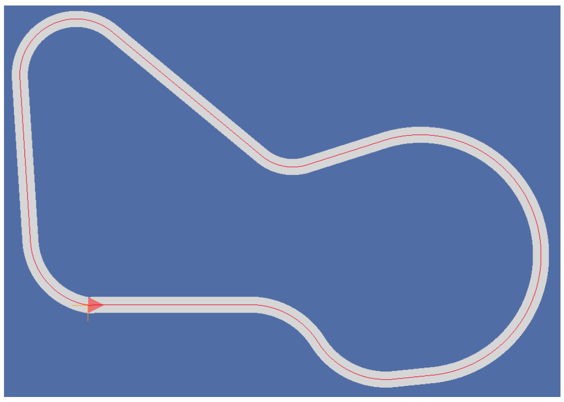


Figure 6.4. Test track used for generalization capability assessment.

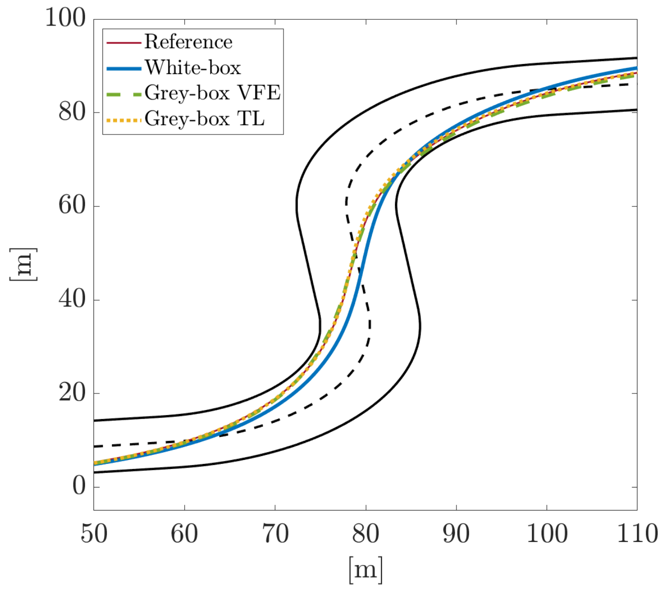
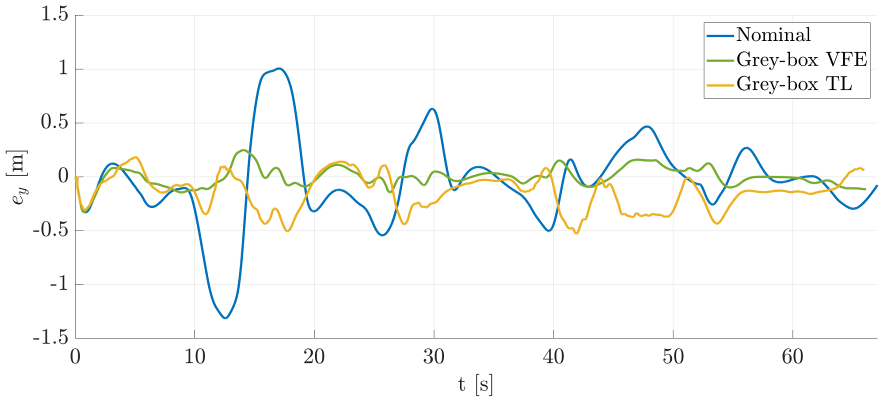


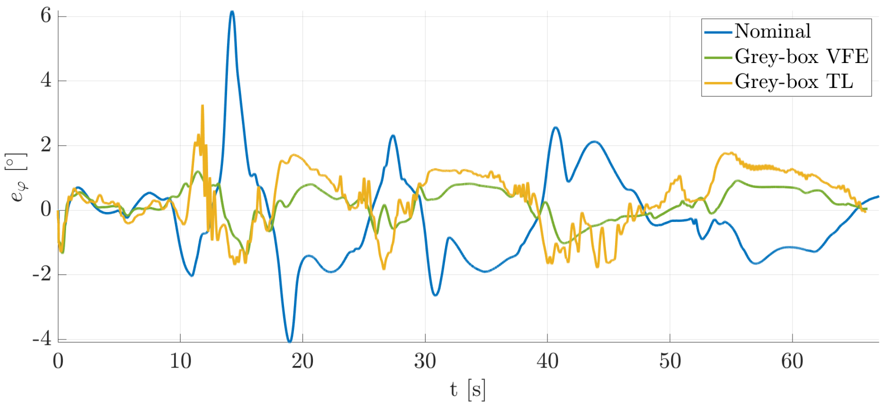
Figure 6.5. Detail of the travelled trajectory on the first chicane of *VI-Track*.

been conducted. Consistently with the results on VI-Track, the tracking performance of both grey-box VFE and grey-box TL are remarkably better then the nominal NMPC (as shown in Fig. 6.8 and reported in Tab. 6.3), obtaining 65% to 85% reduction of the tracking errors using grey-box VFE model, and between 55% to 75% reduction using grey-box TL model. The test shows a certain level of generalization capability of the obtained LbNMPC controller, assessing the efficacy of the learned acceleration errors.

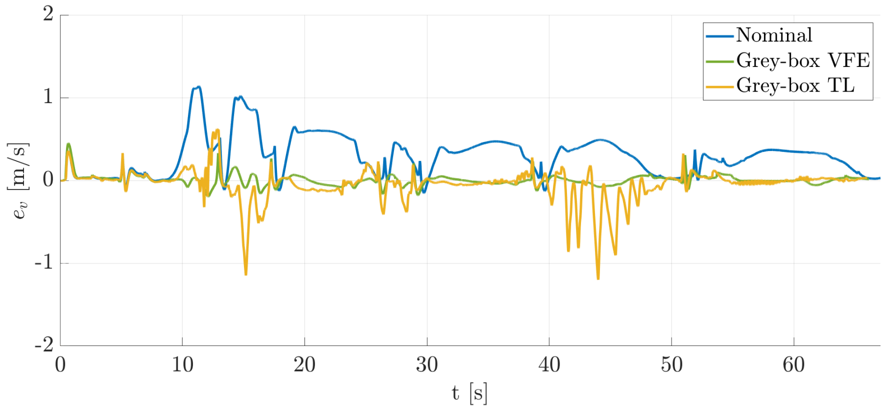
Moreover, given the superior behaviour of the LbNMPC controller with respect to the nominal NMPC, the results also suggest the possibility to ride more extreme maneuvers using the grey-box modeling within the controller.



(a) Lateral trajectory error e_y along the simulation.

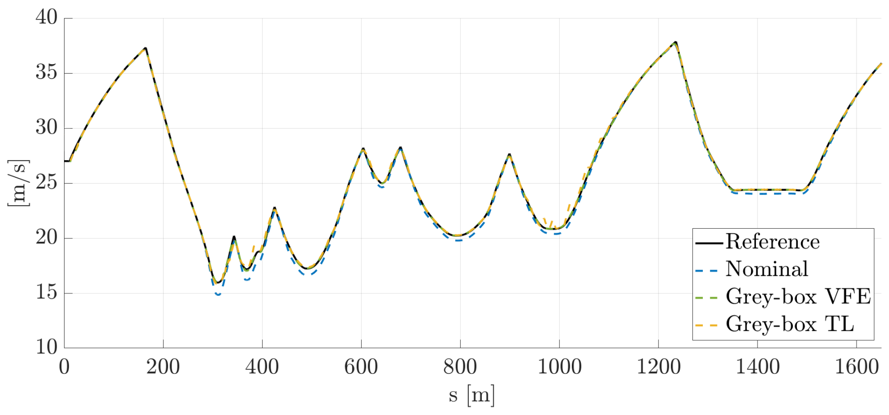


(b) Yaw trajectory error e_φ along the simulation.

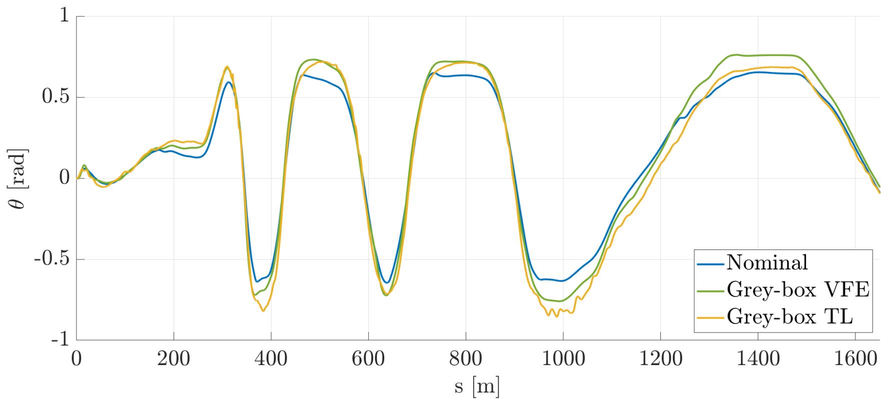


(c) Velocity trajectory error e_v along the simulation.

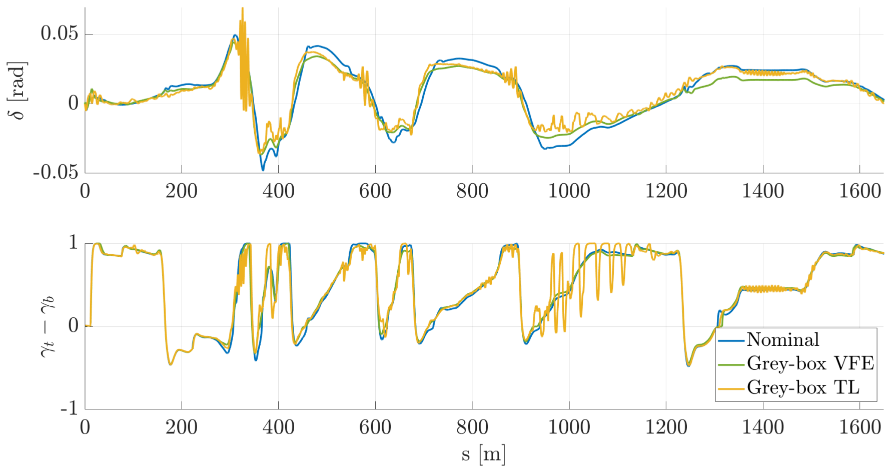
Figure 6.6. Tracking errors obtained in simulation on VI-Track using the nominal physics-based model (blue), grey-box with VFE offline reduction only (green), grey-box with VFE offline reduction and Transduction Learning online reduction (yellow).



(a) Longitudinal velocity v_x along the track.

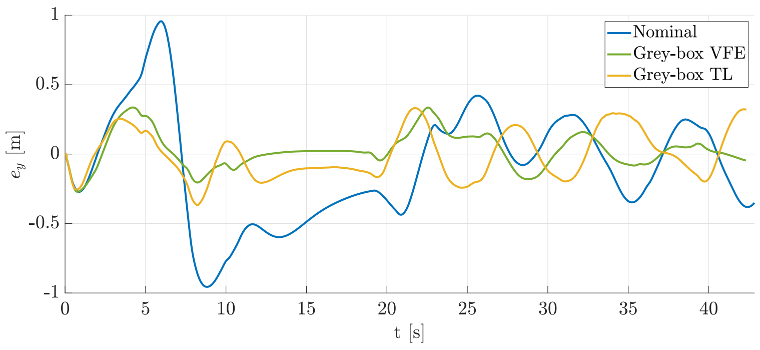


(b) Roll angle θ along the track.

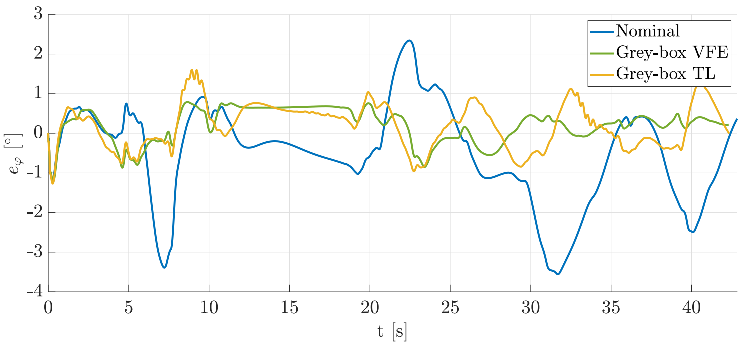


(c) Steering input δ_f and throttle/braking effort $\gamma = \gamma_t - \gamma_b$ along the track.

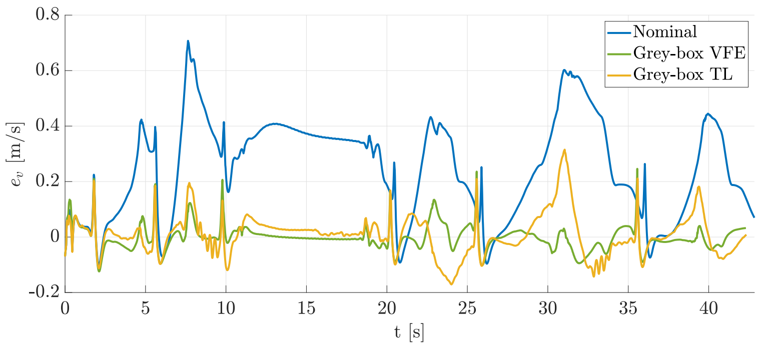
Figure 6.7. Motorcycle behavior obtained in simulation on VI-Track using the nominal physics-based model (blue), grey-box with VFE offline reduction only (green), grey-box with VFE offline reduction and Transduction Learning online reduction (yellow).



(a) Lateral trajectory error e_y along the simulation.



(b) Yaw trajectory error e_φ along the simulation.



(c) Velocity trajectory error e_v along the simulation.

Figure 6.8. Tracking errors obtained in simulation on the test track using the nominal physics-based model (blue), grey-box with VFE offline reduction only (green), grey-box with VFE offline reduction and Transduction Learning online reduction (yellow).

Part III

Four-wheel Vehicles

In this part, the applications related to four-wheels vehicles are presented. Chapter 7 describes an NMPC controller for a virtual high-performance car, that is the foundation of the LbNMPC presented in the next chapter. Indeed, Ch. 8 reports a LbNMPC controller for high-performance driving based on a grey-box model of the vehicle velocities. Finally, Ch. 9 illustrates a LbNMPC strategy based on a black-box model applied to an experimental go-kart.

7

A Contouring NMPC for a Four-wheel Vehicle in Real-Time

In this Chapter, a NMPC controller for a virtual four-wheel vehicle (depicted in 7.1) at the limit of handling is presented. The goal is to minimize the travel time on a desired path, while computing both the trajectory and the velocity profile for the vehicle. A physics-based model that balance the need to provide an accurate prediction in high-performance with limited computational load is detailed in Sec. 7.1. The results, illustrated in Sec. 7.2, show the proficiency of the obtained control action in a high fidelity vehicle simulation environment, comparing the behaviour with a commercial lap-time minimizer.

Figure 7.1. Representation of a virtual 4-wheel vehicle.

Table 7.1. Main model quantities and parameters description.

Symbol	Description
x	longitudinal position
y	lateral position
ψ	yaw angle
β	sideslip angle of the vehicle
δ_f	steering angle of the front wheels
m	vehicle mass
I_z	inertia around the vertical axis of the vehicle
a	front wheels to CM longitudinal distance
b	rear wheels to CM longitudinal distance
c	wheels to CM lateral distance
h_{CM}	height of the vehicle's CM
$F_{\{l,c\}_{\{i,j\}}}$	wheels lateral/longitudinal forces in the tire frames[1]
$F_{\{x,y\}_{\{i,j\}}}$	wheels lateral/longitudinal forces in the vehicle's frame
$F_{z_{\{i,j\}}}$	normal forces on the wheels
F_x^d	downforce in the vehicle's frame
$F_{z_i}^d$	longitudinal drag force in the vehicle frame
s	curvilinear abscissa
e_y	lateral tracking error
e_ψ	angular tracking error

7.1 Model for Control Synthesis

The proposed internal Nonliner Model Predictive Contouring Controller (NMPCC) model is a four-wheel vehicle based on the description in [105], supplemented by load transfers, gear shift predictions, longitudinal force saturation, and an ellipsoidal tire friction constraint, improving the overall prediction capabilities of the controller. Moreover, a time-varying parameter has been used to model the lateral force dependency on the longitudinal slip, in order to handle locally the tire forces coupling. The model has been reformulated with respect to the curvilinear abscissa s.

7.1.1 Model Dynamics

The vehicle dynamics model is described as

$$\dot{\xi} = \phi(\xi(t), u(t); p(t)), \tag{7.1}$$

where $\xi(t) \in \mathbb{R}^{n_x}$ is the state of the vehicle, $u(t) \in \mathbb{R}^{n_u}$ is the input and $p(t) \in \mathbb{R}^{n_p}$ is the time-varying parameter vector.

The system dynamics ϕ is characterized by the equation of motion of the vehicle center

[1]Subscripts $i \in \{f, r\}$ refer to front or rear wheels, $j \in \{l, r\}$ left or right wheels.

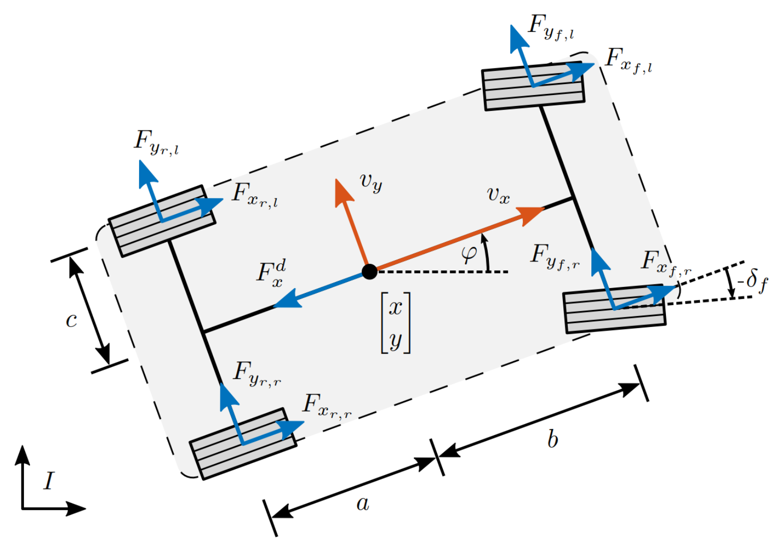


Figure 7.2. Quantities defined in the vehicle model for control.

of mass (CM) [4], i.e.

$$\begin{aligned} \ddot{x} &= \dot{y}\dot{\psi} + \frac{1}{m}\left(\sum_{i,j} F_{x_{i,j}} - F_x^d\right), \quad \ddot{y} = -\dot{x}\dot{\psi} + \frac{1}{m}\left(\sum_{i,j} F_{y_{i,j}}\right), \\ \ddot{\psi} &= \frac{1}{I_z}\left[a\left(\sum_j F_{y_{f,j}}\right) - b\left(\sum_j F_{y_{r,j}}\right) + c\left(\sum_i F_{x_{i,r}} - \sum_i F_{x_{i,l}}\right)\right], \end{aligned} . \tag{7.2}$$

The projection of cornering and longitudinal forces in the vehicle frame yields

$$\begin{aligned} F_{x_{f,j}} &= F_{l_{f,j}} \cos(\delta_f) - F_{c_{f,j}} \sin(\delta_f), \quad F_{x_{r,j}} = F_{l_{r,j}}, \\ F_{y_{f,j}} &= F_{l_{f,j}} \sin(\delta_f) + F_{c_{f,j}} \cos(\delta_f), \quad F_{y_{r,j}} = F_{c_{r,j}}. \end{aligned} \tag{7.3}$$

The steering angle is here assumed to be the same for both front wheels.

In Fig. 7.2 the physical quantities involved and directions and application point of the forces are described.

The dynamics in the inertial frame are then computed as

$$\begin{aligned} \dot{X} &= \dot{x} \cos(\psi) - \dot{y} \sin(\psi), \\ \dot{Y} &= \dot{x} \sin(\psi) + \dot{y} \cos(\psi) \end{aligned} \tag{7.4}$$

where (X, Y) is the position of the vehicle's CM in the inertial frame.

Finally, the *sideslip angle* of the vehicle is defined as

$$\beta = \text{atan}\left(\frac{\dot{y}}{\dot{x}}\right). \tag{7.5}$$

7.1.2 Forces

The longitudinal drag force and the downforce are modeled as [106]

$$F^d_{x,z_i} = \frac{1}{2}\,\rho\, C_{x,z_i}\, A_{x,z_i}\, \dot{x}^2, \tag{7.6}$$

where ρ is the air density, C_{x,z_i} are the drag coefficients and A_{x,z_i} are the interested section area. The longitudinal tire forces in each wheel reference frame are computed as

$$F_{l_{i,j}} = f_{\text{eng}_{i,j}} - f_{\text{brk}_{i,j}}, \tag{7.7}$$

where the engine and braking forces are

$$f_{\text{eng}_{i,j}} = \text{sat}\left(\frac{\tau_{\text{eng}_i}}{r_w},\ \mu F_{z_{i,j}}\right),\quad f_{\text{brk}_{i,j}} = \text{sat}\left(\frac{\tau_{\text{brk}_i}}{r_w}, \mu F_{z_{i,j}}\right), \tag{7.8}$$

where μ is the tire friction coefficient, r_w is the wheel radius and the saturation function is defined as $\text{sat}(f_a, f_b) = \frac{f_b}{1+\exp(-5(\frac{f_a}{f_b}-\frac{1}{2}))}$. Then, the engine and braking torques at the wheels are:

$$\tau_{\text{eng}_i} = \gamma_t\,(\tau^{\text{MAX}}_{\text{eng},i} - \tau^{\text{min}}_{\text{eng},i}) + \tau^{\text{min}}_{\text{eng},i},\quad \tau_{\text{brk}_i} = \gamma_b\,\tau^{\text{MAX}}_{\text{brk},i}, \tag{7.9}$$

where $\gamma_{t,b}$ are the normalized throttle and braking efforts, $\tau^{\text{MAX}}_{\text{brk},i}$ is the maximum torque given by the braking system to front/rear wheels, $\tau^{\text{MAX,min}}_{\text{eng},i}$ are the maximum and minimum torque values expressed by the engine at front/rear wheels at a given gear and are changed as a time-varying parameter to model gearshift. To compute the torques in the prediction horizon, an iterative strategy predicting the engine rpm, and hence gearshift, based on the predicted velocity is used. Specifically, the engine rpm are computed as

$$\text{rpm}^{\text{pred}} = \frac{v_x^{\text{pred}}}{r_w}\,\frac{\text{diff}_{\text{ratio}}}{\text{gear}_{\text{ratio}}}\,\frac{60}{2\pi}, \tag{7.10}$$

where $\text{diff}_{\text{ratio}}$ and $\text{gear}_{\text{ratio}}$ are the input/output torque ratios at the differential and at the gearbox (in a specific gear), respectively. The dependence of $\tau^{\text{MAX,min}}_{\text{eng},i}$ with respect to the engine rotational velocity has been neglected.

The normal forces are modeled as

$$\begin{aligned}
F_{z_{f,l}} &= \frac{F^0_{zf}}{2} + \frac{F^d_{zf}}{2} - \Delta^{\text{lon}}_{Fz} - \Delta^{\text{lat}}_{Fz},\\
F_{z_{f,r}} &= \frac{F^0_{zf}}{2} + \frac{F^d_{zf}}{2} - \Delta^{\text{lon}}_{Fz} + \Delta^{\text{lat}}_{Fz},\\
F_{z_{r,l}} &= \frac{F^0_{zr}}{2} + \frac{F^d_{zr}}{2} + \Delta^{\text{lon}}_{Fz} - \Delta^{\text{lat}}_{Fz},\\
F_{z_{r,r}} &= \frac{F^0_{zr}}{2} + \frac{F^d_{zr}}{2} + \Delta^{\text{lon}}_{Fz} + \Delta^{\text{lat}}_{Fz},
\end{aligned} \tag{7.11}$$

where the load transfer in steady state condition Δ_{Fz} is computed by

$$\Delta^{\text{lon}}_{Fz} = F^{\text{sat0}}_x \frac{h_{\text{CM}}}{a+b},\qquad \Delta^{\text{lat}}_{Fz} = F^{\text{static}}_y \frac{h_{\text{CM}}}{2c}. \tag{7.12}$$

To avoid an algebraic loop in the model, the forces used for the load transfer dynamics computation are F_x^{sat0} (total longitudinal force expressed in the vehicle frame saturated at nominal F_z) and F_y^{static} (the sum of the lateral forces computed at nominal F_z on each wheel).

The lateral forces model is based on *Pacejka's Magic Formula* [107]. The lateral forces expression is

$$F^0_{c_{i,j}} = D\text{sin}(C\text{atan}(B\beta_{i,j} - E(B\beta_{i,j} - \text{atan}(B\beta_{i,j})))), \tag{7.13}$$

where B is the *stiffness factor*, that regulates the slope, C is the *shape factor*, that determines the *stretching* with respect to the side slip, D is the *peak factor*, that defines the peak value, E is the *curvature factor*, that affects the transition to the constant saturation value and $\beta_{i,j}$ are the wheels side slip angles. Moreover, local information about the tire coupling has been included through the introduction of the term $G_{yk_{i,j}}$ as a time-varying parameter, computed as [107]

$$\begin{aligned} G_{yk_{i,j}} =& \frac{1}{G_{yk0}}\{\cos(C_{yk}atan(B_{yk}(\beta_{i,j})k_{i,j} - E_{yk}(B_{yk}(\beta_{i,j})\cdot \\ & \cdot k_{i,j} - atan(B_{yk}(\beta_{i,j})k_{i,j}))))\}, \end{aligned} \tag{7.14}$$

where $k_{i,j}$ is the current longitudinal slip, B_{yk} is a function of the lateral slip, E_{yk}, C_{y_k} and G_{yk0} are tire dependent parameters. The structure of the lateral force factors leads to a cornering force model of the form

$$F_{c_{i,j}} = F^0_{c_{i,j}} \cdot G_{yk_{i,j}} = \Psi(\beta_{i,j}, F_{z_{i,j}}; G_{yk_{i,j}}). \tag{7.15}$$

The sideslip angle of each wheel is computed as

$$\beta_{i,j} = \text{atan}\left(\frac{v_{c_{i,j}}}{v_{l_{i,j}}}\right), \tag{7.16}$$

where the longitudinal and lateral velocities in each wheel frame are

$$\begin{aligned} v_{c_{f,j}} &= v_{y_{f,j}}\text{cos}(\delta_f) - v_{x_{f,j}}\text{sin}(\delta_f), \;\; v_{c_{r,j}} = v_{y_{r,j}}, \\ v_{l_{f,j}} &= v_{y_{f,j}}\text{sin}(\delta_f) + v_{x_{f,j}}\text{cos}(\delta_f), \;\; v_{l_{r,j}} = v_{x_{r,j}}, \end{aligned} \tag{7.17}$$

and the velocities along x and y directions are

$$\begin{aligned} v_{y_{f,j}} &= \dot{y} + a\dot{\psi}, \;\; v_{y_{r,j}} = \dot{y} - b\dot{\psi}, \\ v_{x_{i,l}} &= \dot{x} - c\dot{\psi}, \;\; v_{x_{i,r}} = \dot{x} + c\dot{\psi}. \end{aligned} \tag{7.18}$$

7.1.3 Spatial Reformulation

The dynamics have been reformulated in spatial coordinates with respect to the arc length s along the track in order to allow time to be a minimization variable, to set the spatial constraints, and to eliminate the dependency on the velocity in the trajectory reference for the controller. This has been previously studied, e.g. in [108], [47], and the procedure is summarized in the following for completeness. The reference trajectory σ is parameterized in s and the *curvature* of the trajectory $\zeta = \frac{1}{\rho}$, where ρ is the instantaneous curvature radius, is locally defined as

$$\zeta = \frac{x'y'' - y'x''}{(x'^{\,2} + y'^{\,2})^{\frac{3}{2}}}, \tag{7.19}$$

where $f' = \frac{df(s)}{ds}$ e $f'' = \frac{d^2f(s)}{ds^2}$.

