IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 24 April 2023, accepted 6 May 2023, date of publication 10 May 2023, date of current version 17 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3274836

RESEARCH ARTICLE

A Physics-Driven Artificial Agent for Online Time-Optimal Vehicle Motion Planning and Control

MATTIA PICCININI[1], (Member, IEEE), SEBASTIANO TADDEI[1,2], MATTEO LARCHER[1], MATTIA PIAZZA[1], AND FRANCESCO BIRAL[1]

[1]Department of Industrial Engineering, University of Trento, 38123 Trento, Italy
[2]Department of Electrical and Information Engineering, Politecnico di Bari, 70125 Bari, Italy

Corresponding author: Mattia Piccinini (mattia.piccinini@unitn.it)

ABSTRACT This paper presents a hierarchical framework with novel analytical and neural physics-driven models, to enable the online planning and tracking of minimum-time maneuvers, for a vehicle with partially-unknown parameters. We introduce a lateral speed prediction model for high-level motion planning with economic nonlinear model predictive control (E-NMPC). A low-level steering controller is developed with a novel feedforward-feedback physics-driven artificial neural network (NN). A longitudinal dynamic model is identified to tune a low-level speed-tracking controller. The high- and low-level control models are identified with an automatic three-step scheme, combining open-loop and closed-loop maneuvers to model the maximum acceleration G-G-v performance constraint for E-NMPC, and to capture the effect of the longitudinal acceleration on the lateral dynamics. The proposed framework is used in a simulation environment, for the online closed-loop control of a highly detailed sedan vehicle simulator, whose parameters are partially-unknown. Two different circuits are adopted to validate the approach, and a robustness analysis is performed by varying the vehicle mass and the load distribution. A minimum-time optimal control problem is solved offline and used for a comparison with the closed-loop results. A video demonstrating both the automatic three-step identification scheme and the motion planning and control results is available at the following link: `https://www.youtube.com/watch?v=xQ_T96IjGP8`.

INDEX TERMS Autonomous racing, model learning, model predictive control (MPC), motion planning, neural networks, trajectory optimization.

I. INTRODUCTION

The development of online minimum-time motion planning and control techniques for autonomous vehicles is a complex and partly unsolved problem [1]. Simulations have been setup [2], [3], [4], [5], [6] to validate the algorithms in controlled environments, before the (often risky and expensive) experimental testing [7], [8], [9], [10], [11], [12]. The planning and control tasks become even more involved when most of the vehicle parameters are not known. This scenario is quite common in real-world applications: many vehicle parameters (e.g. vehicle mass, tire and suspension characteristics, center of mass position) may vary in operation and throughout the lifespan of a vehicle, and their measurement (offline or online) is often unaffordable. Identifying the numerous parameters that define the tires, suspensions, powertrain, braking, steering, and aerodynamics can be costly and time-consuming, as is evident from the extensive research on tire characterization by Pacejka [13]. Our approach is to identify and learn analytical and neural physics-based models, to be applied for online minimum-time motion planning and control. We apply our framework to drive a vehicle simulator (VS), whose parameters are almost entirely unknown in advance. In comparison with deep neural network approaches, the physical structure of the proposed models to be identified increases the overall interpretability and generalization capability of the framework, it enables the use

The associate editor coordinating the review of this manuscript and approving it for publication was Jjun Cheng.

of fewer learnable parameters and decreases the size of the training sets.

A literature review is offered in the following subsection, after which the main contribution of this paper is presented.

A. RELATED WORK

Among the motion planning techniques for autonomous vehicles proposed in the literature [1], [14], optimal control (OC)-based approaches have emerged as one of the most suitable for the class of minimum-time planning problems. Nonlinear model predictive control (NMPC) [15] is a popular implementation of optimal control, that uses a receding-horizon approach. Two main variants of NMPC can be distinguished: tracking NMPC and economic NMPC (E-NMPC) [16]. The former aims at steering the plant[1] towards a target setpoint or trajectory, while the latter optimizes directly an economic performance indicator, without relying on a reference trajectory to be tracked. Minimum-time NMPC for trajectory optimization can be seen as a form of E-NMPC, in which the economic cost is the travel time.

To implement online minimum-time E-NMPC or tracking NMPC, it is often necessary to trade off the accuracy and nonlinearity of the vehicle model.

In the context of online minimum-time vehicle motion planning and/or tracking, many authors used tracking NMPC, and few adopted E-NMPC. High-level tracking NMPC was used for example in [5], [9], and [12], to track online a pre-computed race-line. E-NMPC planners were employed in [2], [6], [7], and [17], yet with very simplified vehicle models. Low-level controllers were developed to track the high-level state trajectories, e.g. using NMPC with more complex vehicle models [5], [9], [18], neural networks [2], [19], proportional-integral-derivative (PID) controllers [20] or other methods.