The lateral and angular *tracking errors* are computed as

$$\begin{aligned} e_\psi &= \psi - \psi_s, \\ e_y &= (Y_s - Y)\ \cos(\psi_s) + (X_s - X)\ \sin(\psi_s), \end{aligned} \tag{7.20}$$

where ψ_s is the reference yaw angle and (X_s, Y_s) is the absolute reference position at the current spatial coordinate while (X, Y) and ψ are the absolute current position and yaw angle. The tracking errors dynamics are [98]

$$\begin{aligned} \dot{e}_\psi &= \dot{\psi} - \zeta\ \dot{s}, \\ \dot{e}_y &= v_x\ \sin(e_\psi) + v_y\ \cos(e_\psi). \end{aligned} \tag{7.21}$$

The state vector ξ is differentiated w.r.t s using the *chain rule* as

$$\xi' = \frac{d\xi}{ds} = \frac{d\xi}{dt}\frac{dt}{ds} = \frac{d\xi}{dt}\frac{1}{\dot{s}} = \frac{\dot{\xi}}{\dot{s}},\quad \forall \dot{s} \neq 0, \tag{7.22}$$

where $\dot{s} = \frac{1}{1-\zeta\ e_y}\,(\dot{x}\ \cos(e_\psi) - \dot{y}\ \sin(e_\psi))$.

7.1.4 Full Model

The minimization of the travel time over the prediction horizon must be considered in the definition of the cost function to enable the contouring formulation; hence, the time has been included as a state variable with the following ODE $\dot{t} = \frac{1}{\dot{s}}$. The full state vector is then given by

$$\xi = [\dot{x}, \dot{y}, \dot{\psi}, e_\psi,\ e_y,\ \delta_f,\ \gamma_t,\ \gamma_b, t]^\top, \tag{7.23}$$

where these states are assumed to be fully measured or computed by measurable quantities without noise, and the input computed by the algorithm is

$$u = [\dot{\delta}_f,\ \dot{\gamma}_t,\ \dot{\gamma}_b,\ \epsilon_{\text{slip}},\ \epsilon_{\text{err}},\ \epsilon_{\text{gg}}]^\top, \tag{7.24}$$

where $\dot{\delta}_f$, $\dot{\gamma}_t$, $\dot{\gamma}_b$ are the derivatives of the actual input to the vehicle and ϵ are slack variables, whose role will be discussed in Sec. 7.2.1. This formulation allows a smooth action of the controller and avoids too aggressive, unrealistic behaviors.

The dynamics equation of the model used in the NMPC algorithm can be compactly written as

$$\xi' = \phi(\xi(s), u(s); p(s)), \tag{7.25}$$

where $p(s) = \left[\zeta(s), \tau_{\text{eng},i}^{\text{MAX,min}}(s), G_{yk_{i,j}}(s)\right]^\top$ is the time-varying parameter vector ex-

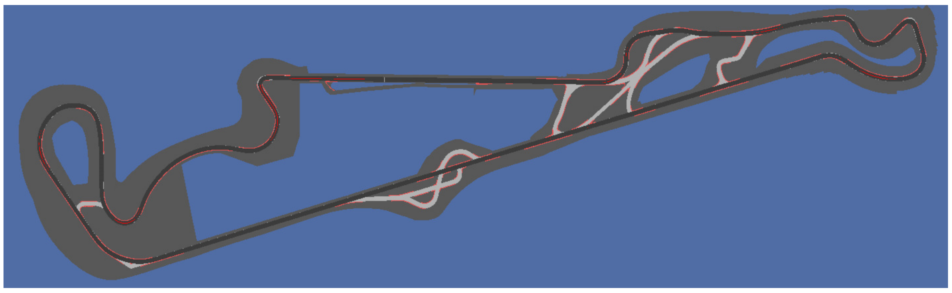


Figure 7.3. The Paul Ricard race track used for testing. The chosen course is highlighted in black.

pressed as function of the curvilinear abscissa s.

7.2 Results

Since the controller must be able to handle challenging high-performance situations, a validation has been made by comparing with a commercial dynamic minimum lap-time tool, `VI-MaxPerformance`. `VI-MaxPerformance` is based on an iterative procedure that acts on the velocity profile to minimize lap time [109]. The comparison has been made on virtualization of the Paul Ricard race track (Le Castellet, France), shown in Fig. 7.3, which is $5665m$ long.

The vehicle dynamics are computed using the simulation tool `VI-CarRealTime` (`VI-CRT`), a multi-body software specifically designed to reproduce vehicle behavior for high performance driving in real-time [109]. The underlying simulation model is composed of 14 degrees of freedom, 6 for the chassis and 2 for each wheel, and it includes comprehensive dynamics of tire, chassis, suspensions, brakes, engine, and transmission.

The co-simulation is performed in `Simulink`, connecting a `VI-CRT` simulation block with the `NMPC` controller. In particular, the controls $\dot{\delta}_f, \dot{\gamma}_t$ and $\dot{\gamma}_b$ computed by the NMPCC controller are integrated and given as inputs to the simulation block. `VI-CRT` is used to simulate the dynamics of the vehicle at 1000 Hz while the control action is updated by the NMPCC controller at $f_c = 50$ Hz.

7.2.1 Controller Setup

The NMPC controller relies on the NLP definition in Eq. (2.3). To effectively generate the vehicle commands, the cost function for the NMPC is defined as

$$\begin{aligned} \boldsymbol{h}_k(\xi_k, u_k) &= [\beta, \gamma_t \cdot \gamma_b, \zeta \cdot \gamma_t, t, \dot{\delta}_f, \dot{\gamma}_t, \dot{\gamma}_b, \epsilon_{\text{slip}}, \epsilon_{\text{err}}, \epsilon_{\text{gg}}]^\top, \\ \boldsymbol{h}_N(\xi_N) &= [\beta, \gamma_t \cdot \gamma_b, \zeta \cdot \gamma_t, t, e_y - e_y^{\text{ref}}, \dot{e}_y, e_\psi - e_\psi^{\text{ref}} + \beta, \dot{e}_\psi]^\top. \end{aligned} \tag{7.26}$$

The penalty on the sideslip β is used to choose how restrained the behavior of the vehicle should be. The cost $\gamma_t \cdot \gamma_b$ penalizes simultaneous throttling and braking, whereas the $\zeta \cdot \gamma_t$ cost is used to make the controller accelerate smoothly while exiting the curves. The objective variable time t supports the computation of a time-minimizing path and its weight can be used to tune the importance of the performance index. The terms on the inputs ensure a smooth control action. The slack variables are used to define the *soft constraints* [110] discussed in the next paragraph. Finally, the terms related to errors e_y and e_ψ, used only as terminal objective variables, are introduced to integrate information about the trajectory over the prediction horizon.

The constraints are defined as

$$\begin{aligned}\boldsymbol{r}_k &= [\delta_f, \gamma_t, \gamma_b, \dot{\delta}_f, \dot{\gamma}_t, \dot{\gamma}_b, \epsilon_{\text{slip}}, \epsilon_{\text{err}}, \epsilon_{\text{gg}}, \beta + \epsilon_{\text{slip}}, e_y + \epsilon_{\text{err}}, (\mu_x \frac{\ddot{x}_{\text{ext}}}{g})^2 + (\mu_y \frac{\ddot{y}_{\text{ext}}}{g})^2 + \epsilon_{\text{gg}}]^\top, \\ \boldsymbol{r}_N &= [\delta_f, \gamma_t, \gamma_b]^\top,\end{aligned} \tag{7.27}$$

where the constraints on δ_f, γ_t and γ_b are intrinsic bounds of the actual vehicle controls, while those on $\dot{\delta}_f, \dot{\gamma}_t$ and $\dot{\gamma}_b$ are added in order to improve the smoothness of the computed inputs and can be used to easily tune the aggressivity of the NMPC driving commands. Also, the slack variables have been constrained in order to help the optimization procedure restricting the *search space* of the inputs. Finally, three *soft constraints* have been added to make the controller predict a trajectory with the desired features while supporting the robustness of the overall procedure. Specifically, the first soft constraint is introduced on the sideslip of the vehicle in order to force the controller to regain control of the vehicle in case of skidding; the second one is used to firmly correct the trajectory in case of out-of-the-track future situations. Finally, the last soft constraint is designed to make the controller respect the required *gg diagram*, which represent the maximum combined longitudinal-lateral acceleration that can be induced by the combined longitudinal-lateral behavior of the tire forces [111]. μ_x and μ_y are the longitudinal and lateral friction coefficient of the tires, respectively, whereas the considered accelerations on the vehicle are

$$\begin{aligned}\ddot{x}_{\text{ext}} &= \frac{\sum_{i,j} F_{x_{i,j}} - F_d^x}{m}, \\ \ddot{y}_{\text{ext}} &= \frac{\sum_{i,j} F_{y_{i,j}}}{m}.\end{aligned} \tag{7.28}$$

The NMPC tool `MATMPC` has been set to use Explicit Runge-Kutta 4 integrator, HPIPM as QP sparse solver [67] with RTI scheme [61]. To achieve satisfactory performance in trajectory planning, the prediction horizon should be long enough to predict the entire dynamics of both the longest corner on the track, to start positioning and braking in advance, and of consecutive corners in the chicanes, to calculate a trajectory consistently. However, to maintain real-time capabilities of the controller, the number of shooting points was limited to $N = 140$. Moreover, to maintain good tracking performances, a dense integration at the beginning of the control horizon is needed. Therefore, Non-Uniform Grid (NUG) steps [104] have been set as follows

$$G = [2, 3, 4, 6, 8, 10, 13\colon 3\colon 390, 392\colon 2\colon 396, 397\colon 1\colon 400], \tag{7.29}$$

where $a\colon b\colon c$ means from a to c with step b in meters, enabling a prediction length of 400 meters on track. Note that the first step, that is 2 meters long, allows computing coherently the first input, i.e. valid for more than one control period ($T_c = f_c^{-1} = 20ms$), until the velocity is less than $100\frac{m}{s}$. The implemented configuration, using RTI scheme and NUG, is specifically realized as to effectively control the vehicle at the limit of handling. In this conditions, in fact, the control frequency must be high enough, e.g.

to react in high sideslip condition with a counter-steering action, while the prediction horizon must be long enough, as stated before. To ensure both features, a very fast solution method is implemented through RTI scheme, while a reduction of the needed points in the prediction horizon is achieved by NUG.

An empirical analysis has been conducted and the weights for the cost function have been manually set as

$$W = \mathrm{diag}([5\cdot 10^3, 10^5, 6\cdot 10^{-1}, 10^1, 4\cdot 10^1, 10^{-2}, 10^{-2}, 3\cdot 10^2, 5\cdot 10^{-1}, 10^2]). \tag{7.30}$$

In particular, the sideslip weight and the time weight have been set as to maintain the correct trade-off between a restrained behavior and high performance, a high weight on $\gamma_t \cdot \gamma_b$ is used for preventing contemporary throttling and braking, the weight on $\zeta \cdot \gamma_t$ has been set to obtain a throttle action at the curve exit that does not lead to tire spinning, the inputs derivatives have been penalized to obtain the desired smoothness of the control actions, the weights on the slack variables for sideslip and lateral errors have been increased until the controller firmly corrects the vehicle behavior in unstable or out-of-the-track future situations, and finally a high penalty on the *gg constraint* slack variable has been set to comply with the desired maximum performance.

The terminal weighting matrix has then been set as

$$W_N = \mathrm{diag}([5\cdot 10^3, 10^5, 6\cdot 10^{-1}, 10^1, 10^2, 10^{-2}, 10^3, 10^0]). \tag{7.31}$$

The terminal weights have been set unchanged for the present variables already analyzed, whereas the weights on the reference errors have been set as to address that the predicted trajectory concludes, at the end of the prediction horizon, within a reference corner-cutting trajectory, whose generation is discussed in the next paragraph.

The reference trajectory for the final point of the prediction horizon employed in this test has been computed using the *Corner Cutting* procedure embedded in the commercial tool `VI-Road`. The process, given the centerline, the track width, and the vehicle width, computes a path that minimizes

$$J_{\text{path}} = \int (w_\iota \cdot \iota(s)^2 + w_\zeta \cdot \zeta(s)^2) ds, \tag{7.32}$$

where $\iota(s)$ is a measure of the path length, $\zeta(s)$ is the local path curvature and $w_\iota = 6\cdot 10^{-3}$, $w_\zeta = 4$ are the respective weights. The minimization is constrained in order to maintain the vehicle within the track.

Finally, the bounds in (7.27) have been set as

$$\begin{aligned}
\underline{\boldsymbol{r}}_k &= [-\frac{\pi}{20}, 0, 0, -2, -10, -100, -2\pi, -10^5, -10^5, -10^5, -\frac{\pi}{15}, -5, 0]^\top,\\
\overline{\boldsymbol{r}}_k &= [+\frac{\pi}{20}, 1, 1, +2, 1.1, +100, +2\pi, +10^5, +10^5, +10^5, +\frac{\pi}{15}, +5, 1.9]^\top,\\
\underline{\boldsymbol{r}}_N &= [-\frac{\pi}{20}, 0, 0]^\top, \quad \overline{\boldsymbol{r}}_N = [+\frac{\pi}{20}, 1, 1]^\top,
\end{aligned} \tag{7.33}$$

where the bound $\delta_f \in [-\frac{\pi}{20}, +\frac{\pi}{20}]$ is the limit of the vehicle steering system; γ_t and $\gamma_b \in [0, 1]$ are normalized; the bounds on the control derivatives are set as to ensure a smooth action of the controller; the bounds on the slack variables $\epsilon \in [-10^5, +10^5]$ are set to

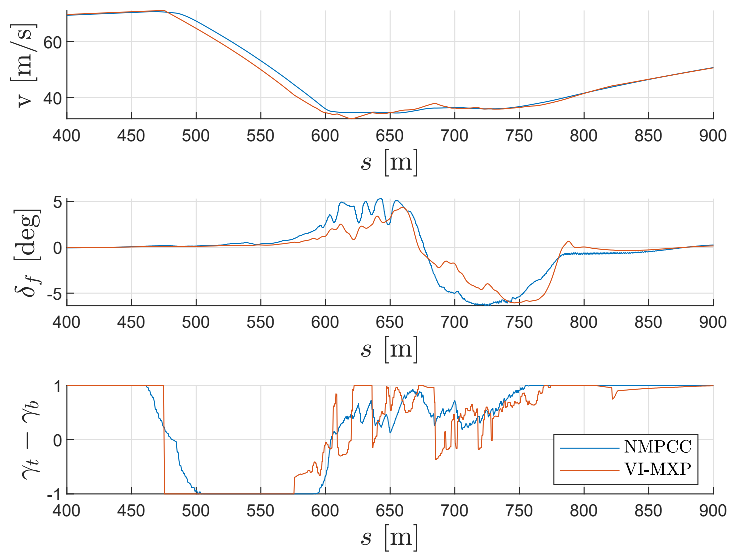


Figure 7.4. The velocity mantained and the controls applied by the NMPCC and the commercial tool VI-MXP on the first chicane of the track (between 400m and 900m).

increase the robustness of the optimization procedure; the limit on the soft constraint $\beta + \epsilon_{\text{slip}} \in [-\frac{\pi}{15}, +\frac{\pi}{15}]$ is set to heavily penalize excessive oversteering and the one on the lateral error $e_y + \epsilon_{\text{err}} \in [-5, +5]$ as to force the vehicle within the track boundaries; finally, the bound on the soft constraint for the gg is set as the vehicle acceleration limit[2].

7.2.2 Comparison with `VI-MaxPerformance`

The results on the *Paul Ricard* track show the capability of the controller to achieve performance comparable with those of the commercial tool `VI-MaxPerformance` (`VI-MXP`)[3]. In particular, `VI-MXP` has been tested both on the *corner-cutting* path and on the NMPCC controller trajectory, leading to a lap-time of 113.27s and 112.46s, respectively, while for the NMPCC controller the lap-time is 112.16s. Interestingly, the NMPCC trajectory results to be faster than the corner-cutting one also for the commercial lap time minimizer.

The commands given by `VI-MXP` following the NMPCC path and the NMPCC are shown in Fig. 7.4, where it can be observed that the NMPCC exhibits a smoother behavior than `VI-MXP`. This affects in particular the velocity profile, which is smoother and more representative of a human behavior in the NMPCC case. Note that the smoothness of the control actions is fundamental for the assistance system in order to propose consistent and human-like behavior.

The mean computational time for the controller is 12ms while the maximum is 16ms,

[2]The vehicle used for testing is a race car with aerodynamic downforce, hence the limit is greater than 1.

[3]`VI-MXP` uses a specific static solver to compute a velocity profile for the given vehicle compatible with the requested trajectory and then it drives the vehicle at the obtained speed to verify if the computed profile is feasible. If the velocity path is unfeasible (maximum path distance exceeded, zero velocity, etc.) the solver turns back to the static and computes a new velocity profile, modifying only the portion nearby the unfeasible point. The process is iterated until a feasible velocity profile is found.

Table 7.2. Comparison between NMPCC and VI-MXP.

	NMPCC	VI-MXP
lap time	$112.16s$	$112.46s$
mean longitudinal velocity	$50.3\ m/s$	$50.0m/s$
maximum lateral acceleration	$1.9\ m/s^2$	$1.9\ m/s^2$
maximum sideslip angle	$3.4deg$	$5.1deg$
maximum throttle and braking derivative	$37.6\ s^{-1}$	$200\ s^{-1}$

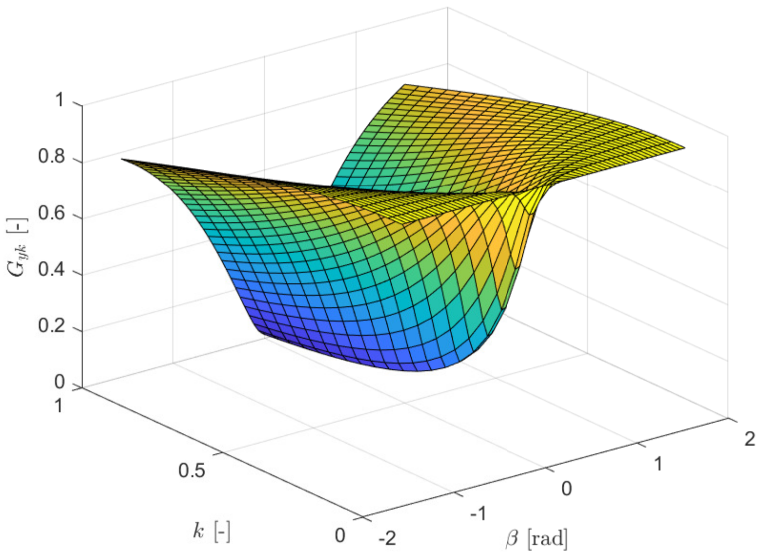


Figure 7.5. Representation of the combined lateral coefficient G_{yk} w.r.t. lateral and longitudinal slip of the tire.

which is compatible with the 50Hz control frequency in this real-time application.

7.2.3 Impact of the local information on the combined tire condition

Controlling a vehicle at the limit of handling requires a sufficiently accurate knowledge of tire forces. In fact, under conditions of high longitudinal forces, e.g. originated by accelerating after a curve or braking before it, a significant reduction in tire lateral force occurs, with a relevant impact on vehicle behavior. The information described through the time-varying parameter G_{yk}, see Eq. (7.14) and Fig. 7.5, is then crucial to correctly characterize such conditions. Since G_{yk} varies at very high frequency, the parameter is provided to the controller and processed as linearly fading in the first 5 steps of the prediction horizon. The impact of such a solution is evaluated on the last turn of the track through two closed-loop simulations obtained with and without scaling of the lateral forces with G_{yk} (Figs. 7.6 and 7.7), namely *Sim w/* G_{yk} and *Sim w/o* G_{yk}. In both cases, one-step-ahead sideslip predictions and steering derivative control are shown. An offline evaluation of the same quantities computed with the alternative approach, i.e., with G_{yk} in the *Sim w/o* G_{yk} and vice-versa, is over-imposed for comparison. It can be noticed that the one-step-ahead sideslip predictions obtained without G_{yk} is highly

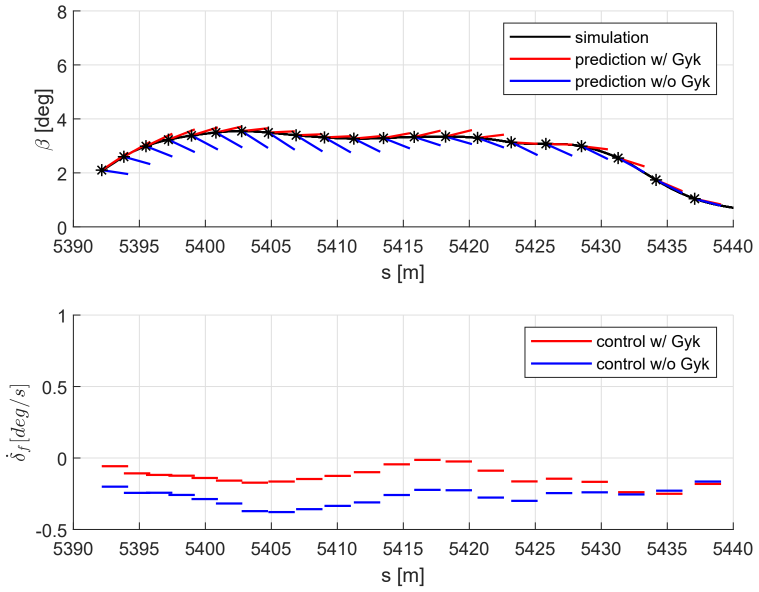


Figure 7.6. The one-step-ahead sideslip predictions and the steering velocity control in *Sim w/* G_{yk}. An offline evaluation of the same quantities computed without G_{yk} is over-imposed for comparison.

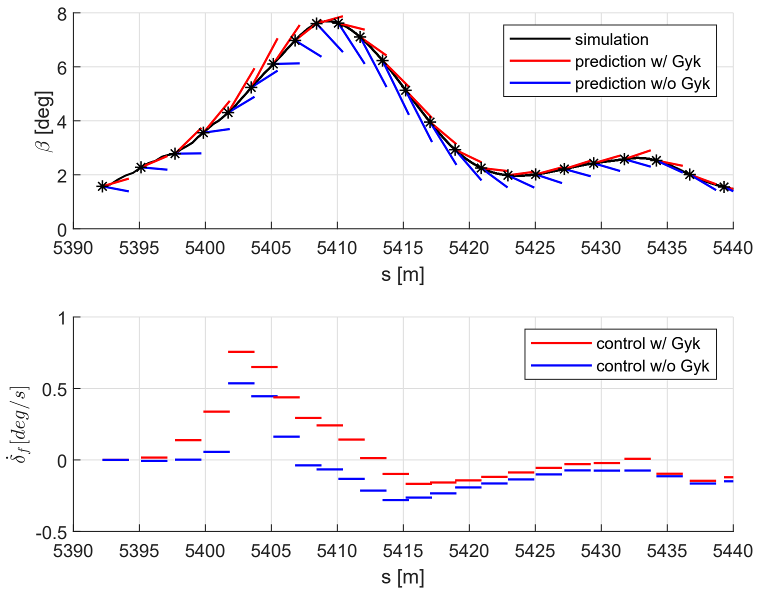


Figure 7.7. The one-step-ahead sideslip predictions and the steering velocity control in *Sim w/o* G_{yk}. An offline evaluation of the same quantities computed with G_{yk} is over-imposed for comparison.

underestimated (blue line), leading to a high side slip condition in the case G_{yk} is not used in closed-loop.

In both cases, without this information, the sideslip evolution is highly underestimated. This fact leads to a curve exit traveled with the expected restrained behaviour (sideslip around 4deg) in Test W/, and to a loss of control of the rear of the car in Test W/O.

8

A LbNMPC for a Virtual Driver in Real-Time

In this Chapter, the definition of a Learning-based NMPC controller for a virtual four-wheel vehicle (depicted in Fig. 8.1) is described. The goal is to enhance the performance of the controller for the lap-time minimization task, by enhancing the model precision. Relying on the model described in Ch. 7, a grey-box prediction model is adopted, characterizing the residual velocity error by Gaussian Processes. To correctly define the model, different kernels have been tested, the dimension of the GP input is reduced through a systematic feature selection strategy, and sparse GP approximations have been exploited, as detailed in Sec. 8.1. The results, illustrated in Sec. 8.2, highlight the effectiveness of the approach, comparing the performance with the NMPC controller described in Ch. 7.

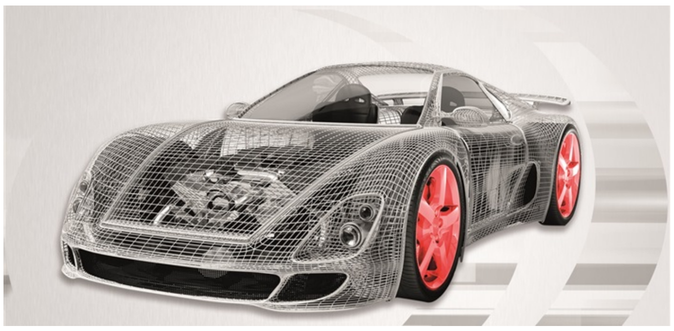


Figure 8.1. Representation of the virtual vehicle in the simulation framework [109].

8.1 Model for Control Synthesis

The dynamics model implemented within the LbNMPC algorithm is based on a nominal characterization of the system compensated for the residual error through Gaussian Processes. The obtained evolution of the system state $\boldsymbol{\xi}_{k+1}$ is characterized as detailed in Sec. 2.4.2, Eq. (2.16), i.e.

$$\boldsymbol{\xi}_{k+1} = \tilde{\boldsymbol{\phi}}(\boldsymbol{\xi}_k, \boldsymbol{u}_k, \boldsymbol{p}_k) + \bar{\boldsymbol{\phi}}(\boldsymbol{\xi}_k, \boldsymbol{u}_k, \boldsymbol{p}_k),$$

where $\tilde{\boldsymbol{\phi}}$ is the nominal characterization, while $\bar{\boldsymbol{\phi}}$ is the GP-based model error estimation. For convenience in the task definition, the model is integrated with respect to the curvilinear abscissa s, hence the nominal dynamics is expressed w.r.t. s and the GPR has been designed to predict the error compensation at the next *space-step*. In the following, the nominal model presented in Ch. 7 is outlined, and the GPR procedure and its results are described.

8.1.1 Nominal Dynamics

The considered nominal dynamics is a four-wheel vehicle model comprehensive of load transfers, gear shift predictions, longitudinal force saturation, and an ellipsoidal tire friction estimate. An extensive description of the model is reported in [50], and the fundamentals are reported hereafter for completeness.

Model Dynamics

The system dynamics is characterized by the following equation of motion of the vehicle center of mass [4]

$$\begin{aligned}
\dot{v}_x &= v_y \dot{\varphi} + \frac{1}{m}\left(\sum_{i,j} F_{x_{i,j}} - F_x^d\right), \\
\dot{v}_y &= -v_x \dot{\varphi} + \frac{1}{m}\left(\sum_{i,j} F_{y_{i,j}}\right), \\
\ddot{\varphi} &= \frac{1}{I_z}\left[a\left(\sum_j F_{y_{f,j}}\right) - b\left(\sum_j F_{y_{r,j}}\right) + c\left(\sum_i F_{x_{i,r}} - \sum_i F_{x_{i,l}}\right)\right].
\end{aligned} \tag{8.1}$$

where the acting forces are given by

- lateral tire forces $F_{y_{i,j}}$, based on Pacejka Magic Formula;
- longitudinal tire forces $F_{x_{i,j}}$, as linearly dependent on the throttle/braking action and saturated on the vertical load;
- drag forces F_i^d, both longitudinal and vertical[1].

[1]The vertical force is not explicitly used in the equation of motion, but impacts on the lateral and

The control model has been formulated in spatial coordinates with respect to the arc length s [96]. The reference trajectory χ is then parameterized in s and defined by its *curvature* $\zeta = \frac{1}{\rho}$, where ρ is the instantaneous curvature radius. Using this approach, it is possible to (i) allow the time to be a minimization variable, and (ii) set the track bounds with respect to the position on the circuit.

In this description, the position of the vehicle within the track is defined with respect to the centerline as the lateral and angular *tracking errors* e_y and e_φ. Moreover, the dynamics can be integrated w.r.t. s using the chain rule, $\forall \dot{s} \neq 0$, as

$$\tilde{\boldsymbol{f}}(\boldsymbol{\xi}(t), \boldsymbol{u}(t), \boldsymbol{p}(t)) = \frac{d\boldsymbol{\xi}}{ds} = \frac{d\boldsymbol{\xi}}{dt}\frac{dt}{ds} = \frac{d\boldsymbol{\xi}}{dt}\frac{1}{\dot{s}} = \frac{\dot{\boldsymbol{\xi}}}{\dot{s}} \tag{8.2}$$

Complete Nominal Model

The resulting state vector is

$$\boldsymbol{\xi} = [v_x,\, v_y,\, \dot{\varphi},\, e_\varphi,\, e_y,\, \delta_f,\, \gamma_t,\, \gamma_b,\, t]^\top, \tag{8.3}$$

where v_x, v_y and $\dot{\varphi}$ are the longitudinal, lateral and yaw velocities, e_φ and e_y are the tracking errors, δ_f, γ_t, and γ_b are the steering, throttle and brake commands, and t is the time. These states are assumed to be fully measured or computed by measurable quantities without noise. The input computed by the algorithm is

$$\boldsymbol{u} = [\dot{\delta}_f,\, \dot{\gamma}_t,\, \dot{\gamma}_b,\, \epsilon_{\text{slip}},\, \epsilon_{\text{err}},\, \epsilon_{\text{gg}}]^\top, \tag{8.4}$$

where $\dot{\delta}_f$, $\dot{\gamma}_t$, $\dot{\gamma}_b$ are the derivatives of the actual input to the vehicle and ϵs are slack variables, whose role is to allow the definition of soft constraints. Finally, the time-varying parameter vector is

$$\boldsymbol{p} = [\zeta, r_{\text{gear}}]^\top \tag{8.5}$$

that contains the current curvature ζ, and gear ratio r_{gear}.

8.1.2 Learning-based Dynamics

Starting from the nominal dynamics previously stated, an additive component characterizing the model error has been obtained through GPR. However, when using the grey-box model as an internal NMPC model, evaluating and optimizing over GPs is highly time-consuming, hence the minimal dimension of the learning-based component has to be developed. In this sense, the most critical quantity to be modeled in our case is the yaw velocity, since it is responsible for the over- or under-steering behavior of the vehicle. Therefore, the learning-based modeling focuses only on yaw velocity, i.e.

$$\bar{\boldsymbol{\phi}} = \left[0, 0, \bar{\phi}^{\dot{\varphi}}(\boldsymbol{x}), 0, \ldots, 0\right]^\top. \tag{8.6}$$

vertical force computation.

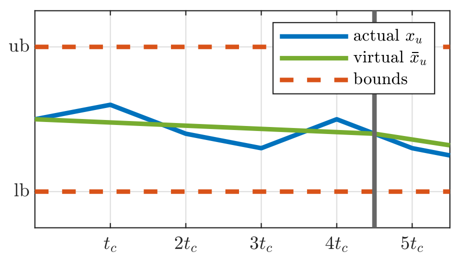


Figure 8.2. Example of the input approximation for data acquisition.

Regression Target

The GP is trained to compensate for the mismatch error between the true model and the nominal model evolution. The model implemented in the LbNMPC algorithm is formulated in spatial coordinates and it is integrated w.r.t. s, hence the quantities for GPR need to be estimated in this domain. In this sense, we introduce the notation $t_k = t(s_k)$, where s_k is the current space coordinate, to indicate the start time of the integration step and $t_k^+ = t(s_k + T_s)$ to indicate the end time of the integration interval, i.e. the time instant reached after the sampling space T_s $[m]$. Due to the integration over space, this is in general different from t_{k+1}, which is the next time instant after a control period T_c $[s]$. Using this notation, the true system state variation is given by the difference after a space-step $\boldsymbol{\Delta\xi}_k = \boldsymbol{\xi}_k^+ - \boldsymbol{\xi}_k$, while the nominal model evolution is given by the integral over space $\boldsymbol{\Delta}\tilde{\boldsymbol{\phi}} = \int_{s_k}^{s_k+T_s} \hat{\boldsymbol{f}}(\boldsymbol{\xi}_k, \boldsymbol{u}_k, \boldsymbol{p}_k)\mathrm{d}s$. Therefore, the regression target, i.e. the error arising due to model mismatches in the space domain, is defined as

$$\boldsymbol{y}_k = \boldsymbol{\Delta\xi}_k - \boldsymbol{\Delta}\tilde{\boldsymbol{\phi}}. \tag{8.7}$$

In particular, we are interested in $y_k^{\dot{\varphi}} = \Delta\dot{\varphi}_k - \boldsymbol{\Delta}\tilde{\boldsymbol{\phi}}_k^{\dot{\varphi}}$.

Training Data Acquisition

The data have been collected online while using the nominal NMPC to drive the vehicle along the track. Accordingly, the NMPC states $\boldsymbol{\xi}_k$ and parameters $\boldsymbol{p}_k$ are stored at each time step t_k. When the car has traveled a space-step T_s, i.e. has reached $s(t) = s_k + T_s$, the system state variation can be computed. However, since the sample rate of the controller is given in the time domain and the length of a prediction step in the space domain, the control action $\boldsymbol{u}$ of the controller (and hence the plant input) can change multiple times during the length of one prediction step[2]. To address this problem, the actual inputs of the vehicle $\boldsymbol{u}_{\text{plant}} = [\delta_f, \gamma_t, \gamma_b]^\top$ are linearly approximated in the predicted dynamics computation, thanks to the fact that the NMPC controls are the derivatives of the actual vehicle inputs. Specifically, denoting the vehicle inputs that are included within the NMPC states as $\boldsymbol{\xi}_{\boldsymbol{u}} = \boldsymbol{u}_{\text{plant}}$, their linear approximation in time domain can be computed as

$$\begin{gathered}\bar{\boldsymbol{u}}_{\text{plant}}(t) = \boldsymbol{\xi}_{\boldsymbol{u}}(s_k) + \bar{\boldsymbol{u}}(s_k)(t - t(s_k)),\\ t \in [t(s_k), t(s_k + T_s))\,,\end{gathered} \tag{8.8}$$

[2]The space step T_s has been designed to be longer than the traveled space in one control period at every feasible velocity.

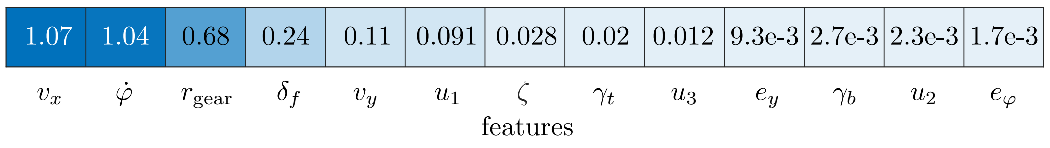


Figure 8.3. Mean feature sensitivity based on five repetitions with 50 inducing points.

where $\bar{\boldsymbol{u}}(s_k)$ is the mean value of the NMPC controls during the considered interval, i.e.

$$\bar{\boldsymbol{u}}(s_k) = \frac{\boldsymbol{\xi}_{\boldsymbol{u}}(s_k + T_s) - \boldsymbol{\xi}_{\boldsymbol{u}}(s_k)}{t(s_k + T_s) - t(s_k)}. \tag{8.9}$$

To illustrate the method, an example on a one-dimensional input is shown in Fig. 8.2.

8.1.3 Feature selection

To reduce the GP prediction time, not all states, inputs and parameters vector $\bar{\boldsymbol{x}}_k = [\bar{x}_k^{(1)}, \cdots, \bar{x}_k^{(n_x+n_u+n_p)}]^\top$ have been considered in the kernel formulation, but only the most important components $\boldsymbol{x}_k \subseteq \bar{\boldsymbol{x}}_k$ for predicting the target $\boldsymbol{y}^{\dot{\varphi}}$, using the permutation-based feature set reduction described in Sec. 3.2.3. In particular, consider a modified dataset $(\pi^l(\bar{X}), \boldsymbol{y}^{\dot{\varphi}})$ where the l-th feature $\bar{\boldsymbol{x}}^{(l)}$ of the data has been shuffled through a random permutation $\pi^l(\cdot)$. By training a GP on the *standard* dataset and another on the *modified* one, it is possible to define the permutation sensitivity as

$$S(l) = \frac{R^2(\bar{X}, \boldsymbol{y}^{\dot{\varphi}}) - R^2(\pi^l(\bar{X}), \boldsymbol{y}^{\dot{\varphi}})}{R^2(\bar{X}, \boldsymbol{y}^{\dot{\varphi}})}, \tag{8.10}$$

where the R^2-score is defined as

$$R^2(\bar{X}, \boldsymbol{y}^{\dot{\varphi}}) = 1 - \frac{\sum_k \left(\bar{\phi}^{\dot{\varphi}}(\bar{X}_k) - y_k^{\dot{\varphi}}\right)^2}{\sum_k (y_k^{\dot{\varphi}})^2}. \tag{8.11}$$

It evaluates the effect on the GP prediction capability of permuting a feature $\bar{\boldsymbol{x}}^{(l)}$, hence a low sensitivity $S(l)$ implies that the l-th feature has a small impact on the prediction performance and can be omitted. By iteratively omitting the least important feature and then training a reduced model, the set of features can be reduced to the most important ones.

In our setup, the whole $\boldsymbol{x} = [\boldsymbol{\xi}, \boldsymbol{u}, \boldsymbol{p}]$ except the time component t of $\boldsymbol{\xi}$ has been used. To obtain reliable results, the analysis averages over five training repetitions, each using an SQ-kernel with 50 inducing points, shown in Fig. 8.3. Based on that, only the features with a sensitivity value greater than 0.05 have been chosen, leading to the features vector

$$\bar{\boldsymbol{x}} = \left[v_x, v_y, \dot{\varphi}, \delta_f, \bar{\dot{\delta}}_f, r_{\text{gear}}\right]. \tag{8.12}$$

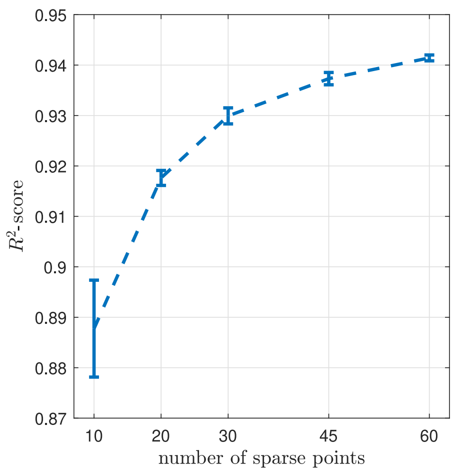


Figure 8.4. Mean and std of R^2-score w.r.t. the no. of inducing points.

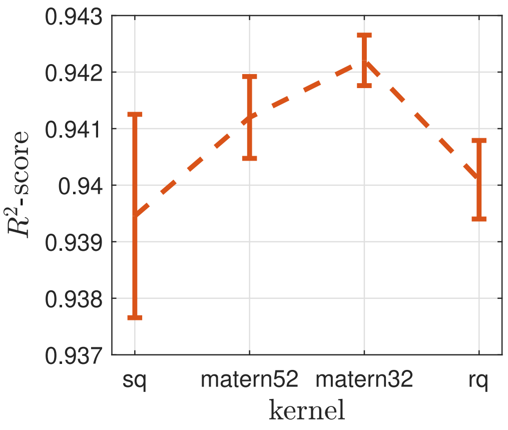


Figure 8.5. Mean and standard deviation of R^2-score with different kernels.

8.1.4 Sparse Approximation

Since the computational load scales with the number of inducing points, it is important to select the most appropriate compromise to minimize the number of points while maximizing the estimation accuracy. In this sense, VFE approximation has been applied as described in Sec. 3.2.1 and an evaluation of the performance while increasing the number of points has been carried out (shown in Fig. 8.4), averaging over five repetitions. The analysis highlighted that, after 45 inducing points, the performance increase tendency is clearly diminishing. Based on this, 50 inducing points have been used for the GP definition.

8.1.5 Kernel Selection

The kernel selection is a key aspect of GPR, since it defines the shape of the covariance function, but it is not straightforward to determine which one is the most suitable in the specific case. Thus, to choose the kernel, a comparison between the GP accuracy obtained using the four ones introduced in Sec 3.1 has been accomplished, averaging over

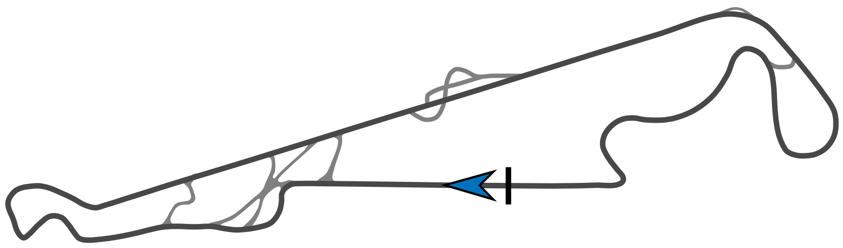


Figure 8.6. Circuit Paul Ricard with highlighted track configuration.

five repetitions. The best performing, both in terms of mean value and variance of the R^2 score, resulted in the Matern 3/2 kernel, as shown in Fig. 8.5.

8.2 Results

The LbNMPC controller designed based on the previously described model has been implemented and tested in a high-fidelity industrial simulation framework, i.e. `VI-CarRealTime` (VI-CRT) [109], on a `Paul Ricard` lap (depicted in Fig. 8.6). An analysis of its prediction capabilities and closed-loop performance has been accomplished, highlighting the enhancements of the proposed solution with respect to the nominal controller described in Ch. 7.

8.2.1 Co-simulation environment

The co-simulation relies on `VI-CRT`, which is specifically designed to reproduce vehicles behaviour for high-performance driving in real time [109]. Its simulation model has 14 degrees of freedom, 6 for the chassis and 2 for each wheel, and it includes comprehensive dynamics of tire, chassis, suspensions, brakes, engine and transmission. The standard demonstration car has been used for simulation.

The co-simulation is performed in `Simulink`, connecting a `VI-CRT` simulation block with the LbNMPC controller. In particular, the controls $\dot{\delta}_f, \dot{\gamma}_t, \dot{\gamma}_b$ computed by the controller are integrated and given as inputs to the simulation block. `VI-CRT` is used to simulate at $f_s^{sim} = 1000\,\text{Hz}$ the dynamics of the vehicle while the control action is updated at $f_s^{ctrl} = 50\,\text{Hz}$.

8.2.2 Controller Setup

The NMPC controller relies on the NLP definition in Eq. (2.3), with the learning-based dicsrete dynamics constraint described in Sec. 2.4.2 and specifically defined in Eq. (2.17). In order to realize a fair comparison with the nominal controller presented in Ch. 7, the nominal NMPC and the proposed LbNMPC have been implemented with the same formulation. However, to obtain the best performances achievable by every model, the weights have been optimized for the different models using a genetic algorithm specifically designed to minimize lap-time [48]. An extensive analysis of the controller formulation and the reasoning behind it are given in Ch. 7, while the main characteristics are reported here for completeness.

The cost function for both the LbNMPC and nominal NMPC has been defined as

$$\begin{aligned} \boldsymbol{h}_k &= [\beta, \gamma_t \cdot \gamma_b, \zeta \cdot \gamma_t, t, \dot{\delta}_f, \dot{\gamma}_t, \dot{\gamma}_b, \epsilon_{\text{slip}}, \epsilon_{\text{err}}, \epsilon_{\text{gg}}]^\top, \\ \boldsymbol{h}_N &= [\beta, \gamma_t \cdot \gamma_b, \zeta \cdot \gamma_t, t, e_y - e_y^{\text{ref}}, \dot{e}_y, e_\varphi - e_\varphi^{\text{ref}} + \beta, \dot{e}_\varphi]^\top. \end{aligned} \tag{8.13}$$

where $\beta = \text{atan}\left(\frac{v_y}{v_x}\right)$ is the sideslip angle of the vehicle and the other components have already been introduced. The main characteristic is the presence of time t in the cost function that supports the computation of a time-minimizing path given the track bounds.

The constraints are defined as

$$\begin{aligned} &\boldsymbol{r}_k = [\delta_f, \gamma_t, \gamma_b, \dot{\delta}_f, \dot{\gamma}_t, \dot{\gamma}_b, \epsilon_{\text{slip}}, \epsilon_{\text{err}}, \epsilon_{\text{gg}}, \beta + \epsilon_{\text{slip}}, e_y + \epsilon_{\text{err}}, (\mu_x \frac{\ddot{x}_{\text{ext}}}{g})^2 + (\mu_y \frac{\ddot{y}_{\text{ext}}}{g})^2 + \epsilon_{\text{gg}}]^\top, \\ &\boldsymbol{r}_N = [\delta_f, \gamma_t, \gamma_b]^\top, \end{aligned} \tag{8.14}$$

where the crucial ones are the second to last, which is used to avoid out-of-the-track trajectories, and the last one, which makes the controller respect the required *gg diagram*, i.e. the maximum combined longitudinal-lateral acceleration that can be induced by the tire forces.

The bounds have been set as

$$\begin{aligned} \underline{\boldsymbol{r}}_k &= [-\frac{\pi}{20}, 0, 0, -2, -10, -100, -2\pi, -10^5, -10^5, -10^5, -\frac{\pi}{15}, -5, 0]^\top, \\ \overline{\boldsymbol{r}}_k &= [+\frac{\pi}{20}, 1, 1, +2, 1.1, +100, +2\pi, +10^5, +10^5, +10^5, +\frac{\pi}{15}, +5, 1.9]^\top, \\ \underline{\boldsymbol{r}}_N &= [-\frac{\pi}{20}, 0, 0]^\top, \ \ \overline{\boldsymbol{r}}_N = [+\frac{\pi}{20}, 1, 1]^\top. \end{aligned} \tag{8.15}$$

Explicit Runge-Kutta 4 integrator has been used, and HPIPM has been selected as QP sparse solver [67] with RTI scheme [61]. The sampling space has been set as $T_s = 2\,\text{m}$ with $N = 140$ shooting points, allowing a prediction of 280 m on track.

8.2.3 Prediction Capabilities

Since the NMPC applies a moving horizon strategy, it does not allow a straightforward comparison between the prediction of the NMPC and the actual closed-loop trajectory, as the latter is given by subsequent input recomputation. Thus, to assess the prediction capabilities of the models, an actual driven trajectory[3] has been compared with the open loop integration of the nominal and the grey-box dynamics applying the actual inputs for the length of a representative prediction horizon. An apparent enhancement in predicting the future trajectory is obtained by the grey-box model, as shown in Fig. 8.7. In particular, the nominal model steers into the chicane much earlier and ends up with a significant offset from the actual trajectory. The grey-box model steers in slightly too early as well, but much later than the nominal one, and follows the actual trajectory much better in the further course. Indeed, the lateral displacement Δe_y with respect to the actual trajectory is more than halved, as shown in Fig. 8.8.

[3]The trajectory computed by VI-CRT is referred to as the actual one.

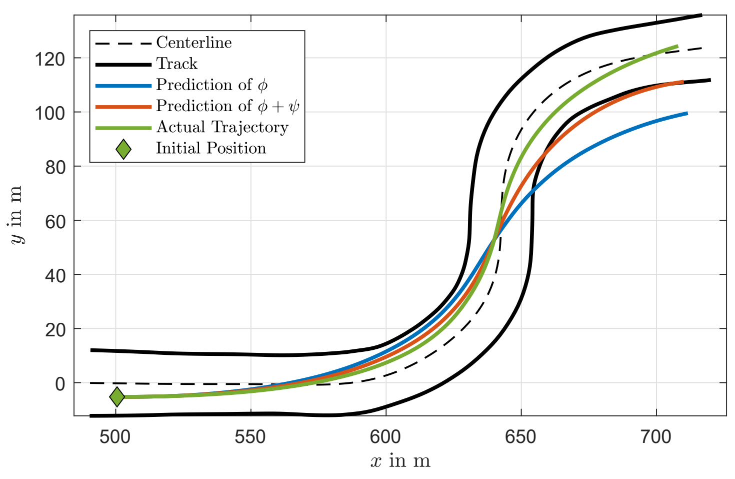


Figure 8.7. Open-loop trajectory prediction for the first chicane.

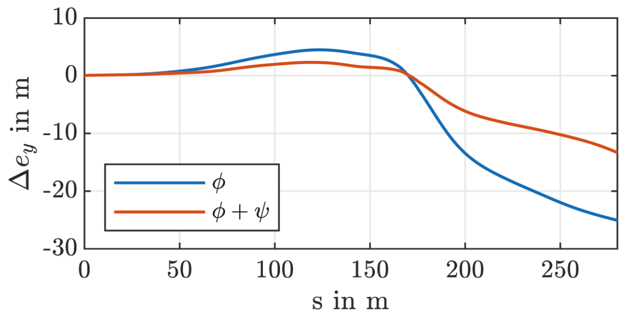


Figure 8.8. Lateral displacement w.r.t. the actual trajectory in the open-loop trajectory prediction for the first chicane.

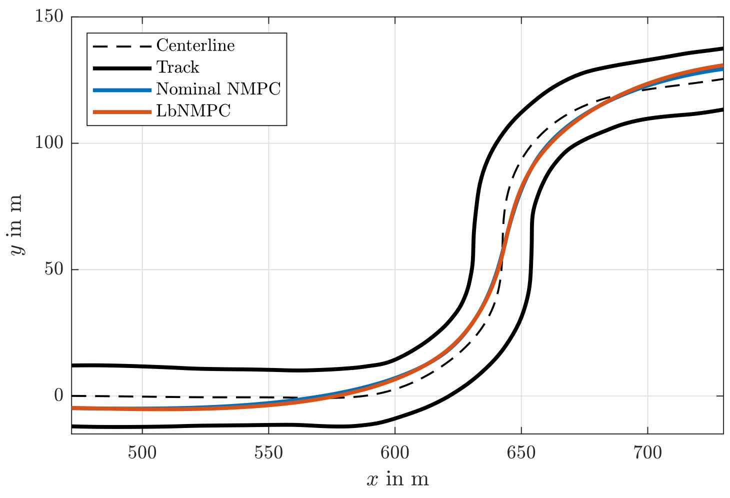


Figure 8.9. Detail of the closed-loop trajectory on the first chicane.

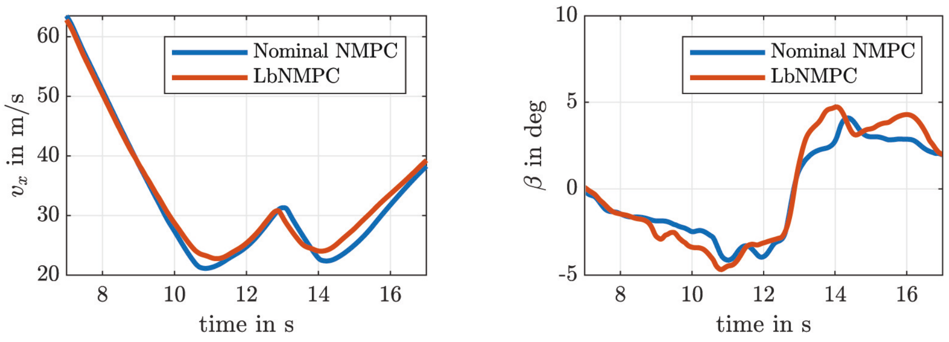


Figure 8.10. Detail of the velocity and sideslip angle during a closed-loop simulation on the first chicane.

8.2.4 Closed-loop Performance

controller	nominal NMPC	LbNMPC
lap time in s	130.98	128.70
average comp. time in ms	12.01	18.76

Table 8.1. Average computation and lap time on the Circuit Paul Ricard.

The LbNMPC controller obtained by using the grey-box model has been used for driving the vehicle in closed-loop, and compared with the nominal NMPC. As reported in Tab. 8.1, the LbNMPC achieves a significantly lower lap time, i.e. a difference of more than 2 seconds, while maintaining real-time capabilities at 50 Hz[4]. In general, the driven trajectory is quite similar, but the LbNMPC is able to travel the curves at a higher velocity and with a slightly higher sideslip angle. A plot of these quantities in the first chicane is shown in Fig. 8.9 and 8.10, where this behavior is clearly visible.

[4]The computational time has been obtained on a PC mounting an Intel Core i7-7700 CPU @ 3.60GHz.

9
A LbNMPC for a Real Go-Kart in Real-Time

In this Chapter, the definition of a Learning-based NMPC controller for an experimental go-kart (depicted in Fig. 9.1) is described. The goal is to obtain a controller that is able to drive the vehicle on a circuit, while also computing trajectory and velocity profile, relying only on a data-driven model. Indeed, a pure black-box prediction model is adopted, as described in Sec. 9.1, characterizing the accelerations by Gaussian Processes, exploiting sparse GP approximations, both offline and online to obtain a real-time controller. Sec. 9.2 presents both a preliminary feasibility study in simulation to tune the most relevant hyperparameters and the obtained results on the real go-kart vehicle in the experimental setup.

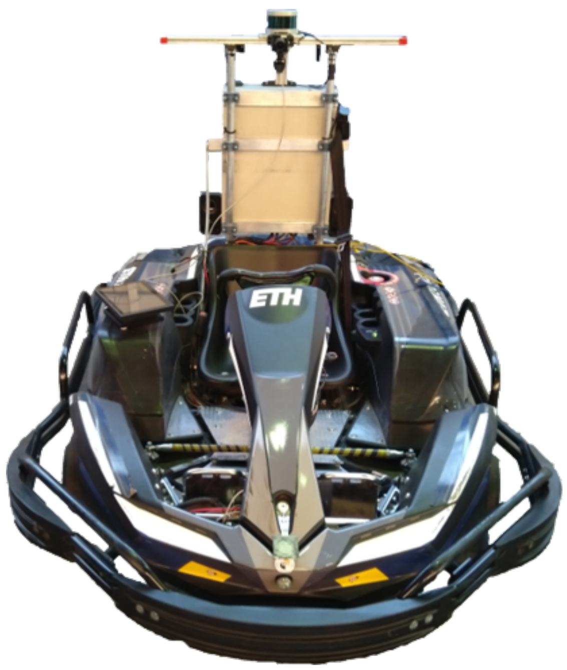


Figure 9.1. The experimental go-kart platform.

9.1 Model for Control Synthesis

Effectively characterizing 4-wheel vehicles is a challenging task, and several models have been proposed, ranging from complex dynamics models to simple bicycle models. Each formulation has advantages and disadvantages in terms of computational resources and prediction capabilities. Moreover, a precise characterization of car dynamics involves proper identification of tires, drag and friction forces, suspensions, steering systems, etc., which may require expensive and time-consuming specific measurements. Therefore, the exploitation of IMU and localization data to infer the vehicle dynamics could remove the necessity of these explicit analyses. The following dynamics are completely data-driven, i.e. both the direct lateral acceleration $\dot{v}_y$ and yaw acceleration $\ddot{\theta}$ are entirely characterized by two different Gaussian Processes. The longitudinal command of the go-kart is based directly on the requested longitudinal acceleration, i.e. the desired acceleration is transmitted to the control system of the electric motors on the wheels that is responsible for actuation, so the characterization of the longitudinal acceleration is not needed. Both simulation and experimental scenarios are considered, in order to evaluate the strategy also in absence of noise and actuator dynamics. On the other hand, the spatial domain reformulation has been mathematically described, characterizing the velocity rotations, in order to define a time-minimization strategy. Finally, a previously derived nominal dynamics model is presented for comparison.

9.1.1 Black-box Dynamics Model

The black-box dynamics model has been obtained as detailed in Ch. 3 and applying the Subset of Data (SoD) and Nearest Neighbor (NN) approximations described in Sec. 3.2. In particular, the go-kart velocity and actual control state components (see Sec. 9.1.3) have been used as regressors for the GP training, i.e. $\boldsymbol{x} = [v_x, v_y, \dot{\theta}, \gamma, \beta, \tau_v]$, and a minimum $\sigma_n = 0.15$ has been imposed to enhance regularization properties, both for simulation and experimental scenarios. The GP has then been trained in batch, using a batch size of 100 data points, for 400 epochs and a learning rate of 0.001. Furthermore, SoD reduction has been applied with threshold $1.0\sigma_n^2$. To check the prediction capabilities of the method that will be employed online, the NN strategy with $T_p^{\dot{v}_y} = 30$ and $T_p^{\ddot{\theta}} = 50$ for $\dot{v}_y$ and $\ddot{\theta}$, respectively, has been applied on the obtained GP. Such dimensions, in fact, allow balancing between the computational burden and prediction capabilities of the controller. Moreover, the nominal model (described in Sec. 9.1.4) has been tested on the same data.

Simulation

Due to the simulation environment structure, a nominal controller-driven run, i.e. using the nominal model within the NMPC controller, was recorded and used for black-box model training. As the simulation serves as a preliminary stage for the experimental setup and manual driving was not possible due to hardware limitations, we used a closed-loop control action to mimic human driving. This step allows for verifying the capability

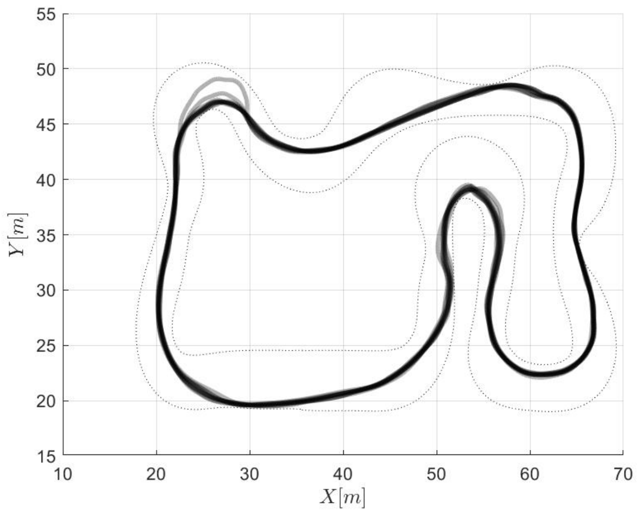


Figure 9.2. Trajectories of the data acquired for training.

of the GP to fit the acceleration data in a fully controlled environment and comparing the accuracy with respect to the nominal model. The sparse GP after SoD reduction contains 53 inducing points for both $\dot{v}_y$ and $\ddot{\theta}$, hence the local approximation with T_p points precision is almost similar. The Root Mean Squared Error (RMSE) of both lateral and yaw accelerations for nominal and black-box dynamics resulted to be very similar (reported in Tab. 9.1), hence supporting the possibility to use the developed strategy for the real platform.

Experimental

To excite the highly nonlinear and complex dynamics of the system, 10 human-driven laps close to the vehicle's maximum acceleration were performed. A total number of around 5000 data points was recorded, comprising IMU, localization, and velocity data, along with steering position and longitudinal acceleration commands. A depiction of the acquired data in terms of path, velocity profiles and used commands is reported in Fig. 9.2-9.4. The travelled trajectories are quite similar, as expected in a track driving task, but the different laps allow to give sufficient variety in the training data. Additionally, it is worth mentioning that GPs do not include the curvilinear abscissa as a predictor variable. Instead, they are capable of estimating accelerations by exploiting the system's velocities and inputs, which separates the vehicle's dynamic behavior from its position on the track. As a result, the various turns on the track present different operating conditions for the

Table 9.1. RMSE of lateral and yaw accelerations for nominal and black-box models (both using SoD and NN reductions) in simulation.

model	$\dot{v}_y$ $[m/s^2]$	$\ddot{\theta}$ $[rad/s^2]$
nominal	0.62	1.10
black-box SoD	0.44	0.73
black-box NN	0.61	0.74

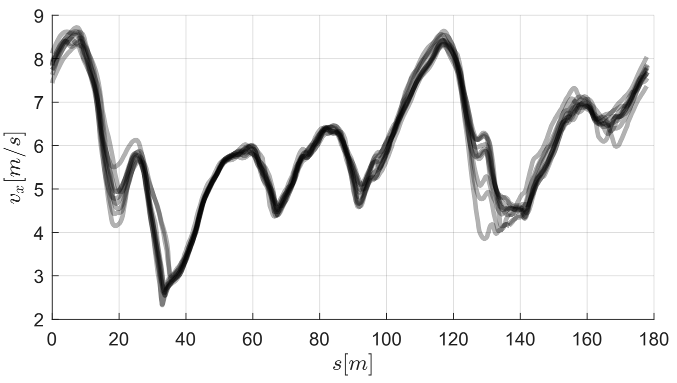


Figure 9.3. Longitudinal velocity of the data acquired for training.

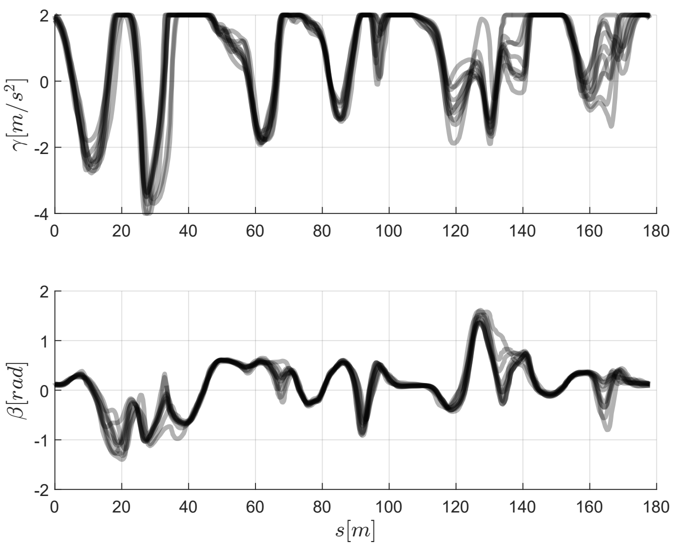


Figure 9.4. Commands given to the go-kart of the data acquired for training.

system, making the 10 laps a sufficiently information-rich training dataset. The resulting data, acquired at different frequencies depending on the sensor, were interpolated and the IMU and velocity data were filtered through a Forward Backward Kalman Filter [112], resulting in aligned and smooth acceleration data for the GP training, shown in Fig. 9.5. SoD reduction led to sparse GPs of 98 and 114 points for $\dot{v}_y$ and $\ddot{\theta}$, respectively. The resulting accelerations estimates, i.e. the mean of the GP for lateral $\psi_{\dot{v}_y}$ and yaw $\psi_{\ddot{\theta}}$ accelerations, are shown in Fig. 9.6 and 9.7, together with the real accelerations on the go-kart. Both the ability of the model to fit the data and the approximation given by the reduction are clearly visible. In particular, the NN strategy on a 6-dimensional space is not always effective and may generate some spikes. At the same time, note that also the nominal model is not able to always describe correctly the acceleration values, but mostly the tendency. However, the Root Mean Squared Error (RMSE) of lateral and yaw accelerations, reported in Tab. 9.2, is still comparable with the nominal dynamics ones.

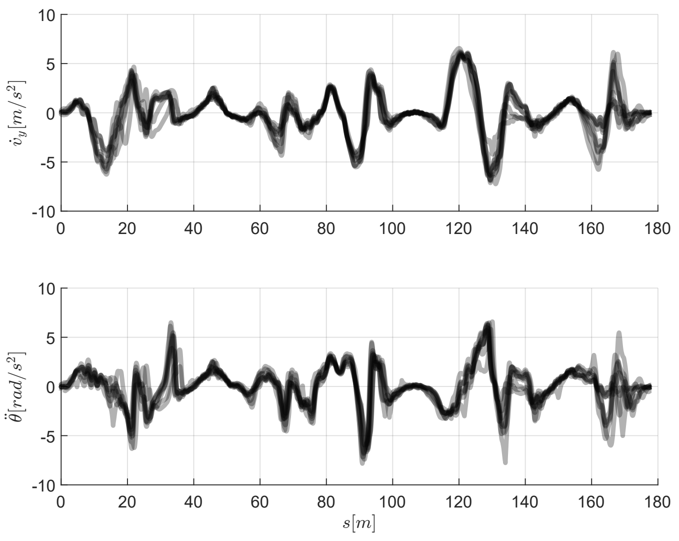


Figure 9.5. Direct lateral acceleration and yaw acceleration of the data acquired for training.

Table 9.2. RMSE of lateral and yaw accelerations for nominal and black-box models (both using SoD and NN reductions) on the real go-kart.

model	$\dot{v}_y$ $[m/s^2]$	$\ddot{\theta}$ $[rad/s^2]$
nominal	1.68	1.28
black-box SoD	1.03	1.22
black-box NN	1.95	1.81

9.1.2 Spatial Reformulation

The complete model is comprehensive of the yaw e_θ and lateral e_y errors with respect to the track centerline and the integration has been applied in spatial coordinates with respect to the arc length s along the track, previously presented in [50, 105]. This strategy allows for using time as a minimization variable, setting the spatial constraints, and eliminating the dependency on the velocity in the trajectory reference for the controller. The kinematic relation between velocities and the angular and lateral errors can be expressed as

$$\begin{aligned} \dot{e}_\theta &= \dot{\theta} - \zeta\ \dot{s}, \\ \dot{e}_y &= v_x\ \sin(e_\theta) + v_y\ \cos(e_\theta), \end{aligned} \tag{9.1}$$

where $v_x, v_y, \dot{\theta}$ are the longitudinal, lateral, and yaw velocities, respectively, $\zeta = \frac{1}{\rho}$ is the trajectory curvature, and $\dot{s} = \frac{1}{1-\zeta\ e_y}\ (\dot{x}\ \cos(e_\theta) - \dot{y}\ \sin(e_\theta))$.

9.1.3 Complete Model

The problem has been formulated in *velocity form*, i.e., the inputs are the derivatives of the actual go-kart commands. Hence, the state has been defined as

$$\boldsymbol{\xi} = [v_x,\ v_y,\ \dot{\theta},\ e_\theta,\ e_y,\ \gamma,\ \beta,\ \tau_v,\ t]^\top, \tag{9.2}$$

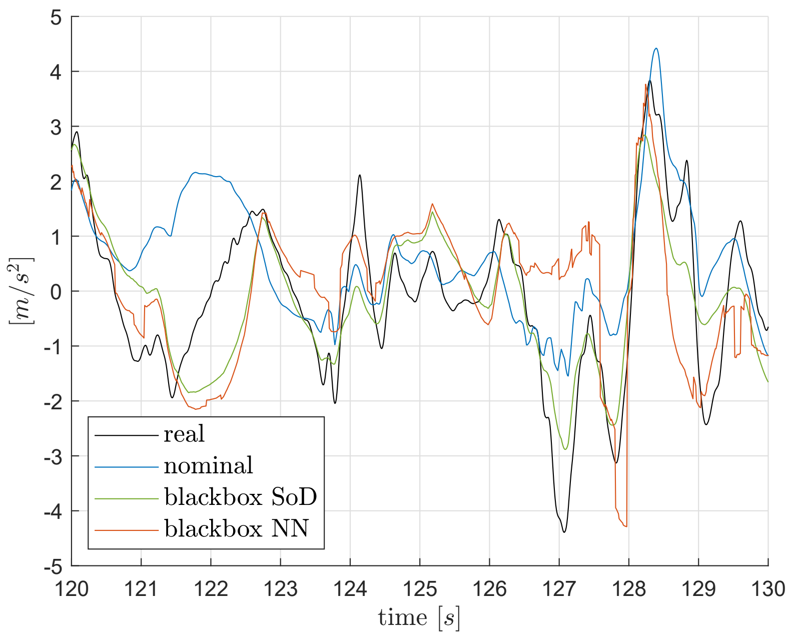


Figure 9.6. Comparison of nominal and black-box models (both with SoD and NN reductions) with respect to the real *direct* lateral acceleration $\dot{v}_y$ on the real go-kart.

where γ, β, and τ_v are the actual commands to the go-kart, i.e. the desired longitudinal acceleration, the steering angle, and the torque vectoring component, respectively, and t is the time, while the input vector is

$$\boldsymbol{u} = [\dot{\gamma},\ \dot{\beta},\ \dot{\tau}_v,\ \eta]^\top, \tag{9.3}$$

where $\dot{\gamma}$, $\dot{\beta}$, $\dot{\tau}_v$ are the derivatives of the actual commands, and η is a slack variable needed for implementing a soft constraint [110]. The resulting dynamics are expressed by

$$\dot{\boldsymbol{\xi}} = [\gamma,\ \psi_{\ddot{y}},\ \psi_{\ddot{\theta}},\ \dot{e}_\theta,\ \dot{e}_y,\ \dot{\gamma},\ \dot{\beta},\ \dot{\tau}_v,\ 1]^\top. \tag{9.4}$$

Finally, the state vector x is differentiated w.r.t s using the *chain rule* as

$$\boldsymbol{\xi}' = \frac{d\boldsymbol{\xi}}{ds} = \frac{d\boldsymbol{\xi}}{dt}\frac{dt}{ds} = \frac{d\boldsymbol{\xi}}{dt}\frac{1}{\dot{s}} = \frac{\dot{\boldsymbol{\xi}}}{\dot{s}},\ \ \forall \dot{s} \neq 0, \tag{9.5}$$

thus defining the integration with respect to the curvilinear abscissa s.

9.1.4 Nominal Model

A complete dynamics model was already available and has been used for comparison. It is based on a three-wheel vehicle and includes different important dynamics, such as

- an identified Pacejka's magic formula for lateral tire forces,
- an approximation of the adherence ellipsoid for combined tire behavior,
- longitudinal load transfer dynamics.

In particular, the Pacejka's formulas for front and rear wheels are the core of the model,

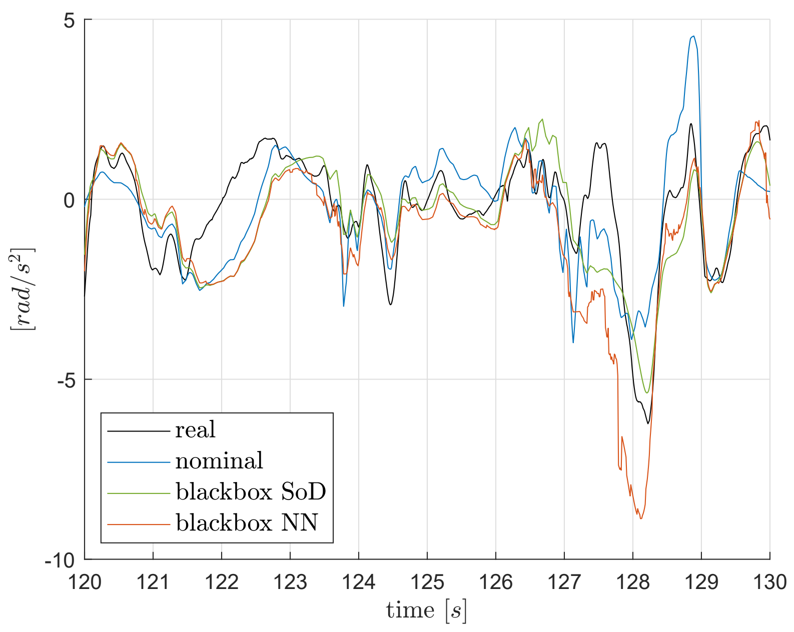


Figure 9.7. Comparison of nominal and black-box models (both with SoD and NN reductions) with respect to the real yaw acceleration $\ddot{\theta}$.

as they require specific tests to correctly identify the shape of the curves. Further details on the formulation can be found in [113].

9.2 Results

In this section, the results obtained are presented: a preliminary simulative trial allows the viability of the procedure to be analyzed in a fully controlled environment, while the experimental scenario illustrates the actual validation of the strategy. The controller must determine both the trajectory and the velocity profile in the prediction horizon, knowing the track bounds, while the time-minimization is enforced in the cost function. This formulation allows adapting to unexpected behaviors of the vehicle since the desired path and speed are recomputed at every time step. The prediction capabilities are also investigated, considering the one-step ahead velocity prediction error, i.e. $\hat{\boldsymbol{e}}_{\dot{q}} = \dot{\boldsymbol{q}} - \tilde{\dot{\boldsymbol{q}}}$, where $\tilde{\dot{\boldsymbol{q}}}$ is the one-step-ahead velocity prediction and $\dot{q}$ is the real velocity of the go-kart. To fulfill hard real-time requirements, the Subset of Data and Nearest Neighbors local approximations described in Sec. 3.2 have been adopted.

Through the whole section, the proposed LbNMPC based on black-box modeling and an NMPC based on nominal dynamics are compared, referring to the two controllers as *black-box* and *nominal* strategy, respectively.

9.2.1 Controller Setup

The NMPC controller relies on the NLP definition in Eq. (2.3), with the learning-based continuous dynamics constraint described in Sec. 2.4.3 and specifically defined in Eq.

(2.23). The cost function for the LbNMPC is defined as

$$\begin{aligned} \boldsymbol{h}_j(\boldsymbol{\xi}_j, \boldsymbol{u}_j) &= [\dot{\gamma}, \dot{\tau}_v, \dot{\beta}, \eta]^\top, \\ \boldsymbol{h}_N(\boldsymbol{\xi}_N) &= [t, e_\psi - e_\psi^{\text{ref}}, e_y - e_y^{\text{ref}}]^\top. \end{aligned} \tag{9.6}$$

The input terms allow a smooth control action, while the objective variable time t supports the computation of a time-minimizing path and its weight can be used to tune the importance of the lap time performance. The slack variable η is used to define the soft constraint on the track bounds violations. The terminal cost terms related to errors e_θ and e_y are adopted to enforce reasonable dynamics of the vehicle at the end of the prediction horizon. The trajectory to be used for this task has been precomputed by minimizing the path curvature. To maintain the same configuration in all the tests and to allow fair comparisons of the results in the different cases, the controller tuning has been accomplished through an empirical analysis, resulting in the following weight matrices

$$\begin{aligned} W &= diag([2 \cdot 10^{-3}, 5 \cdot 10^{-2}, 10^{-2}, 5 \cdot 10^{1}]), \\ W_N &= diag([10^{-1}, 10^{3}, 10^{2}]). \end{aligned} \tag{9.7}$$

The constraints are defined as

$$\begin{aligned} \boldsymbol{r}_j &= [v_x, e_\theta, \gamma, \beta, \tau_v, \dot{\gamma}, \dot{\tau}_v, \dot{\beta}, \eta, e_y + \eta]^\top, \\ \boldsymbol{r}_N &= [v_x, e_\theta, \gamma, \beta, \tau_v, e_y + \eta]^\top, \end{aligned} \tag{9.8}$$

where the constraints on v_x and e_θ are used to exclude singularities in the model kinematics and numerical issues, the ones on the states γ, β and τ_v are the integration of the computed inputs and represent intrinsic bounds of the actual vehicle commands, while those on $\dot{\gamma}$, $\dot{\beta}$ and $\dot{\tau}_v$ are added in order to improve the smoothness of the computed inputs and can be used to tune the aggressivity of the NMPC driving commands. Finally, the constraint on η allows setting a maximum track bound exceed and the one on e_y defines the width of the track. The bounds are hence defined as

$$\begin{aligned} \underline{\boldsymbol{r}} &= [2.5, -\pi/2, -4.2, -\pi/2, -1.7, -10^3, -10^2, -10^1, -5, e_y^{lb}]^\top, \\ \overline{\boldsymbol{r}} &= [15, +\pi/2, +2, +\pi/2, +1.7, +10^3, +10^2, +10^1, +5, e_y^{ub}]^\top, \end{aligned} \tag{9.9}$$

where e_y^{lb} and e_y^{ub} are the track bounds updated online based on the current position, slightly reduced by $0.5m$ to effectively implement the soft constraint.

HPIPM [67] has been used within LbMATMPC as QP sparse solver and Real-Time Iteration (RTI) scheme [69] has been adopted. By using the RTI scheme, local convergence of the algorithm is ensured by RTI guarantees on contractivity and boundness of the loss of optimality compared to optimal feedback control [68, 69]. The integration step has been set as $T_s = 0.3m$ for $N = 80$ steps, allowing a prediction horizon of $24m$, and a reduction of the shooting points has been obtained through a non-uniform integration grid of $r = 33$ points distributed as $G = \{1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 16, 18, 20, 23, 26, 29, 32, 35, 38, 41, 44, 47, 50, 53, 56, 59, 62, 65, 68, 71, 74, 77, 80\}$. The grid allows accurate control actions to be computed at the beginning of the horizon while reducing the computational burden and enabling less precision for subsequent points in the future. The technique, with the number of data points chosen for sparse GPs, i.e. $T_{\dot{v}_y}^{NN} = 30$ and $T_{\ddot{\theta}}^{NN} = 50$ for $\dot{v}_y$ and $\ddot{\theta}$,

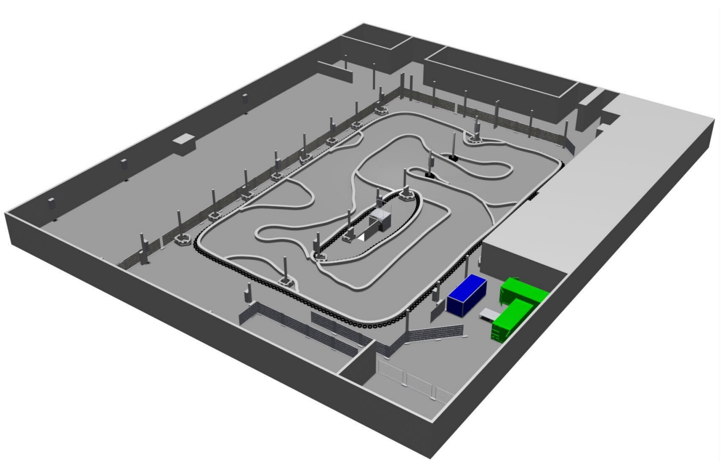


Figure 9.8. The simulation environment in Gazebo.

respectively, allows the controller to run at a control frequency $f_c = 20Hz$ on the real go-kart.

In such complex experimental scenarios, theoretical guarantees on the stability of the algorithm are hardly achievable. Moreover, the adoption of a black-box model identified in the continuous domain and then discretized further complicates the mathematical description. However, our approach *empirically* supports the recursive feasibility of NLP by adopting the following expedients:

1. a soft constraint on e_y, which is the only constraint where the model uncertainty is emphasized, that, by marginally reducing the bound value, allows for a collision-free trajectory;

2. a sufficiently long prediction horizon that includes the next curve, allowing the controller to compute a velocity profile compatible with the maximum lateral acceleration and the track boundaries;

3. a final reference with high weights on lateral and angular errors e_y and e_ψ, exploiting the well-known effect of the final cost to lead to conservative behavior.

9.2.2 Preliminary simulation study

A preliminary study in simulation has been accomplished in order to validate the feasibility of the developed strategy and identify a tuning set of the hyperparameters, to be used as a starting point for the experimental scenario in a completely controllable environment. In particular, the control algorithm is forced to fulfill real-time requirements while no sensor noise and actuation delays are present, allowing setting the GP number of point T^{NN}, the prediction horizon length N, the integration step T_s and the grid G.

A previously developed simulation environment [114], [115], based on *C++* and Gazebo within *ROS* (Robot Operating System) [116], has been used for this phase (see Fig. 9.8). It uses the same communication protocol and reproduces the same channels as the real go-kart, allowing for a practical test of the controller as it would be in the experimental scenario.

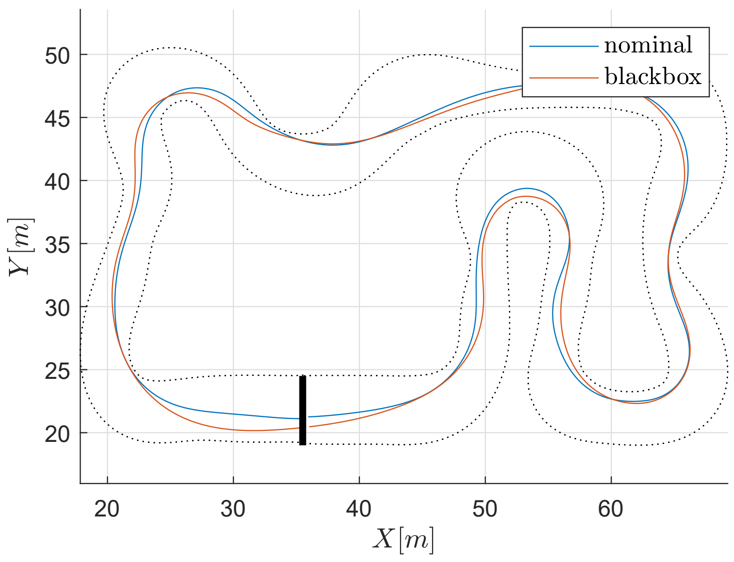


Figure 9.9. Comparison of trajectory obtained by the (Lb)NMPC using the nominal and black-box models in simulation.

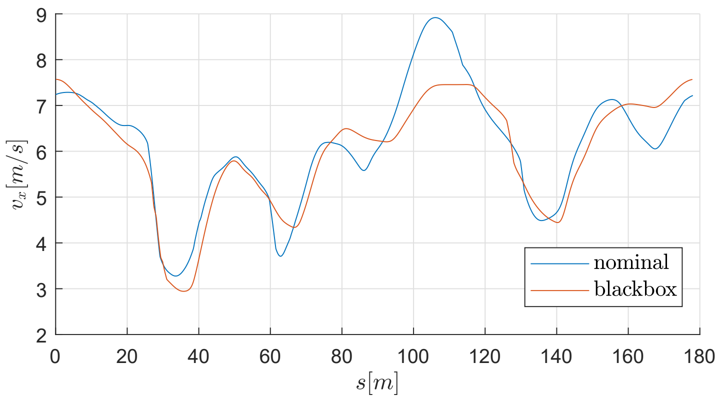


Figure 9.10. Comparison of longitudinal velocity obtained by the (Lb)NMPC using the nominal and black-box models in simulation.

The LbNMPC controller has been run in the simulation framework, adopting the same configuration that will be tested on the real go-kart, except for the black-box prediction model within. As expected, the black-box model is fairly precise in predicting online the one-step-ahead velocities for both v_y and $\dot{\theta}$, as shown in Tab. 9.3. In particular, the error is in the same order of magnitude as the nominal model, counting half the error for v_y and slightly lower for $\dot{\theta}$, confirming that the GPs are actually describing the system dynamics, in average, accurately. This fact is reflected in a superior closed-loop behavior of the black-box strategy in some curves, but yet worse in others. Indeed, the driven paths are quite similar (see Fig. 9.9), while the turns are traveled at different velocities (shown in Fig. 9.10), leading to nearly the same lap-time overall, i.e. 26.80s for black-box model and 26.55s for the nominal model. The obtained behaviour is relevant in the perspective of using the scheme in the experimental environment since it illustrates the validity of the proposed approach.

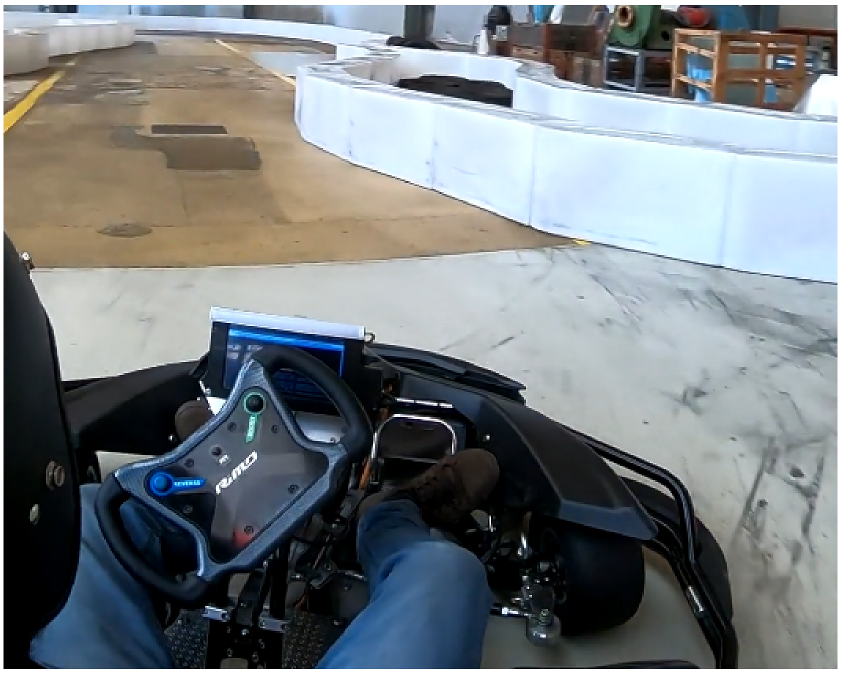


Figure 9.11. The real go-kart platform driving autonomously.

9.2.3 Experimental Results

The LbNMPC controller based on the model presented in Sec. 9.1.1 has been implemented and tested on the real go-kart platform. An indoor 180m long track has been used as a test bench, comparing the performance obtained using the black-box modeling with respect to the nominal ones. In the experimental scenario relevant approximations for the data-driven modeling have been observed, therefore reducing prediction capabilities. In particular, the higher complexity of the learned functions leads to higher errors when applying the NN local approximation (as reported in Tab. 9.2). In fact, the real scenario pose highly complex concerns due to the intrinsic uncertainty in the sensor readings, the needed filtering of the signals, and the slight environment changes (e.g., temperature, humidity, etc.) in different tests. The mean time for solving the LbNMPC problem resulted in $28.53ms$, while it resulted in $10.21ms$ for the nominal NMPC one.

Go-kart Platform and Experimental Environment

The go-kart is based on a RiMO SiNUS iON electric rear wheel-driven go-kart platform. The vehicle is equipped with different sensors for state estimation and localization such

Table 9.3. RMSE of online one-step-ahead velocity predictions $\hat{e}_{\dot{q}}$ for nominal and black-box model in simulation.

model	v_y $[m/s]$	$\dot{\theta}$ $[rad/s]$
nominal	$4.03 \cdot 10^{-2}$	$4.89 \cdot 10^{-2}$
black-box	$2.04 \cdot 10^{-2}$	$4.57 \cdot 10^{-2}$

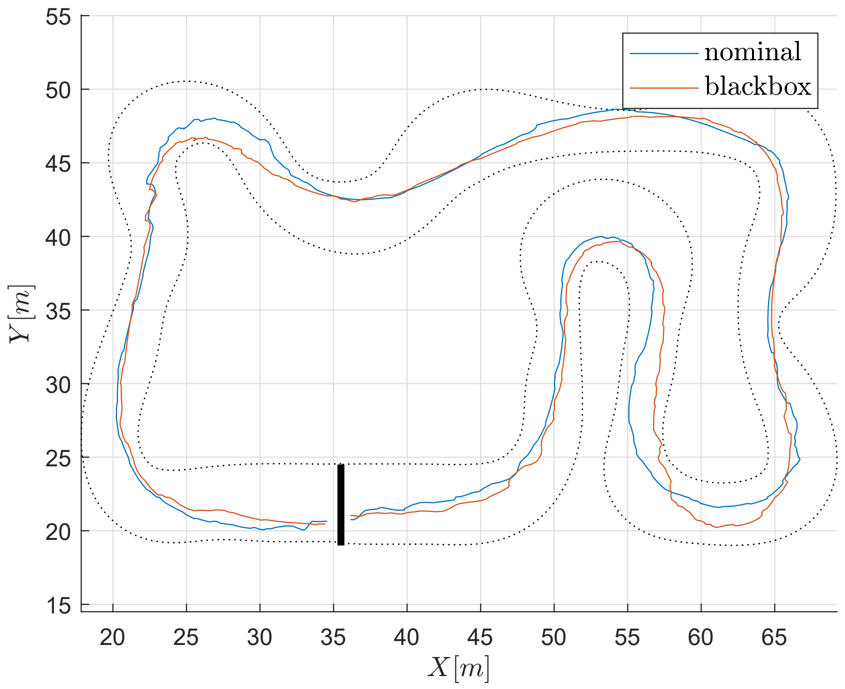


Figure 9.12. Comparison of trajectory obtained by the (Lb)NMPC using the nominal and black-box models on the real go-kart.

as IMU and LiDAR. It mounts a custom computer on the back, i.e. an Intel Xeon D-1540 @2.00Ghz CPU with 8 cores. The computer interfaces with the sensors and the actuators, then it processes the data and runs the localization, mapping, and state estimation modules while executing the controller. The localization system is based on a complementary filter that merges the accelerations from the IMU and the Lidar measurements, matching the sensed nearby obstacles to obtain the position of the go-kart on the track. The track is a 180m long custom-made path defined by safety barriers and tires inside an industrial hangar. The communication between the different frameworks is done using ROS which gives a structured communication layer above the Ubuntu operating system.

Results on Track

The LbNMPC presented in the previous section has been employed to drive the go-kart along the track, depicted in Fig. 9.12, obtaining comparable results with respect to the nominal strategy. Specifically, the driven trajectories resulted very similarly over most of the track, with slight differences in specific parts. The path, velocity, sideslip and controls are shown in Fig. 9.12-9.15. In particular, the black-box approach travels the bottom-left

Table 9.4. RMSE of online one-step-ahead velocity predictions $\hat{e}_{\dot{q}}$ for nominal and black-box model on real go-kart.

model	$v_y\ [m/s]$	$\dot{\theta}\ [rad/s]$
nominal	$7.63 \cdot 10^{-2}$	$5.97 \cdot 10^{-2}$
black-box	$9.80 \cdot 10^{-2}$	$8.05 \cdot 10^{-2}$

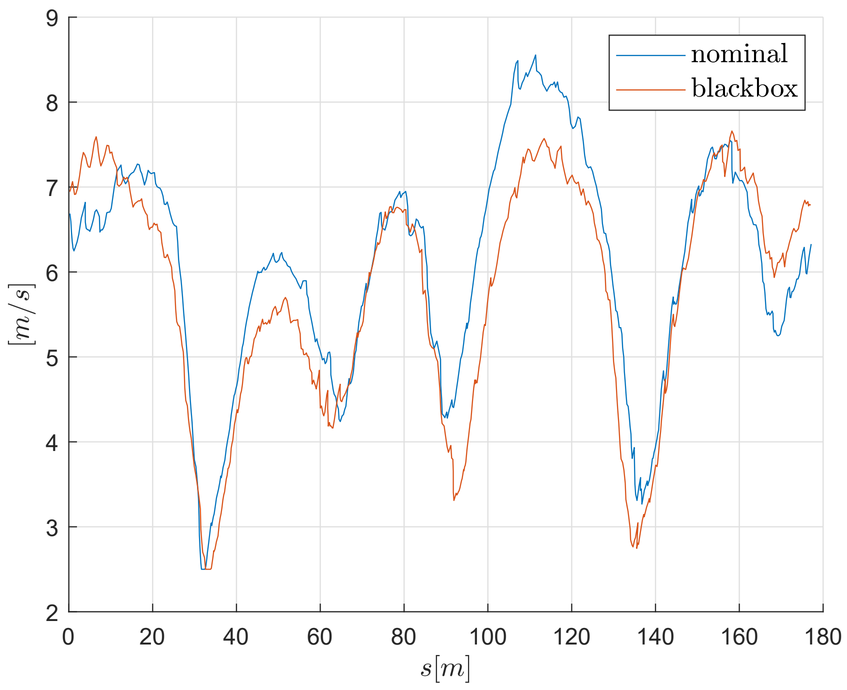


Figure 9.13. Comparison of longitudinal velocity obtained by the (Lb)NMPC using the nominal and black-box models on the real go-kart.

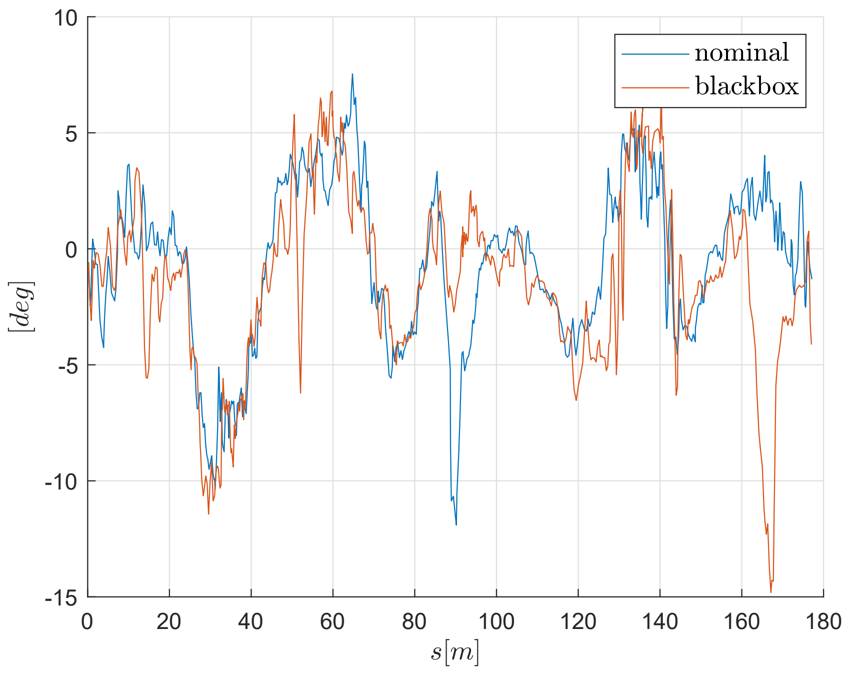


Figure 9.14. Comparison of sideslip obtained by the (Lb)NMPC using the nominal and black-box models on the real go-kart.

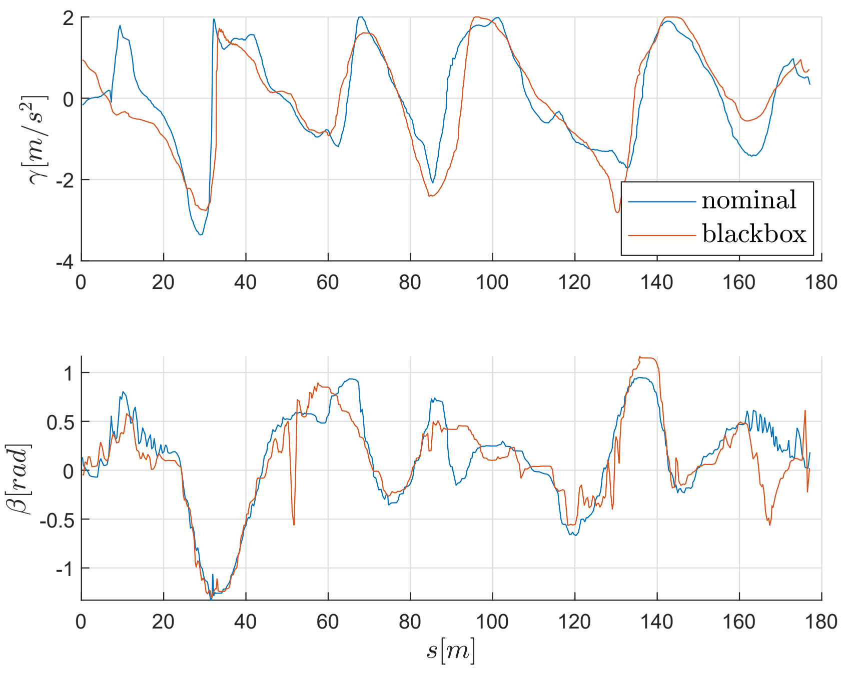


Figure 9.15. Comparison of controls computed by the (Lb)NMPC using the nominal and black-box models on the real go-kart.

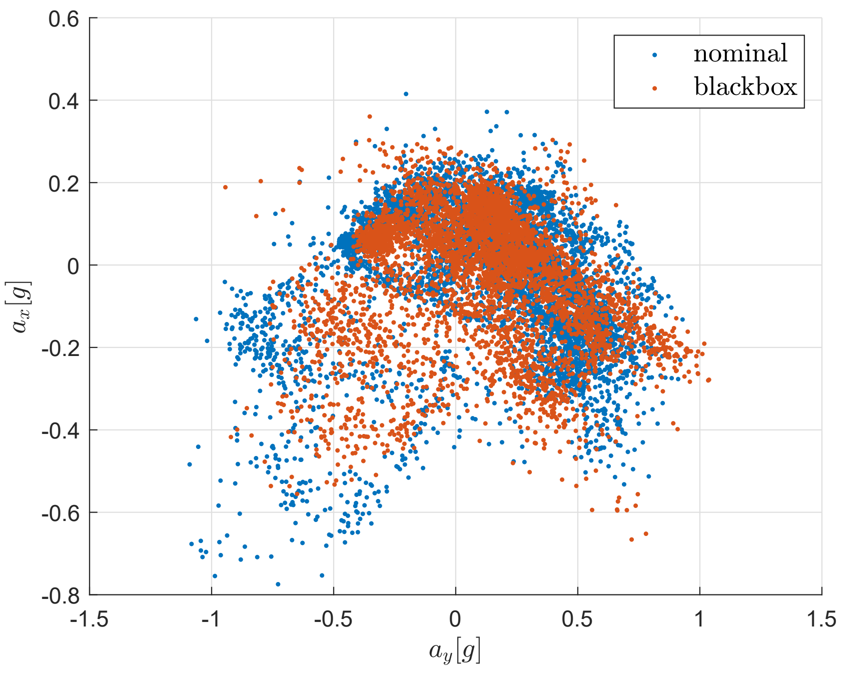


Figure 9.16. Comparison of gg-diagram (longitudinal vs lateral accelerations) obtained by the (Lb)NMPC using the nominal and black-box models on the real go-kart.

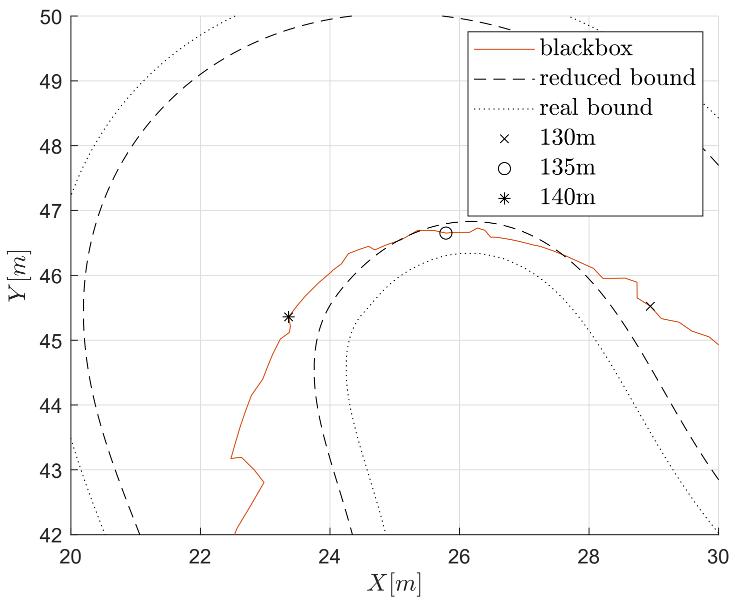


Figure 9.17. Detail of the trajectory travelled by the LbNMPC controller, highlighting the reduced bound on lateral error during the top left curve.

turn (at $160 - 180m$) with higher velocity, controlling a high sideslip, gaining speed over the whole bottom straight. The two schemes obtain similar performance at the top-left corner ($120 - 150m$) and between the first and the second U turn (at $40 - 60m$), with different strategies. In both cases, the black-box controller is cutting the curve more than the nominal, travelling less distance but slightly lowering the speed at the curve center. On the other hand, in the top-right corner (at $80 - 100m$) the center of the turn is travelled at a lower velocity by the black box strategy, making the whole top straight with a speed gap with respect to the nominal controller. This behaviour results in a lap time of $29.3s$ for black-box, that is 3.75% slower than the nominal lap time, i.e. $28.2s$. Interestingly, both models resulted in similar behavior under sideslip conditions (Fig. 9.14), with the black-box model allowing to stabilize the vehicle after an oversteer with a side slip of $15deg$ during the exit of the bottom-left curve, at $160 - 180m$. The actual commands sent to the go-kart using the different modeling, shown in Fig. 9.15, are mostly consistent, except for the counter-steering actions at $50m$ and $170m$ for the black-box controller and at $90m$ for the nominal one. The maximum accelerations are similar for both control strategies (Fig. 9.16), with peaks of $\pm 1g$ for lateral and $[+0.2, -0.6]g$ for longitudinal (i.e. the longitudinal maximum accelerations given by motor and braking system) respectively. In this framework, the nominal model is more precise on average in predicting velocities with respect to the black-box one, i.e. velocity errors around 25% lower, as reported in table 9.4: indeed, the GP approximations needed to obtain a real-time controller led to a relevant reduction in the prediction capabilities. However, the obtained system representation is sufficiently informative to effectively complete the driving task on the experimental go-kart. It is expected that, it could be further improved by providing more computing resources.

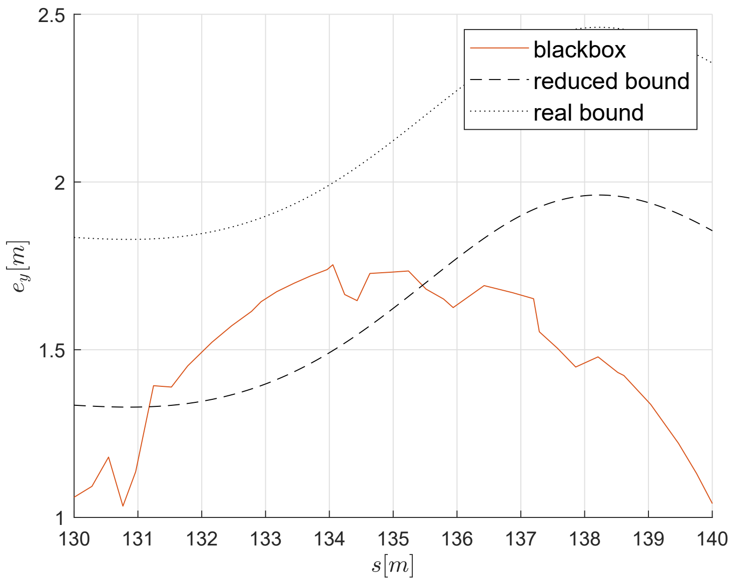


Figure 9.18. Detail of the lateral error value and its reduced bound during the top left curve.

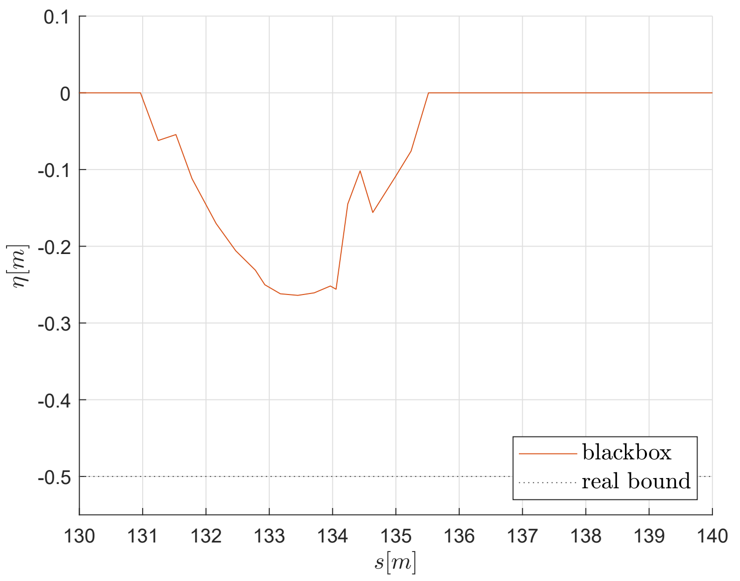


Figure 9.19. Detail of the slack variable value during the top left curve.

Empirical Feasibility Validation

The expedients employed to empirically achieve recursive feasibility, i.e. avoiding hitting track boundaries, of the NLP described in Sec. 9.2.1 have been validated to establish their effectiveness. In particular, the effect of soft constraint on the lateral error can be clearly observed in the top-left turn of the track. In Fig. 9.17-9.19 details of trajectory, lateral error, and slack variable values are shown. While travelling the curve, the go-kart is very close to the left bound of the track, and the slack variable is needed to ensure the feasibility of the problem, while, at the same time, pushing the controller to move away from the limit. In fact, once the *reduced bound* has been overcome, the slack variable is activated, avoiding both the controller to fail and the go-kart to crash.

Conclusion

10

Conclusions and Further Development

The thesis discusses the implementation of Learning-based Nonlinear Model Predictive Control methods for both two-wheel and four-wheel vehicles. The research focuses mainly on the development of grey-box and black-box dynamics models for vehicle control, with specific emphasis on meeting real-time constraints. Gaussian Process Regression has been used to model the data-driven components of the dynamics due to the possibility to employ dimensionality reduction techniques, both offline and online, while preserving model characteristics, e.g. continuity and smoothness. Various reduction techniques were explored, including feature selection procedures, sparse and local GP approximations. Moreover, the tuning of a NMPC virtual rider through a genetic algorithm has been accomplished, specifically considering the trade-off between performance and sensitivity to parameter changes.

10.1 Conclusions

The results demonstrate the effectiveness of the previously mentioned methodologies in the automotive framework. Indeed, GPs resulted suitable for defining control models for both two-wheel and four-wheel vehicles, allowing real-time feasibility through tailored feature selection procedures and offline and online sparse GP approximations. Moreover, the tailored strategy for tuning a NMPC for a virtual motorcycle effectively revealed the trade-off between high-performance and robust configurations of the controller, allowing to exploit the most convenient tuning for the specific context.

10.1.1 Two-wheel vehicles

On the learning dynamics side, the development of a grey-box model significantly enhances the tracking performance of an NMPC controller for a virtual motorcycle. In particular, a grey-box model obtained by offline Variational Free Energy (VFE model) reduction with 20 points, and a grey-box model obtained by online Transduction Learning (TL model) with 10 points have been compared with the nominal physics-based model. Both the proposed models highly improve the closed-loop performance, while highlighting the trade-off between computational burden and input consistency. Indeed, the TL model, based on only 10 points, increases the solver time by 33% with respect to the nominal model, while the VFE model, based on 20 points, needs 77% more time, but the control inputs obtained with TL model exhibit significant high-frequency vibrations.

On the learning design side, the application of an algorithmic tuning procedure for a NMPC virtual rider demonstrates the effectiveness of the approach, finding configurations that perform better than a previously published manual tuning. Furthermore, a comparison on closed-loop performance of the fastest with respect to the most robust configurations illustrates the regularization effect induced in the control aggressiveness by the sensitivity requirements. Finally, an analysis carried on by varying the weights in the neighborhood confirms the suitability of the sensitivity definition, highlighting the robustness induced by the chosen configuration.

10.1.2 Four-wheel vehicles

Firstly, it has been verified that the integration of a learning-based component over the physics-based dynamics in a grey-box characterization improves the efficacy of a nonlinear model predictive contouring controller for lap-time minimization in a high-performance virtual car. In particular, a thorough analysis has been conducted, evaluating the number of inducing points in the sparse approximation, the kernel function, and the feature selection, leading to the most suitable Gaussian Process characterization. The obtained LbNMPC controller outperforms the nominal NMPC both in prediction capabilities and closed-loop performance, maintaining the real-time property.

Secondly, the implementation of a LbNMPC approach using a black-box model for a real go-kart demonstrates the strategy applicability in a challenging, purely data-driven model scenario. In particular, the formulation exploits IMU and localization data collected while a human is driving to obtain GP models of the dynamics, eliminating the need for an a priori known dynamics model. The computational burden of this method has been addressed by reducing the GP size, exploiting local approximations, and applying fast NMPC solutions. The strategy achieved satisfactory results on a 180m long indoor track, comparable to a NMPC based on a detailed nominal dynamic model, although with slightly lower performances, limited by the needed approximations.

10.2 Further development

On one side, research inquiry in the LbNMPC domain for two-wheel vehicles has recently commenced, thus offering substantial avenues for further exploration. For instance, employing black-box modeling techniques for a LbNMPC virtual rider emerges as an interesting perspective. This is particularly pertinent due to the inherent complexities associated with developing precise motorcycle models, which pose considerable challenges.

On the other side, the LbNMPC research on four-wheel vehicles is extensive and expanding, and further development could include analyses of different reduction methods and local approximations, beyond the usage of models leveraging the online data acquired while controlling the system. Moreover, the probabilistic models can be exploited to quantify uncertainty and satisfy the constraints with a given probability by chance constrained optimization. The latter is an emerging technique that is currently being studied and gaining increasing scientific interest due to the possibility of ensuring safety in unknown environments, and it is the privileged direction for further development.

Appendix

A

LbMATMPC toolbox

In this chapter, the toolbox `LbMATMPC`, that implements Learning-based Nonlinear Model Predictive Control in the Lerning Dynamics framework, is presented. It is an extension of a previous developed toolbox, i.e. `MATMPC` [117], that allows to implement NMPC controllers. The open-source code is available at `https://github.com/pi-cos/MATMPC/tree/LbMATMPC`.

A.1 Algorithm

`MATMPC` is an open source software built in `MATLAB` for implementing NMPC strategies. It is designed to facilitate modelling, controller design and simulation for a wide class of NMPC applications. `MATMPC` has a number of algorithmic modules, including automatic differentiation, direct multiple shooting, condensing, linear quadratic program (QP) solver and globalization.

In `LbMATMPC`, i.e. Learning-based `MATMPC`, the possibility to use grey- or black-box GP-based models has been introduced. At every discrete time instant k of the control task, a NLP is formulated by applying direct multiple shooting [60] to an Optimal Control Problem (OCP) over the prediction horizon, which is divided into N *shooting intervals* $[t_{0|k}, t_{1|k}, \dots, t_{N|k}]$, as expressed in Eq. (2.3). To include the discrete dynamics, the continuity constraint (2.3c) and its QP approximation is modified as detailed in Sec. 2.4.2.

A.1.1 Learning-based features

In the following, peculiar features related to the GP modelling are described, while specifics on the available *standard* NMPC features can be found in [117].

Within the toolbox, it is possible to perform the GPR through the *gpr-pytorch* library in the MATLAB framework, given either model mismatch or input-output measurements. The procedure automatically computes the information needed for the GP prediction within the NMPC problem. At the current stage of development, it is possible to set different regularization values to avoid numerical issues. We implemented this feature allowing the possibility of setting a minimal value of σ_n. Moreover, we already implemented two strategies to limit the computational burden, aimed at reducing the number of samples. The first consists of dowsampling data with constant step. The second, named *Subset Of Data* (SoD), reduces the data set discarding less informative samples, according to a tunable relevance threshold. In the implementation, Squared Exponential kernel is defined, and any custom kernel definition can be used within the

NMPC once its analytical expression is given. Besides, the GP structure within the NMPC allows to update the GP parameters online, without any re-compilation of the code.

During the problem solution, all the involved quantities are accessible within the NMPC structure, at the end of every SQP iteration, and those specifically dealing with GP-based integration are:

- $x\_gp \in \mathbb{R}^{n_x \times (N+1)}$, results of the integration from GP, i.e. $\bar{\boldsymbol{\phi}}(\boldsymbol{x}_j, \boldsymbol{u}_j)\ \forall j \in [0, N]$
- $x\_ode \in \mathbb{R}^{n_x \times (N+1)}$, results of the integration from ODE, i.e. $\tilde{\boldsymbol{\phi}}(\boldsymbol{x}_j, \boldsymbol{u}_j)\ \forall j \in [0, N]$
- $A\_gp \in \mathbb{R}^{n_x \times N n_x}$, state transition matrix for GP component, i.e. $A_{j,GP}\ \forall j \in [0, N-1]$
- $A\_ode \in \mathbb{R}^{n_x \times N n_x}$, state transition matrix for ODE component, i.e. $A_{j,ODE}\ \forall j \in [0, N-1]$
- $B\_gp \in \mathbb{R}^{n_x \times N n_u}$, control transition matrix for GP component, i.e. $B_{j,GP}\ \forall j \in [0, N-1]$
- $B\_ode \in \mathbb{R}^{n_x \times N n_u}$, control transition matrix for ODE component, i.e. $B_{j,ODE}\ \forall j \in [0, N-1]$

Finally, note that the LbNMPC formulation needs *only* the analytical definition of the GP estimate function $\bar{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{u})$ that can be implemented by the user, and the GPR can also be accomplished by any other chosen tool.

A.1.2 Computational burden

An important aspect within the NMPC strategy is the computational burden of the NLP problem solution, as it has to be obtained at every control instant. For this reason, the required time needed for computing each GP-related component of the problem formulation for a generic GP model in the form described in Sec. 3.1 has been analyzed, and the results are summarized in Tab. A.1. The computations, as expected, results to be linear with respect to the no. of training points of the GP. Interestingly, the time required is affine (constant + linear) w.r.t. the dimension of the GP training points. Note that each of this computation has to be accomplished for every instant in the prediction horizon, hence, e.g., having 400 5-dimensional training point with a prediction horizon of $N = 100$ instants, the total time required for the GP-related computations is around $20ms$ on a PC with Intel Core i7-3770@3.50GHz CPU.

A.2 Experimental trial

In this section, experimental results using grey-box modeling within `LbMATMPC` on a real Furuta Pendulum (depicted in Fig A.2) are presented.

Table A.1. The time required for the computation of each GP-related component w.r.t. the no. of training points for the GP (T) on a PC with Intel Core i7@3.50GHz CPU in microseconds.

T	10	20	50	75	100	150	200	300	400
GP evaluation	5.9	8.1	10.3	15.0	18.1	24.5	31.4	44.8	59.9
GP jacobian	7.2	9.4	13.2	18.8	23.2	31.8	41.9	60.8	79.1
GP adjoint sens	6.2	8.8	11.5	18.7	22.9	38.3	41.3	60.2	78.2
Total	19.3	26.2	35.0	52.9	64.3	94.7	114.7	165.7	217.3

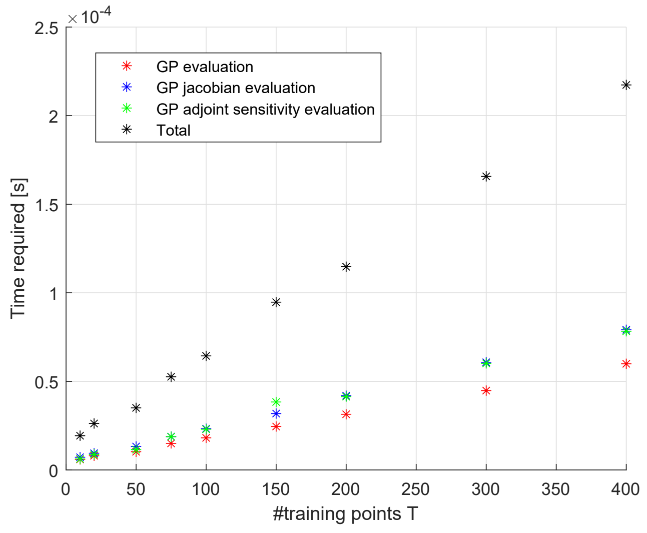


Figure A.1. The time required for the computation of each GP-related component w.r.t. the no. of training points for the GP (T) on a PC with Intel Core i7@3.50GHz CPU.

A.2.1 Nominal Dynamics

For the grey-box model, the nominal dynamics used within the NMPC is given by a model of the Furuta pendulum with generalized coordinates

$$\boldsymbol{q} = [\alpha,\ \theta]^T, \tag{A.1}$$

where α is the shaft angle and θ is the pendulum angle (as 0 when pointing upwards). The equations of motion are

$$M(\boldsymbol{q})\ \ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}}) + G(\boldsymbol{q}) = B\tau, \tag{A.2}$$

with

$$\begin{aligned} M &= \begin{bmatrix} m_2L_1 + I_{zz} & m_2L_1l_2cos(\theta) \\ m_2L_1l_2cos(\theta) & m_2l_2^2 \end{bmatrix}, \qquad & C &= \begin{bmatrix} -m_2L_1l_2sin(\theta)\dot{\theta}^2 \\ -m_2l_2sin(\theta)\dot{\theta}\dot{\alpha} \end{bmatrix}, \\ G &= \begin{bmatrix} 0 \\ -m_2gl_2sin(\theta) \end{bmatrix}, & B &= \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \end{aligned} \tag{A.3}$$

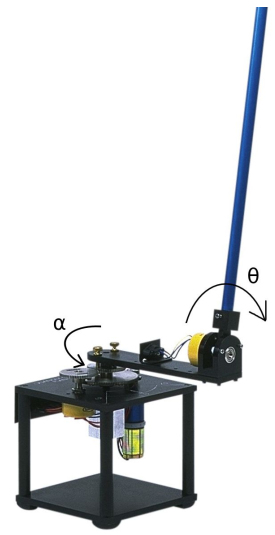


Figure A.2. The Furuta pendulum used in the experiments.

Table A.2. The Furuta pendulum nominal quantities.

m_2	L_1	I_{zz}	l_2	N_m	k_t	R_a	R_s	k_e
0.21kg	0.145m	$0.0044kgm^2$	0.305m	70	$7.68{\cdot}10^{-3}\frac{Nm}{A}$	2.6Ω	0.5Ω	$7.68{\cdot}10^{-3}\frac{Vs}{rad}$

where L_1 is the shaft length, m_2 is the pendulum mass, l_2 is the position of the pendulum center of mass along its axis, I_{zz} is the pendulum inertia, and g is the gravity acceleration. The torque expressed on the shaft is described by

$$\tau = N_m \frac{k_t}{(R_a + R_s)}(u_{drv} - u_e), \tag{A.4}$$

where v is the control input, $u_{drv} = k_{drv}v$ is the driver output voltage, $u_e = k_e N_m \dot{\alpha}$ is the back electromotive force, N_m is the motor-shaft reduction ratio, $k_e, k_t, k_{drv}, R_s, R_a$ are motor constants and armature resistances.

A.2.2 Grey-box model

The dynamics of the real pendulum presents different unmodeled non-linear behaviours, such as friction, motor electrical dynamics, control application delays, etc., that make the NMPC predictions inaccurate.

The GPR procedure described in Ch. 3 has been employed setting the minimum value of *normalized* σ_n to 0.05 and SoD active. The model-mismatch data have been obtained applying the NMPC strategy with the nominal dynamics model, and exploiting the computed one-step-ahead predictions. We derived a grey-box model in the form of Eq. (2.16).

A.2.3 Controller Setup

The proposed strategy has been implemented to control the Furuta pendulum, where the task is to track with the shaft a square wave of $\pm 45deg$ while maintaining the pendulum

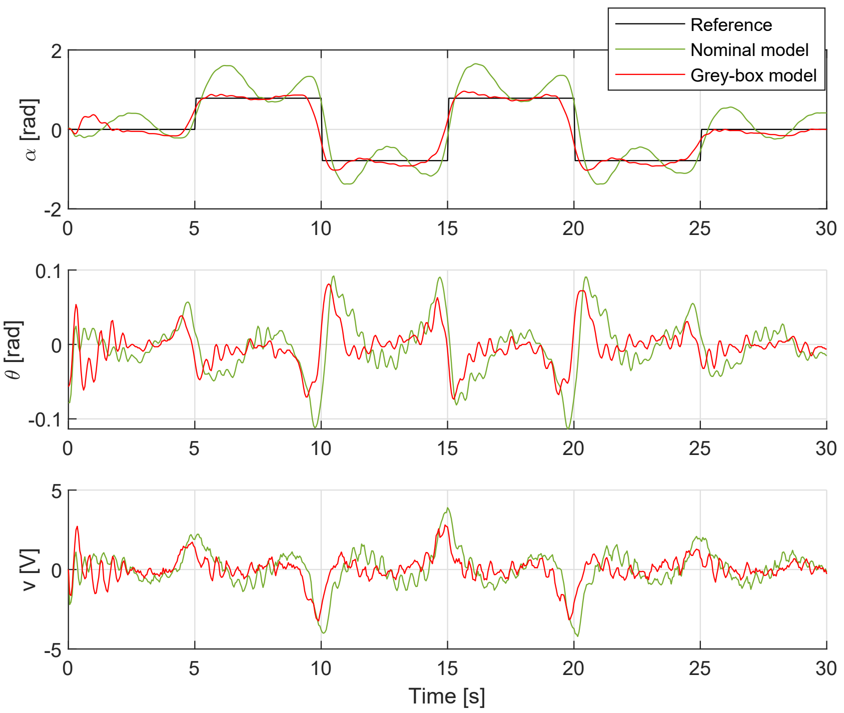


Figure A.3. The shaft angle α, pendulum angle θ and the voltage input v of the Furuta pendulum, with and without the GP modeling.

in the upright position. The control frequency is $20Hz$, the GP has been trained with $T = 400$ points and the prediction length has been set as $N = 70$. The mean and maximum computational time required for solving the problem resulted in $20.27ms$ and $24.94ms$, respectively, on PC with Intel Core i7-9700@3.0GHz CPU.

The NMPC model has been formulated in velocity form, i.e. defining the control input $u = \dot{v}$ and the state $\boldsymbol{\xi} = [\alpha, \theta, \dot{\alpha}, \dot{\theta}, v]^T$. `MATMPC` has been set to use HPIPM as QP sparse solver [67] and Real-Time Iteration scheme [61]. The adopted cost function is

$$\boldsymbol{h}(\boldsymbol{\xi}, u) = [\alpha, \theta, \dot{\alpha}, \dot{\theta}, v, \dot{v}]^T, \tag{A.5}$$

with weight matrix $W = diag([5 \cdot 10^1, 5 \cdot 10^2, 10^{-3}, 10^{-3}, 10^{-6}, 5 \cdot 10^{-2}])$, and the constraint function

$$\boldsymbol{r}(\boldsymbol{\xi}, u) = [\alpha, v, \dot{v}]^T, \tag{A.6}$$

with bounds $\underline{\boldsymbol{r}} = [-\pi, -10, -200]^T$ and $\overline{\boldsymbol{r}} = [+\pi, +10, +200]^T$.

A.2.4 Results

The dynamics of the real pendulum presents different unmodeled non-linear behaviours, such as friction, motor electrical dynamics, control application delays, etc., that make the NMPC predictions inaccurate. Fig. A.3 shows how the introduction of the grey-box model considerably enhance the control performance of the system, highly reducing the overshoot in the shaft position. Moreover, the prediction error on the system velocities has been significantly lowered, as depicted in Fig. A.4 and reported in Tab. A.3, where Root-Mean Squared Error (RMSE) of one-step-ahead prediction errors are stated.

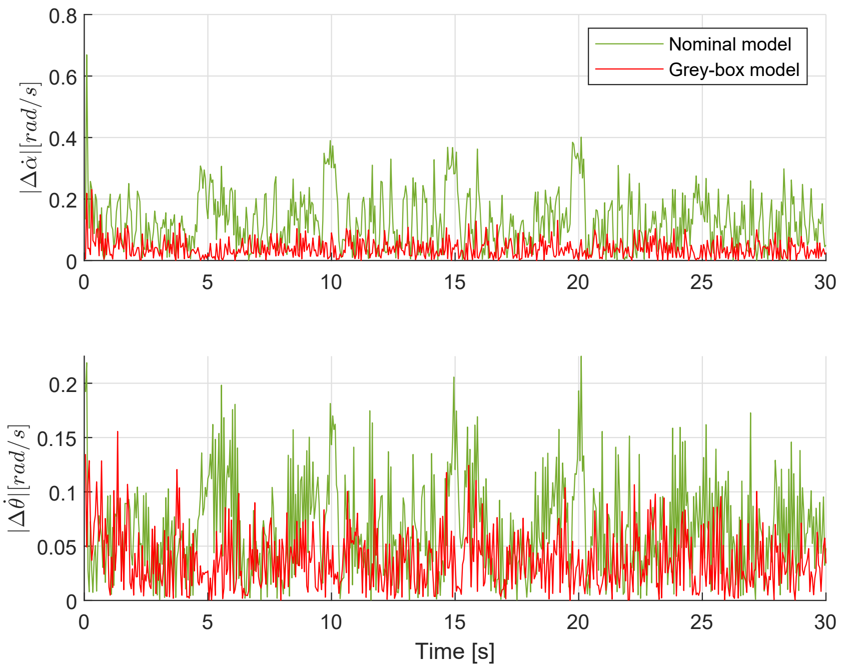


Figure A.4. The one-step-ahead-prediction errors on the shaft $\dot{\alpha}$ and pendulum $\dot{\theta}$ angular velocities, with and without the GP modeling.

Table A.3. RMSE of the NMPC predictions for the furuta pendulum.

	nominal	grey-box
$\lvert\Delta\dot{\alpha}\rvert$ [rad/s]	$1.6 \cdot 10^{-1}$	$0.4 \cdot 10^{-1}$
$\lvert\Delta\dot{\theta}\rvert$ [rad/s]	$7.9 \cdot 10^{-2}$	$4.5 \cdot 10^{-2}$

A.3 Simulative trials

We applied `LbMATMPC` on a simulated inverted cart-pole system. To consider the effects of model mismatch, in the nominal model the length of the pendulum is incorrect ($l_{true} = 0.5m$ vs $l_{nom} = 0.8m$). Moreover, a Monte Carlo analysis of the control performance with different number of training points and different modeling is presented.

The model of the system is described in [118] and depicted in Fig. A.5. The state is

$$\boldsymbol{\xi} = [\boldsymbol{q}, \dot{\boldsymbol{q}}]^T = [p, \theta, v, \omega]^T, \tag{A.7}$$

where p is the cart position, θ is the pendulum angle (0 in the upright position), and v and ω are the respective velocities, while the input u is the force applied to the cart F.

A.3.1 Controller setup

The NMPC tool `MATMPC` has been set to use HPIPM as QP sparse solver [67] and RTI scheme [61] with control frequency $f_c = 40Hz$ and prediction horizon $N = 80$. The cost

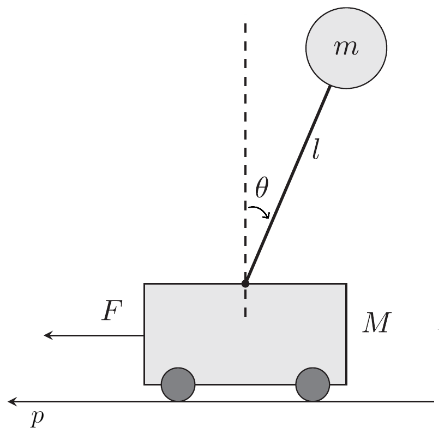


Figure A.5. Model of the inverted cart-pole system.

Table A.4. RMSE of the NMPC predictions in the swing-up meaneuver.

	nominal	grey-box	black-box	correct
$\|\Delta v\|$ [m/s]	$4.0 \cdot 10^{-3}$	$1.8 \cdot 10^{-3}$	$4.0 \cdot 10^{-3}$	$1.1 \cdot 10^{-3}$
$\|\Delta \omega\|$ [rad/s]	$11.1 \cdot 10^{-2}$	$2.8 \cdot 10^{-2}$	$2.2 \cdot 10^{-2}$	$1.7 \cdot 10^{-2}$

function has been defined as

$$\boldsymbol{h}(\boldsymbol{\xi}, u) = [p, \theta, v, \omega, F]^T, \tag{A.8}$$

with weight matrix $W = diag([10, 10, 0.1, 0.1, 0.01])$.

A.3.2 Single swing up maneuver

A comparison on the controller performance during the swing up maneuvers has been performed with four different model definitions: (i) nominal model, (ii) grey-box model, (iii) black-box model, (iv) correct analytical model. For the grey-box model, the GP has been trained on the one-step-ahead prediction errors $\dot{\boldsymbol{q}}_{\boldsymbol{k}} - \hat{\dot{\boldsymbol{q}}}_{\boldsymbol{k}}$ of the nominal model NMPC, while the black-box model has been obtained training the GP on the velocity increments $\dot{\boldsymbol{q}}_{\boldsymbol{k+1}} - \dot{\boldsymbol{q}}_{\boldsymbol{k}}$, both using 200 training points and the minimum value of *normalized* σ_n as 0.01. The systems have been then formulated in the form (2.16).

Fig. A.6 shows the capability of both the grey-box and the black-box model NMPC to complete the task with performance very close to the correct analytical model, both in terms of control effectiveness and input usage. Moreover, the prediction error is highly reduced in both cases, comparable with the correct analytical model (where only linearization error is present), as shown in Fig. A.7 and reported in Tab. A.4.

A.3.3 Monte Carlo analysis

A more comprehensive evaluation of the control performance has been carried out by choosing 100 random configurations of training points with incremental dimensions

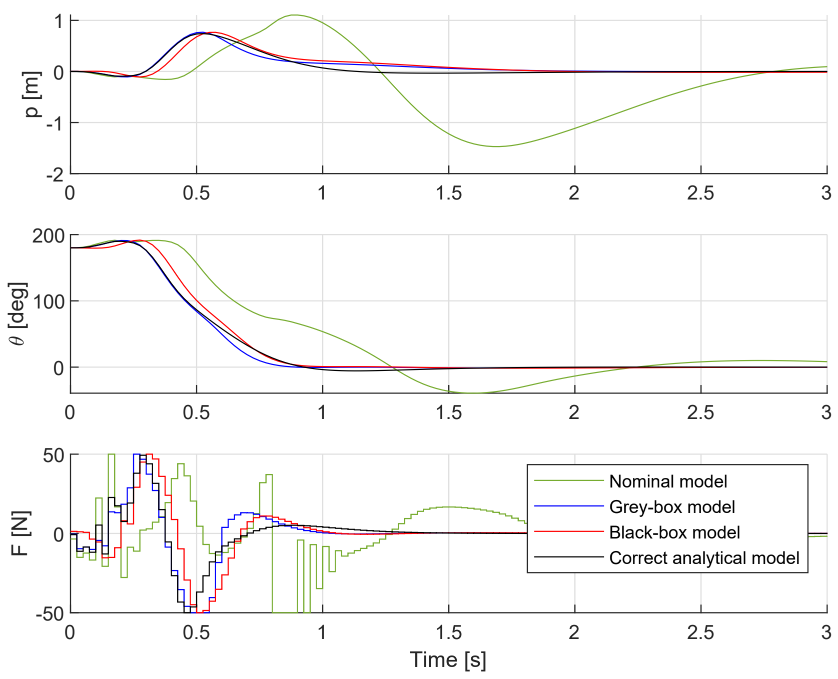


Figure A.6. Swing up of the inverted pendulum using nominal parameters for MPC model, grey-box GP model, black-box model and correct parameters for MPC model.

(between 20 to 199 training points). The following performance index has been defined

$$c = \sum_k \left[1 - exp\left(- \left((\frac{\theta_k}{l_\theta})^2 + (\frac{p_k}{l_p})^2 \right) \right) \right], \tag{A.9}$$

with $l_\theta = \pi, l_p = 1$, that accounts for the total displacement from the target both in pendulum angle and in cart position. The results have been depicted in Fig. A.8, where only the completed[1] task in a confidence interval of [10, 90]% have been considered, while percentage of the task completion have been reported in Fig. A.9. The grey-box model performs remarkably better for a low number of training points, where, in addition, the black-box model completes the task in only 10% of the trials. For higher number of training points, the difference between grey-box and black-box models gets less evident, and with 199 points both reach a performance index very close to the correct analytical model. Note that the whole controller configuration has been maintained the same in all the trials, and, in particular, the minimum value of normalized σ_n, i.e. the regularization parameter, has been set the same for both grey-box and black-box models. This parameter can affect significantly the predictions capabilities of the model, and, in case there is no measurement noise, a black-box model could possibly obtain better performances with lower σ_n.

[1]the task has been considered completed if no numerical issues during the NMPC solution arised, and, in the last second of the simulation, the pendulum angle was in the range $\theta \in [-10, +10] deg$.

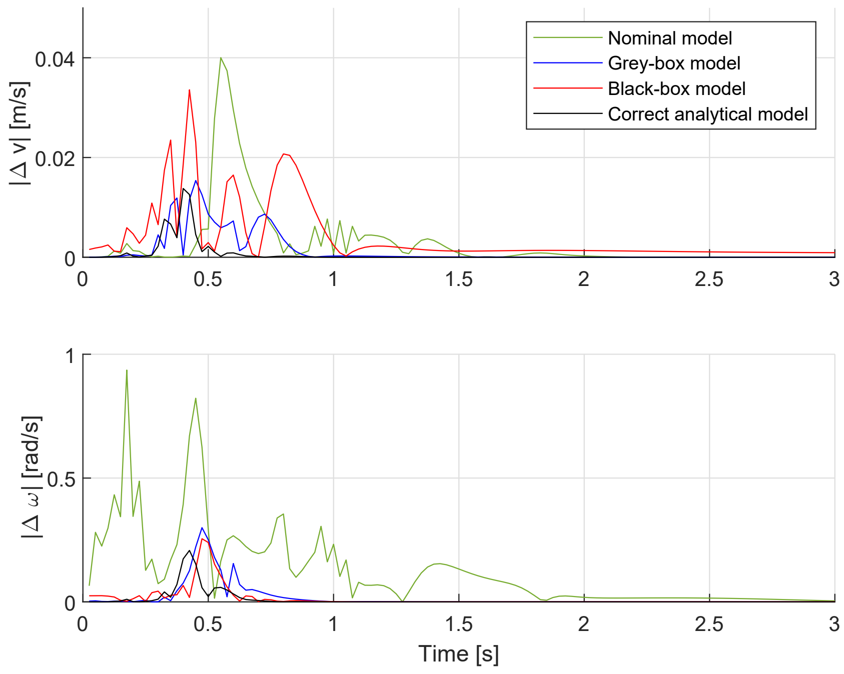


Figure A.7. The one-step-ahed prediction error using nominal parameters for MPC model, grey-box GP model, black-box model and correct parameters for MPC model.

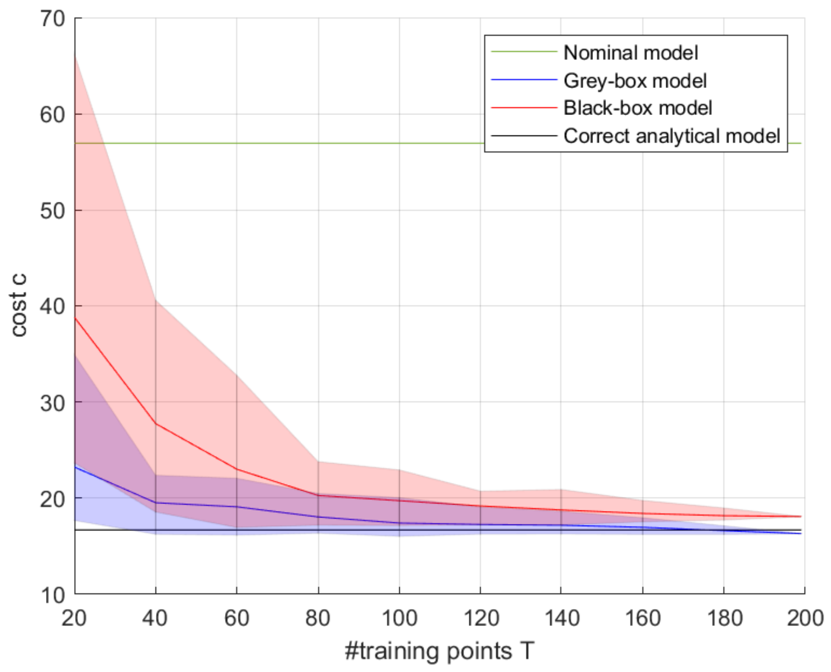


Figure A.8. The cumulative cost using nominal parameters for MPC model, grey-box GP model, black-box model and correct parameters for MPC model. Solid line indicates the mean of the cost obtained using the model, while the coloured area represents the [10, 90]% percentiles.

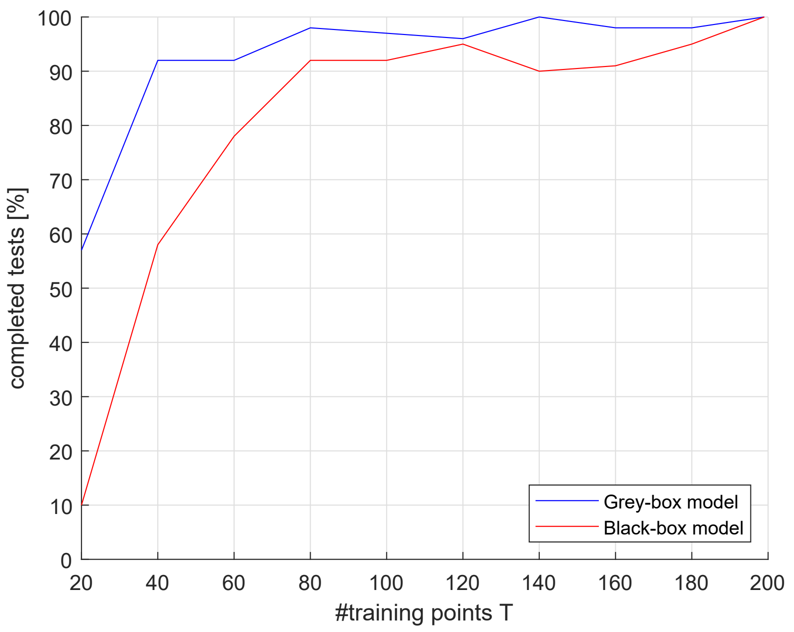


Figure A.9. Percentage of task completion for grey-box and black-box models w.r.t the number of training points T.

B

Comparison of Learning-based Continuous vs Discrete Dynamics

The learning-based continuous dynamics presented in Sec. 2.4 has been validated in an experimental scenario, and compared with the learning-based discrete dynamics usually employed in this framework [46].

In the following, the controller setup is described and the GPR procedure and data gathering are illustrated. A specific comparison between continuous and discrete GP modeling is obtained and the possibility to use the learned continuous model to implement different sampling times and integration grids is exhibited.

For the grey-box model, the nominal dynamics used within the NMPC is given by a model of the Furuta pendulum as described in A.2.1.

B.1 Controller Setup

In order to fairly compare different methods, the same setup for the NMPC controller has been maintained. The task is to hold the Furuta pendulum in the upright position, while tracking with the shaft a square wave of $\pm 45 deg$.

The NMPC model has been formulated in velocity form, i.e. defining the state $\boldsymbol{\xi} = [\alpha, \theta, \dot{\alpha}, \dot{\theta}, v]^T$ and the control input $u = \dot{v}$. HPIPM has been used within MATMPC as QP sparse solver [67] and RTI scheme [69] has been adopted. Moreover, a slack variable s has been used to define a soft constraint on the shaft angle α.

The constraint function has been defined as

$$\boldsymbol{r}(\boldsymbol{\xi}, u) = [\alpha + s, v, \dot{v}, s]^T, \tag{B.1}$$

with bounds

$$\underline{\boldsymbol{r}} = [-\pi, -10, -200, -\pi]^T, \tag{B.2}$$

$$\overline{\boldsymbol{r}} = [+\pi, +10, +200, +\pi]^T. \tag{B.3}$$

The adopted cost function is

$$\boldsymbol{h}(\boldsymbol{\xi}, u) = [\alpha, \theta, \dot{\alpha}, \dot{\theta}, v, \dot{v}, s]^T, \tag{B.4}$$

with weight matrix $W = diag([5 \cdot 10^0, 10^1, 10^{-3}, 10^{-3}, 10^{-6}, 5 \cdot 10^{-3}, 10^1])$. At the terminal stage, $\boldsymbol{r}_N(\boldsymbol{\xi}_N)$, $\boldsymbol{h}(\boldsymbol{\xi}_N)$ and W_N are set as before, excluding the inputs releted terms.

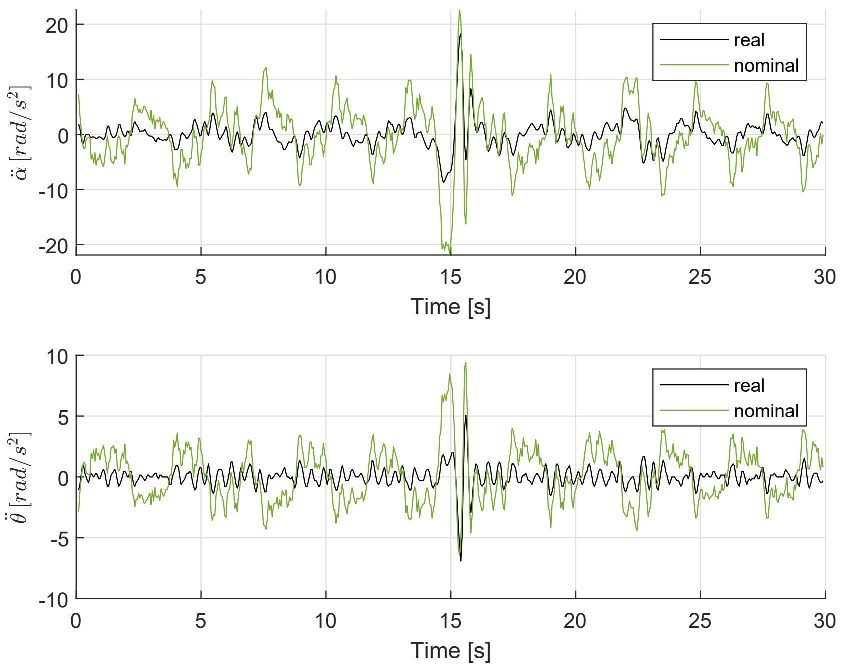


Figure B.1. The real and nominal accelerations data gathered during the nominal NMPC run.

B.2 Data Acquisition and Gaussian Process Regression

Firstly, the task using NMPC with nominal dynamics at $20Hz$ has been accomplished. Starting from α_k, θ_k, $\dot{\alpha}_k$, $\dot{\theta}_k$, v_k measurements, the nominal accelerations $\tilde{\ddot{q}}$ have been computed inverting the nominal dynamics model presented in Eq. (A.2), i.e.

$$\tilde{\ddot{\boldsymbol{q}}} = M(\boldsymbol{q})^{-1}\left[B\tau - C(\boldsymbol{q},\dot{\boldsymbol{q}}) - G(\boldsymbol{q})\right]. \tag{B.5}$$

Moreover, real accelerations $\ddot{\alpha}$, $\ddot{\theta}$ of the system have been computed by central difference as

$$\ddot{q}_k = \frac{\dot{q}_{k+1} - \dot{q}_{k-1}}{2 \cdot T_s}. \tag{B.6}$$

Due to the absence of frictions, control application delays, electrical dynamics, etc., in the nominal model, the computed accelerations are inaccurate, as shown in Fig. B.1. The difference between real and nominal accelerations $\ddot{\boldsymbol{q}} - \tilde{\ddot{\boldsymbol{q}}}$ have been used as target for the GP $\boldsymbol{y}_{GP}$. The GP has been trained with $T = 600$ points and setting the minimum value of *normalized* σ_n to 0.1. The GPR procedure described in Sec. 2.4 has been employed, exploiting *Subset of Data* (SoD) method [85], to derive a model of the acceleration errors (shown in Fig. B.2), which resulted to have 49 inducing points. Using the obtained GP, a model in the form (2.21) has been defined.

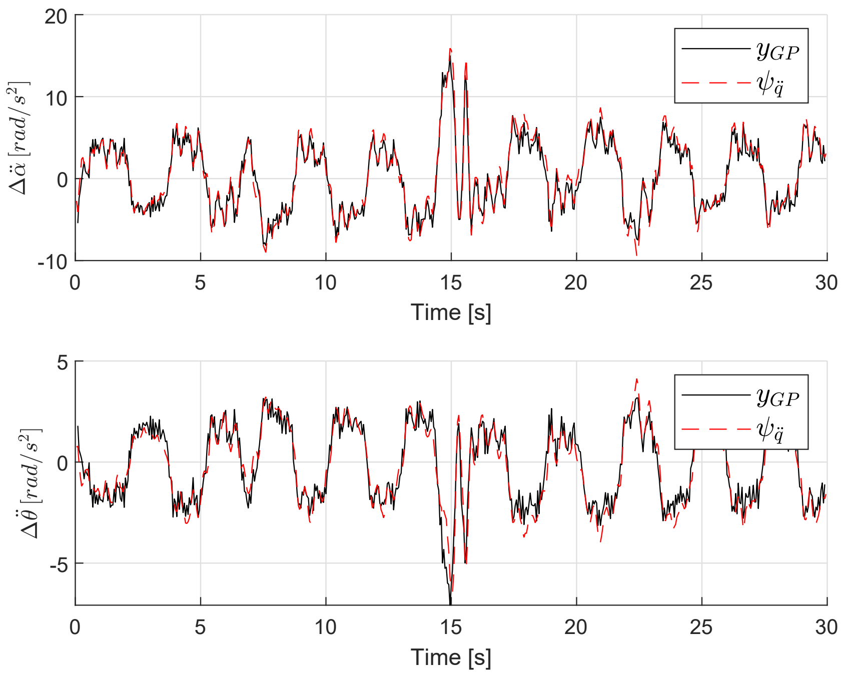


Figure B.2. The acceleration difference $y_{GP} = \ddot{q} - \tilde{\ddot{q}}$ for the data gathered during the nominal NMPC run and their GP estimation $\psi_{\ddot{q}}$.

B.3 Results

A test of continuous vs discrete grey-box modeling is presented, using both control frequencies $20Hz$ and $50Hz$. The nominal NMPC is also employed for comparison. The discrete modeling is obtained as described in [53] and Ch. A, for the two different control frequencies. In particular, a GP $\bar{\boldsymbol{\phi}}^D_{\dot{q}}(\boldsymbol{\xi}_k, \boldsymbol{u}_k)$ has been trained on the velocity prediction errors and the resulting grey-box model is

$$\boldsymbol{\xi}_{k+1} = \tilde{\boldsymbol{\phi}}(\boldsymbol{\xi}_k, \boldsymbol{u}_k) + \bar{\boldsymbol{\phi}}^D(\boldsymbol{\xi}_k, \boldsymbol{u}_k), \tag{B.7}$$

where $\tilde{\boldsymbol{\phi}}$ is the integration of the nominal dynamics and $\bar{\boldsymbol{\phi}}^D = \left[\bar{\boldsymbol{\phi}}^T_{\dot{q}} \cdot \frac{T_s}{2}, \bar{\boldsymbol{\phi}}^T_{\dot{q}}\right]^T$ and $\bar{\boldsymbol{\phi}}_{\dot{q}}$ is the GP-based estimation of the velocity increment error. Notably, in the discrete case the GP had to be trained differently when changing the control frequency, while the learned

Table B.1. RMSE of the NMPC one-step-ahead-predictions on the velocities.

		$\lvert\Delta\dot{\alpha}\rvert$ [rad/s]	$\lvert\Delta\dot{\theta}\rvert$ [rad/s]
20Hz	nominal	$19.1 \cdot 10^{-2}$	$8.4 \cdot 10^{-2}$
	continuous	$7.1 \cdot 10^{-2}$	$4.4 \cdot 10^{-2}$
	discrete	$8.6 \cdot 10^{-2}$	$5.4 \cdot 10^{-2}$
	continuous NUG	$7.3 \cdot 10^{-2}$	$4.5 \cdot 10^{-2}$
50Hz	nominal	$9.5 \cdot 10^{-2}$	$6.5 \cdot 10^{-2}$
	continuous	$5.3 \cdot 10^{-2}$	$5.3 \cdot 10^{-2}$
	discrete	$5.4 \cdot 10^{-1}$	$5.1 \cdot 10^{-2}$

Table B.2. RMS of the NMPC tracking error on the angles.

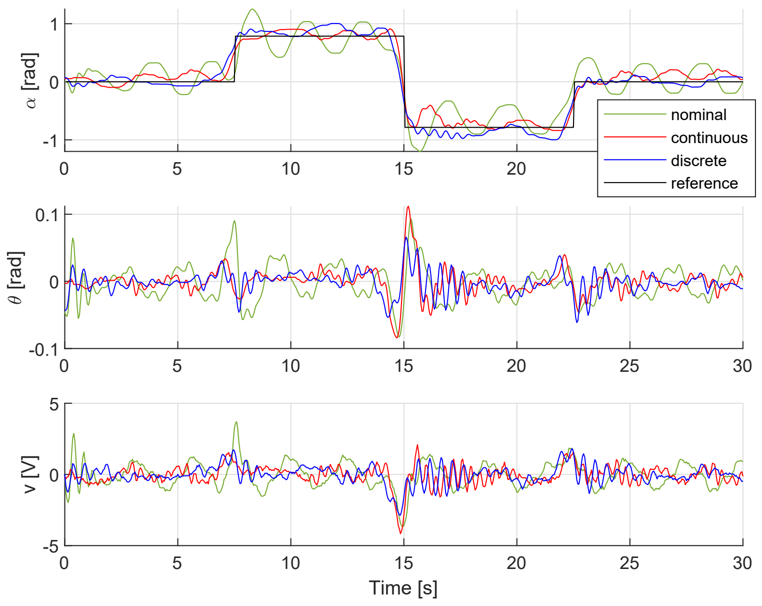
		$\lvert\alpha - \alpha^{ref}\rvert$ [rad]	$\lvert\theta\rvert$ [rad]
20Hz	nominal	$24.5 \cdot 10^{-2}$	$2.6 \cdot 10^{-2}$
	continuous	$12.8 \cdot 10^{-2}$	$1.9 \cdot 10^{-2}$
	discrete	$13.7 \cdot 10^{-2}$	$1.6 \cdot 10^{-2}$
	continuous NUG	$14.2 \cdot 10^{-2}$	$2.0 \cdot 10^{-2}$
50Hz	nominal	$24.0 \cdot 10^{-2}$	$2.3 \cdot 10^{-2}$
	continuous	$13.0 \cdot 10^{-2}$	$1.8 \cdot 10^{-2}$
	discrete	$12.4 \cdot 10^{-1}$	$1.5 \cdot 10^{-2}$

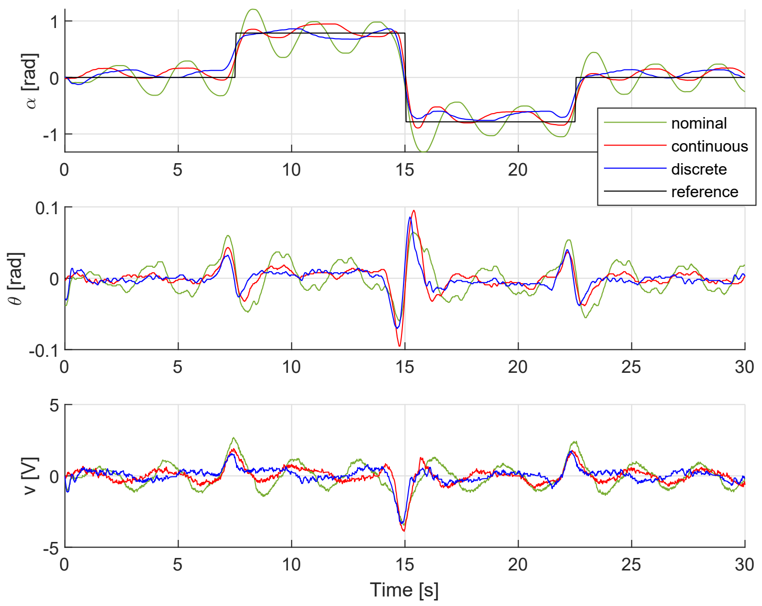
model for the continuous case has been maintained the same as described in Sec. B.2, since the change in integration frequency does not require a new training.

Very similar results have been obtained applying continuous and discrete modeling. Fig. B.3a and B.3b shows how both the learning-based models at both the control frequencies considerably overcome the control performance of the nominal NMPC, highly reducing the low-frequency oscillation in the shaft position, reflected in a lower usage of the control input. Moreover, the prediction error on the system velocities is significantly lowered, as depicted in Fig. B.4a, B.4b and reported in Tab. B.1. Also, the tracking errors on both shaft angle α and pendulum angle θ are effectively reduced, as reported in B.2. The mean time required for the nominal MPC, the continuous with GP using 49 points and the discrete using 101 points are $0.98ms$, $8.49ms$ and $3.8ms$, respectively. Note that, in this case, the discrete modelling required more inducing points with the same SoD threshold.

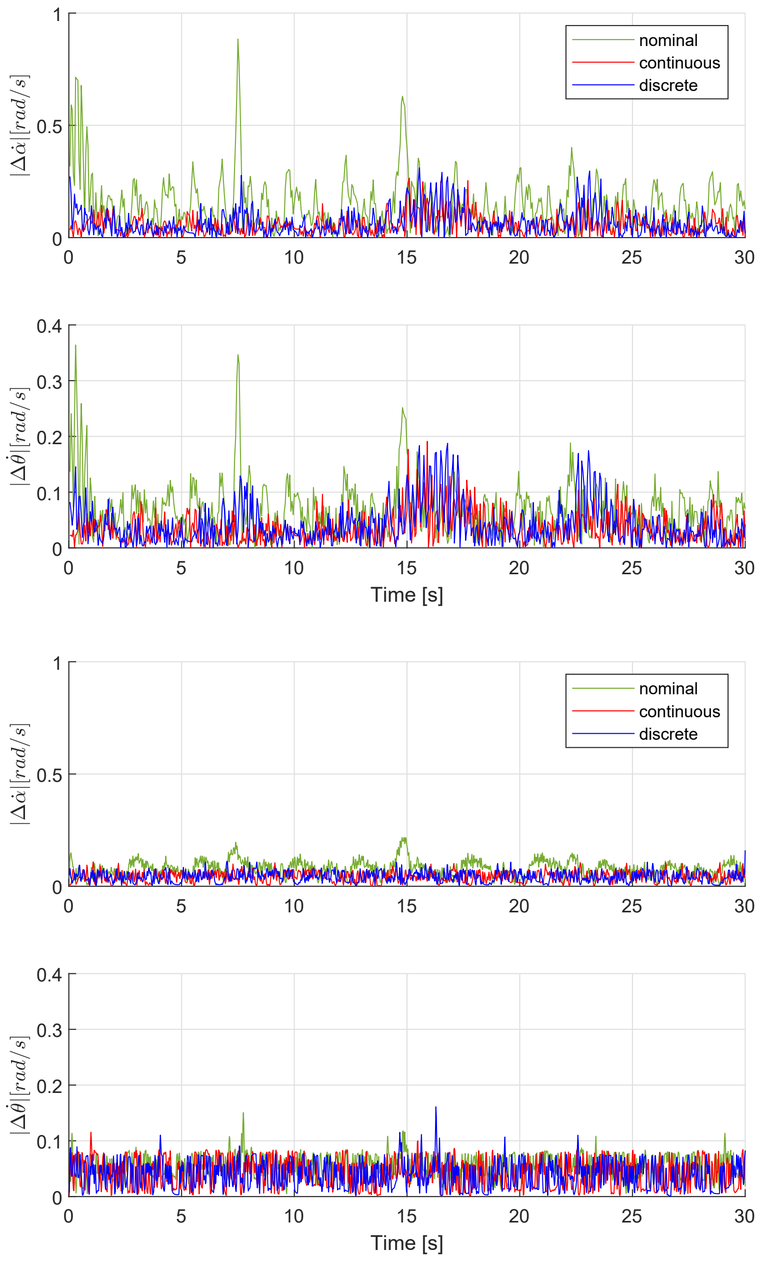
B.3.1 Comparison with Non-Uniform Grid

A peculiar feature of the continuous model definition is that, once it is trained, it is suitable for any integration frequency. In particular, the formulation allows to use Non-Uniform Grid (NUG) integration, and an implementation using a grid of $r = 14$ points distributed as $G = [1,\ 2,\ 5{:}7{:}61,\ 66{:}2{:}70]$, where $a{:}b{:}c$ is a vector from a to c at step b, has been accomplished. This approach allows to maintain the prediction horizon length while highly reducing the computational time. Since the objective function definition changes its meaning while using NUG, different weights for the shaft position and velocities have been used, i.e. $W = diag([10{\cdot}10^1, 5 \cdot 10^{-3}, 10^{-3}, 10^{-6}, 5 \cdot 10^{-3}, 10^1])$. Results in terms of positions and control input are provided in Fig. B.5a, while prediction errors are shown in Fig. B.5b. Notably, the system behaviour, control input and prediction errors are very close to the Uniform Grid, while mean time required to solve the problem resulted $1.8ms$, comparable to the nominal NMPC one.



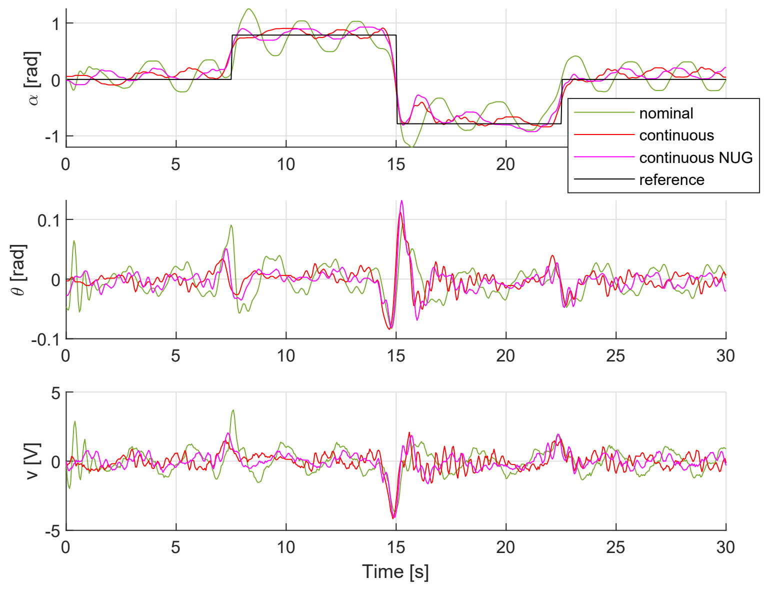


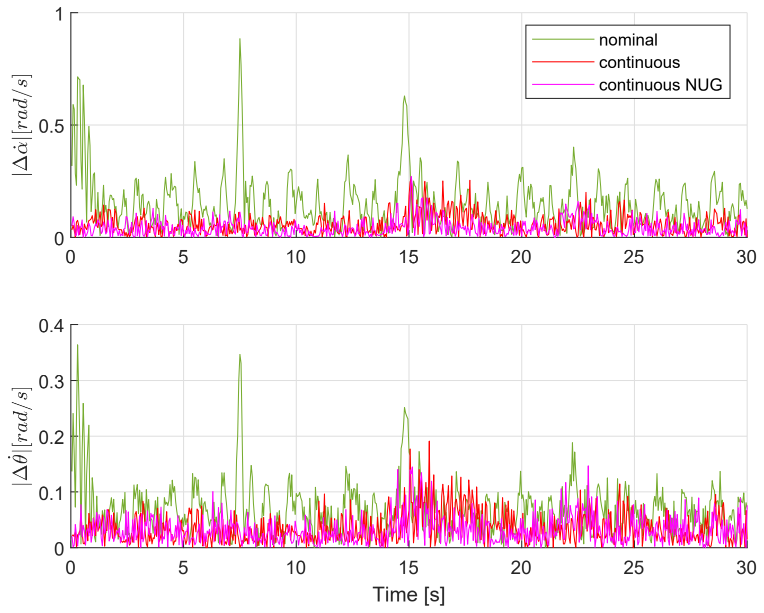
(b) Positions and input with control frequency $f_c = 50Hz$.



(b) One-step-ahead prediction errors with $f_c = 50Hz$.

Figure B.4. Continuous-time versus discrete-time comparison: (a-b) shaft angle α, pendulum angle θ and the voltage input v of the Furuta pendulum; (c-d) one-step-ahead-prediction errors on the shaft $\dot{\alpha}$ and pendulum $\dot{\theta}$ angular velocities.





(b) One-step-ahead-prediction errors on angular velocities.

Figure B.5. Comparison nominal, continuous with Uniform Grid and continuous with Non Uniform Grid integration at 50Hz.

Bibliography

[1] A. Lindgren and F. Chen, "State of the art analysis: An overview of advanced driver assistance systems (adas) and possible human factors issues," in *Human factors and economics aspects on safety*, 2006.

[2] M. Morari and J. H. Lee, "Model predictive control: past, present and future," *Computers & Chemical Engineering*, vol. 23, no. 4-5, pp. 667–682, 1999.

[3] J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl, "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles," in *European Control Conference (ECC)*, 2013, pp. 4136–4141.

[4] A. Carvalho, Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "Predictive control of an autonomous ground vehicle using an iterative linearization approach," in *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on.* IEEE, 2013, pp. 2335–2340.

[5] M. Massaro, "A nonlinear virtual rider for motorcycles," *Vehicle system dynamics*, vol. 49, no. 9, pp. 1477–1496, 2011.

[6] D. M. Giner, "Symbolic-numeric tools for the analysis of motorcycle dynamics: development of a virtual rider for motorcycles based on model predictive control," Ph.D. dissertation, Universidad M.H. de Elche, 2016.

[7] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, pp. 628–647, 09 2015.

[8] R. Verschueren, M. Zanon, R. Quirynen, and M. Diehl, "Time-optimal race car driving using an online exact hessian based nonlinear MPC algorithm," in *European Control Conference (ECC)*, 2016, pp. 141–147.

[9] J. Kabzan, M. I. Valls, V. J. F. Reijgwart, H. F. C. Hendrikx, C. Ehmke, M. Prajapat, A. Bühler, N. Gosala, M. Gupta, R. Sivanesan, A. Dhall, E. Chisari, N. Karnchanachari, S. Brits, M. Dangel, I. Sa, R. Dubé, A. Gawel, M. Pfeiffer, A. Liniger, J. Lygeros, and R. Siegwart, "Amz driverless: The full autonomous racing system," *Journal of Field Robotics*, vol. 37, no. 7, pp. 1267–1294, 2020.

[10] R. Frezza and A. Beghi, "A virtual motorcycle driver for closed-loop simulation," *IEEE control systems*, vol. 26, no. 5, pp. 62–77, 2006.

[11] S. Rowell, A. Popov, and J. Meijaard, "Application of predictive control strategies to the motorcycle riding task," *Vehicle System Dynamics*, vol. 46, no. S1, pp. 805–814, 2008.

[12] S. Rowell, A. A. Popov, and J. P. Meijaard, "Predictive control to modelling motorcycle rider steering," *International journal of vehicle systems modelling and testing*, vol. 5, no. 2-3, pp. 124–160, 2010.

[13] T. Chu and C. Chen, “Modelling and model predictive control for a bicycle-rider system,” *Vehicle system dynamics*, vol. 56, no. 1, pp. 128–149, 2018.

[14] G. Lars and P. Jürgen, “Nonlinear model predictive control theory and algorithms,” 2011.

[15] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, “Learning-based model predictive control: Toward safe learning in control,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020.

[16] E. T. Maddalena and C. N. Jones, “Learning non-parametric models with guarantees: A smooth lipschitz regression approach∗∗this work has received support from the swiss national science foundation under the risk project (risk aware data-driven demand response, grant number 200021 175627.” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 965–970, 2020, 21st IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896320316670

[17] M. Tanaskovic, L. Fagiano, and V. Gligorovski, “Adaptive model predictive control for linear time varying mimo systems,” *Automatica*, vol. 105, pp. 237–245, 2019.

[18] G. Schildbach, L. Fagiano, C. Frei, and M. Morari, “The scenario approach for stochastic model predictive control with bounds on closed-loop constraint violations,” *Automatica*, vol. 50, no. 12, pp. 3009–3018, 2014.

[19] M. Mammarella, E. Capello, F. Dabbene, and G. Guglieri, “Sample-based smpc for tracking control of fixed-wing uav,” *IEEE Control Systems Letters*, vol. 2, no. 4, pp. 611–616, 2018.

[20] M. Mammarella, M. Lorenzen, E. Capello, H. Park, F. Dabbene, G. Guglieri, M. Romano, and F. Allgöwer, “An offline-sampling smpc framework with application to autonomous space maneuvers,” *IEEE Transactions on Control Systems Technology*, vol. 28, no. 2, pp. 388–402, 2020.

[21] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, “Provably safe and robust learning-based model predictive control,” *Automatica*, vol. 49, no. 5, pp. 1216–1226, 2013.

[22] L. Sforni, I. Notarnicola, and G. Notarstefano, “Learning-driven nonlinear optimal control via gaussian process regression,” in *2021 60th IEEE CDC*. IEEE, 2021, pp. 4412–4417.

[23] R. Miklos, L. N. Petersen, N. K. Poulsen, C. Utzen, J. B. Jørgensen, and H. H. Niemann, “Greybox model for multistage spray drying plants constrained to small datasets,” *Advanced Control for Applications*, vol. 3, no. 1, p. e61, 2021.

[24] M. Canale, L. Fagiano, and M. Signorile, “Nonlinear model predictive control from data: a set membership approach,” *International Journal of Robust and Nonlinear Control*, vol. 24, no. 1, pp. 123–139, 2014. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rnc.2878

[25] S. Piche, B. Sayyar-Rodsari, D. Johnson, and M. Gerules, “Nonlinear model predictive control using neural networks,” *IEEE Control Systems Magazine*, vol. 20, no. 3, pp. 53–62, 2000.

[26] J. M. Manzano, D. Limon, D. Muñoz de la Peña, and J.-P. Calliess, “Robust learning-based mpc for nonlinear constrained systems,” *Automatica*, vol. 117, p. 108948, 2020.

[27] E. Alcala, O. Sename, V. Puig, and J. Quevedo, “Ts-mpc for autonomous vehicle using a learning approach,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 15 110–15 115, 2020, 21st IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896320326720

[28] E. T. Maddalena, P. Scharnhorst, Y. Jiang, and C. N. Jones, “KPC: Learning-based model predictive control with deterministic guarantees,” in *Conference on Learning for Dynamics and Control*, vol. 144, 2021, pp. 1015–1026.

[29] C. E. Rasmussen, “Gaussian processes in machine learning,” in *Summer school on machine learning.* Springer, 2003, pp. 63–71.

[30] A. Carron, E. Arcari, M. Wermelinger, L. Hewing, M. Hutter, and M. N. Zeilinger, “Data-driven model predictive control for trajectory tracking with a robotic arm,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3758–3765, 2019.

[31] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, “Learning-based model predictive control for autonomous racing,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, 2019.

[32] C. J. Ostafew, A. P. Schoellig, T. D. Barfoot, and J. Collier, “Learning-based nonlinear model predictive control to improve vision-based mobile robot path tracking,” *Journal of Field Robotics*, 2014.

[33] A. Arab and J. Yi, “Safety-guaranteed learning-predictive control for aggressive autonomous vehicle maneuvers,” in *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2020, pp. 1036–1041.

[34] L. Hewing, J. Kabzan, and M. N. Zeilinger, “Cautious model predictive control using gaussian process regression,” *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2736–2743, 2019.

[35] U. Rosolia and F. Borrelli, “Learning model predictive control for iterative tasks. a data-driven control framework,” *IEEE Transactions on Automatic Control*, vol. 63, no. 7, pp. 1883–1896, 2017.

[36] S. Gros and M. Zanon, “Data-driven economic nmpc using reinforcement learning,” *IEEE Transactions on Automatic Control*, vol. 65, no. 2, pp. 636–648, 2019.

[37] J. L. Garriga and M. Soroush, “Model predictive control tuning methods: A review,” *Industrial & Engineering Chemistry Research*, vol. 49, no. 8, pp. 3505–3515, 2010.

[38] M. Mehndiratta, E. Camci, and E. Kayacan, "Automated tuning of nonlinear model predictive controller by reinforcement learning," in *2018 IEEE/RSJ IROS*. IEEE, 2018, pp. 3016–3021.

[39] Q. Lu, R. Kumar, and V. M. Zavala, "Mpc controller tuning using bayesian optimization techniques," *arXiv preprint arXiv:2009.14175*, 2020.

[40] E. Tofighi and A. Mahdizadeh, "Automatic weight determination in nonlinear model predictive control of wind turbines using swarm optimization technique," in *Journal of Physics: Conference Series*, vol. 753. IOP Publishing, 2016, p. 052033.

[41] M. Vallerio, J. Van Impe, and F. Logist, "Tuning of nmpc controllers via multi-objective optimisation," *Computers & Chemical Engineering*, vol. 61, pp. 38–50, 2014.

[42] V. Ramasamy, R. K. Sidharthan, R. Kannan, and G. Muralidharan, "Optimal tuning of model predictive controller weights using genetic algorithm with interactive decision tree for industrial cement kiln process," *Processes*, vol. 7, no. 12, p. 938, 2019.

[43] M. H. Arshad, M. A. Abido, A. Salem, and A. H. Elsayed, "Weighting factors optimization of model predictive torque control of induction motor using nsga-ii with topsis decision making," *Ieee Access*, vol. 7, pp. 177 595–177 606, 2019.

[44] P. R. U. Guazzelli, W. C. de Andrade Pereira, C. M. R. de Oliveira, A. G. de Castro, and M. L. de Aguiar, "Weighting factors optimization of predictive torque control of induction motor by multiobjective genetic algorithm," *IEEE Transactions on Power Electronics*, vol. 34, no. 7, pp. 6628–6638, 2018.

[45] A. Norouzi, H. Heidarifar, H. Borhan, M. Shahbakhti, and C. R. Koch, "Integrating machine learning and model predictive control for automotive applications: A review and future directions," *Engineering Applications of Artificial Intelligence*, vol. 120, p. 105878, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197623000623

[46] E. Picotti, A. D. Libera, R. Carli, and M. Bruschetta, "Continuous- time acceleration modeling through gaussian processes for learning-based nonlinear model predictive control," in *2022 IEEE Conference on Control Technology and Applications (CCTA)*, 2022, pp. 1228–1233.

[47] M. Bruschetta, E. Picotti, A. De Simoi, Y. Chen, A. Beghi, M. Nishimura, Y. Tezuka, and F. Ambrogi, "Real-time nonlinear model predictive control of a virtual motorcycle," *IEEE Transactions on Control Systems Technology*, pp. 1–9, 2020.

[48] E. Picotti, L. Facin, A. Beghi, M. Nishimura, Y. Tezuka, F. Ambrogi, and M. Bruschetta, "Data-driven tuning of a nmpc controller for a virtual motorcycle through genetic algorithm," in *2022 IEEE Conference on Control Technology and Applications (CCTA)*, 2022, pp. 1222–1227.

[49] E. Picotti, F. Bianchin, and M. Bruschetta, "Learning-based nonlinear model predictive control of a virtual motorcycle in real-time," *Control Engineering Practice*, 2023, submitted.

[50] E. Picotti, M. Bruschetta, E. Mion, and A. Beghi, "A nonlinear model-predictive contouring controller for shared control driving assistance in high-performance scenarios," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2022.

[51] J. Rühle, E. Picotti, and M. Bruschetta, "Learning-based nonlinear model predictive control for a high performance virtual driver," in *Conference on Control Technologies and Applications.* IEEE, 2023, submitted.

[52] E. Picotti, E. Mion, A. Dalla Libera, J. Pavlovic, A. Censi, E. Frazzoli, A. Beghi, and M. Bruschetta, "A learning-based nonlinear model predictive controller for a real go-kart based on black-box dynamics modeling through gaussian processes," *IEEE Transactions on Control System Technology*, 2023, accepted.

[53] E. Picotti, A. Dalla Libera, R. Carli, and M. Bruschetta, "Lbmatmpc: an open-source toolbox for gaussian process modeling within learning-based nonlinear model predictive control," in *European Control Conference.* IEEE, 2022.

[54] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[55] Y. Chen, "Algorithms and applications for nonlinear model predictive control with long prediction horizon," Ph.D. dissertation, University of Padova, 2018.

[56] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear mpc and moving horizon estimation," *Nonlinear model predictive control: towards new challenging applications*, pp. 391–417, 2009.

[57] F. Allgöwer and A. Zheng, *Nonlinear model predictive control.* Birkhäuser, 2012, vol. 26.

[58] T. Binder, L. Blank, H. G. Bock, R. Bulirsch, W. Dahmen, M. Diehl, T. Kronseder, W. Marquardt, J. P. Schlöder, and O. von Stryk, "Introduction to model based optimization of chemical processes on moving horizons," *Online optimization of large scale systems*, pp. 295–339, 2001.

[59] R. Quirynen, "Numerical simulation methods for embedded optimization," Ph.D. dissertation, Katholieke Universiteit Leuven, 2017.

[60] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," in *IFAC*, 1984.

[61] M. Diehl, H. G. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer, "Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations," *Journal of Process Control*, vol. 12, no. 4, pp. 577–585, 2002.

[62] S.-P. Han, "Superlinearly convergent variable metric algorithms for general nonlinear programming problems," *Mathematical Programming*, vol. 11, no. 1, pp. 263–282, 1976.

[63] M. J. Powell, "A fast algorithm for nonlinearly constrained optimization calculations," in *Numerical Analysis: Proceedings of the Biennial Conference Held at Dundee, June 28–July 1, 1977.* Springer, 1977, pp. 144–157.

[64] D. Kouzoupis, A. Zanelli, H. Peyrl, and H. J. Ferreau, "Towards proper assessment of qp algorithms for embedded model predictive control," in *2015 European Control Conference (ECC).* IEEE, 2015, pp. 2609–2616.

[65] H. J. Ferreau, S. Almér, R. Verschueren, M. Diehl, D. Frick, A. Domahidi, J. Jerez, G. Stathopoulos, and C. Jones, "Embedded optimization methods for industrial automatic control," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 13 194–13 209, 2017.

[66] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpoases: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[67] G. Frison and M. Diehl, "Hpipm: a high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020.

[68] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM Journal on control and optimization*, vol. 43, no. 5, pp. 1714–1736, 2005.

[69] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear mpc: bridging the gap via the real-time iteration," *International Journal of Control*, vol. 93, no. 1, pp. 62–80, 2020.

[70] M. Zanon, S. Gros, and A. Bemporad, "Practical reinforcement learning of stabilizing economic mpc," in *2019 18th European Control Conference (ECC).* IEEE, 2019, pp. 2258–2263.

[71] D. Piga, M. Forgione, S. Formentin, and A. Bemporad, "Performance-oriented model learning for data-driven mpc design," *IEEE control systems letters*, vol. 3, no. 3, pp. 577–582, 2019.

[72] J. Branke, "Creating robust solutions by means of evolutionary algorithms," in *International Conference on Parallel Problem Solving from Nature.* Springer, 1998, pp. 119–128.

[73] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments-a survey," *IEEE Transactions on evolutionary computation*, vol. 9, no. 3, pp. 303–317, 2005.

[74] Y. Jin and B. Sendhoff, "Trade-off between performance and robustness: An evolutionary multiobjective approach," in *International conference on evolutionary multi-criterion optimization.* Springer, 2003, pp. 237–251.

[75] K. Deb and H. Gupta, "Introducing robustness in multi-objective optimization," *Evolutionary computation*, vol. 14, no. 4, pp. 463–494, 2006.

[76] V. Pareto, *Manuale di economia politica: con una introduzione alla scienza sociale.* Società editrice libraria, 1919, vol. 13.

[77] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[78] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary computation*, vol. 2, no. 3, pp. 221–248, 1994.

[79] M. Buisson-Fenet, F. Solowjow, and S. Trimpe, "Actively learning gaussian process dynamics," in *Learning for dynamics and control.* PMLR, 2020, pp. 5–15.

[80] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When gaussian process meets big data: a review of scalable gps," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4405–4423, Jan. 2020.

[81] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes," in *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, D. van Dyk and M. Welling, Eds., vol. 5. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 16–18 Apr 2009, pp. 567–574.

[82] H. Bijl, J.-W. van Wingerden, T. B. Schön, and M. Verhaegen, "Online sparse gaussian process regression using fitc and pitc approximations," *IFAC-PapersOnLine*, vol. 48, no. 28, pp. 703–708, 2015.

[83] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems*, Y. Weiss, B. Schölkopf, and J. Platt, Eds., vol. 18. MIT Press, 2005. [Online]. Available: https://proceedings.neurips.cc/paper/2005/file/4491777b1aa8b5b32c2e8666dbe1a495-Paper.pdf

[84] A. Datta, S. Banerjee, A. O. Finley, and A. E. Gelfand, "Hierarchical nearest-neighbor gaussian process models for large geostatistical datasets," *Journal of the American Statistical Association*, vol. 111, no. 514, pp. 800–812, 2016.

[85] J. Quinonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression," *The Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.

[86] F. Rossi, A. Lendasse, D. Francois, V. Wertz, and M. Verleysen, "Mutual information for the selection of relevant variables in spectrometric nonlinear modelling," *Chemometrics and Intelligent Laboratory Systems*, vol. 80, no. 2, pp. 215–226, 2006.

[87] K. Schelthoff, C. Jacobi, E. Schlosser, D. Plohmann, M. Janus, and K. Furmans, "Feature selection for waiting time predictions in semiconductor wafer fabs," *IEEE Transactions on Semiconductor Manufacturing*, vol. 35, no. 3, pp. 546–555, 2022.

[88] M. Bauer, M. van der Wilk, and C. E. Rasmussen, "Understanding probabilistic sparse gaussian process approximations," *Advances in neural information processing systems*, vol. 29, 2016.

[89] *VI-BikeRealTime 18.0 Documentation*, VI-grade engineering software & services, 2017.

[90] A. Saccon, J. Hauser, and R. Frezza, "Control of a bicycle using model predictive control strategy," *IFAC Proceedings Volumes*, vol. 37, no. 13, pp. 633–638, 2004.

[91] J. K. Vandiver, "An introduction to lagrange equations," MIT OpenCourseWare. Engineering Dynamics, 2011.

[92] D. Limebeer and M. Massaro, *Dynamics and optimal control of road vehicles.* Oxford University Press, 2018.

[93] N. Getz, "Control of balance for a nonlinear nonholonomic non-minimum phase model of a bicycle," in *Proceedings of 1994 American Control Conference-ACC'94*, vol. 1. IEEE, 1994, pp. 148–151.

[94] H. Pacejka, *Tire and vehicle dynamics.* Elsevier, 2005.

[95] V. Cossalter, *Motorcycle dynamics.* Lulu. com, 2006.

[96] N. van Duijkeren, R. Verschueren, G. Pipeleers, M. Diehl, and J. Swevers, "Path-following nmpc for serial-link robot manipulators using a path-parametric system reformulation," in *2016 European Control Conference (ECC).* IEEE, 2016, pp. 477–482.

[97] M. Gerdts, S. Karrenberg, B. Müller-Beßler, and G. Stock, "Generating locally optimal trajectories for an automatically driven car," *Optimization and Engineering*, vol. 10, no. 4, p. 439, 2009.

[98] Y. Gao, A. Gray, J. V. Frasch, T. Lin, E. Tseng, J. K. Hedrick, and F. Borrelli, "Spatial predictive control for agile semi-autonomous ground vehicles," in *Proceedings of the 11th international symposium on advanced vehicle control*, 2012.

[99] G. Frison, D. K. M. Kufoalor, L. Imsland, and J. B. Jørgensen, "Efficient implementation of solvers for linear model predictive control on embedded devices," in *2014 IEEE Conference on Control Applications (CCA).* IEEE, 2014, pp. 1954–1959.

[100] J. Nocedal and S. Wright, *Numerical optimization.* Springer Science & Business Media, 2006.

[101] R. Frezza, A. Beghi, and A. Saccon, "Model predictive for path following with motorcycles: application to the development of the pilot model for virtual prototyping," in *2004 43rd IEEE Conference on Decision and Control (CDC)*, vol. 1. IEEE, 2004, pp. 767–772.

[102] S. Baskar, S. Tamilselvi and P. Varshini, "Matlab code for constrained nsga ii," 2015.

[103] M. Bruschetta, E. Picotti, A. De Simoi, Y. Chen, A. Beghi, M. Nishimura, Y. Tezuka, and F. Ambrogi, "Real-time nonlinear model predictive control of a virtual motorcycle," *IEEE Transactions on Control Systems Technology*, pp. 1–9, 2020.

[104] Y. Chen, N. Scarabottolo, M. Bruschetta, and A. Beghi, "Efficient move blocking strategy for multiple shooting-based non-linear model predictive control," *IET Control Theory & Applications*, vol. 14, no. 2, pp. 343–351, 2019.

[105] M. Bruschetta, E. Picotti, E. Mion, Y. Chen, A. Beghi, and D. Minen, "A nonlinear model predictive control based virtual driver for high performance driving," in *2019 IEEE Conference on Control Technology and Applications (CCTA)*, 2019, pp. 9–14.

[106] T. D. Gillespie, "Vehicle dynamics," *Warren dale*, pp. 97–98, 1997.

[107] H. Pacejka, "Tyre and vehicle dynamics/hans b. pacejka," pp. 187–189, 2006.

[108] V. Cossalter, M. Da Lio, R. Lot, and L. Fabbri, "A general method for the evaluation of vehicle manoeuvrability with special emphasis on motorcycles," *Vehicle system dynamics*, vol. 31, no. 2, pp. 113–135, 1999.

[109] *VI-CarRealTime 18.0 Documentation*, VI-grade engineering software & services, 2017.

[110] M. N. Zeilinger, C. N. Jones, and M. Morari, "Robust stability properties of soft constrained mpc," in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 5276–5282.

[111] R. Brach and M. Brach, "The tire-force ellipse (friction ellipse) and tire characteristics," SAE Technical Paper, Tech. Rep., 2011.

[112] O. Tracey, "Forward backwards kalman filter," 2022.

[113] E. Mion, "Mpc-based control strategies for human-machine interaction in automotive applications," Ph.D. dissertation, University of Padova, 2021.

[114] R. Suri, "Towards a benchmarking framework for the go-kart platform," Master's thesis, IDSC, ETH Zürich, 2021.

[115] D. Ankenbrand, "Multi-agent simulation for wheel-to-wheel racing," Master's thesis, IDSC, ETH Zürich, 2022.

[116] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3. Kobe, Japan, 2009, p. 5.

[117] Y. Chen, M. Bruschetta, E. Picotti, and A. Beghi, "Matmpc - a matlab based toolbox for real-time nonlinear model predictive control," in *2019 18th European Control Conference (ECC)*, 2019, pp. 3365–3370.

[118] R. Quirynen, M. Vukov, M. Zanon, and M. Diehl, "Autogenerating microsecond solvers for nonlinear mpc: A tutorial using acado integrators," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 685–704, 2015.