A point-mass model was adopted in [5] to compute a terminal velocity constraint for a low-level NMPC controller, to track a pre-computed minimum curvature path. A single-track vehicle model was used in [9] to compute offline a minimum time trajectory, which they tracked online with a Formula SAE car. Using simple kinematic models for the yaw rate dynamics, point-mass models were constrained with G-G-v diagrams[2] in [6], [8], [21], [22], and [23]. A Tube-MPC formulation was adopted in [8] and [24] for motion planning, with a simple linear point-mass model, and applied in the Indy Autonomous Challenge. The longitudinal load transfer was modeled in [23] to improve a G-G diagram.[3] The impact of model fidelity for minimum-time trajectory optimization was analyzed in [25].

[1]The term plant is used with its meaning in control theory (*i.e.,* the combination of actuators and process).

[2]The G-G-v diagram is a 3D graphical representation of the performance envelope of a vehicle, where the x, y, and z axes correspond to the maximum vehicle's longitudinal acceleration, lateral acceleration, and longitudinal velocity.

[3]The G-G diagram is a 2D graphical representation of the performance envelope of a vehicle, where the x and y axes correspond to the maximum vehicle's longitudinal and lateral accelerations.

Some researchers used machine learning tools to augment NMPC tracking controllers. The authors in [11] and [26] proposed an online improvement of a nominal vehicle model for NMPC using Gaussian processes. The cost function and convex safe sets for a model predictive control (MPC) problem were iteratively learned in [10], to minimize the travel time. In [4] and [27], neural prediction models of the vehicle lateral dynamics were used for NMPC tracking problems.

While the works cited above used a direct OC approach for both tracking and economic NMPC, our previous studies [2], [7] employed an indirect OC technique for online minimum-time E-NMPC. Interested readers can find a comparison of direct and indirect methods for (offline) minimum-time OC in [28]. In [7], we used a simplified vehicle model to control a 1:8 remote control (RC) car near the handling limits, along a circuit, modeling the nonlinear handling diagram and a G-G-v constraint. Recently, in [2] we presented a hierarchical framework for online minimum-time motion planning and control of a black-box vehicle model. An offline learning phase enabled the identification of the high- and low-level models. However, the lateral speed was not modeled in the high-level E-NMPC, which may decrease the feasibility of the planned trajectory and lead to trajectory tracking errors [25].

Some authors recently used robust control approaches to deal with model uncertainties, for motion planning [8], [24], trajectory tracking [29] and low-level stabilization [30].

Other authors used artificial neural networks, hereafter named simply neural networks (NNs), as low-level controllers of the longitudinal or lateral vehicle dynamics. A survey of deep learning control methods for autonomous vehicles can be found in [1] and [31]. A comparison of a basic feedforward and a convolutional NN was presented in [32], to compute the steering angle and the driving torques for path tracking at constant speed. A stochastic policy was learned in [33] to imitate the human behavior near the handling limits. In [2], we used a recurrent NN as a steering controller for a black-box vehicle model, to track minimum-time maneuvers. A two-layer feedforward NN was adopted in [34] for path tracking, in combination with a feedback controller.

Recently, some authors started developing neural networks with a structure inspired by the physical laws. In [19] and [35], neural models were devised with an architecture based on the vehicle dynamic laws, respectively in pure longitudinal and lateral conditions. In comparison with general-purpose NNs, the networks with a physics-based structure resulted in a better interpretability and improved performance, for the same amount of learnable parameters.

To the best of the authors' knowledge, the cited literature papers have limitations in at least one of the following aspects:

1) NNs with a physics-driven structure have never been developed to control the combined longitudinal-lateral vehicle dynamics, near the handling limits.

2) Many papers on time-optimal motion planning and control assume prior knowledge or identification of vehicle and tire parameters.
3) The authors performing online minimum-time trajectory optimization with E-NMPC on medium-long planning horizons, like [2], [6], and [7], used very simple vehicle models, that poorly predict the transient trajectory curvature of a more complex vehicle simulator or real car.

B. PAPER CONTRIBUTION AND STRUCTURE

In this paper, we present an artificial agent to automatically identify the dynamics, plan and execute online time-optimal maneuvers with a partially-unknown vehicle simulator, using white-box physics-driven controllers arranged in a hierarchical structure.

The contributions of this paper are threefold:

1) We present a kineto-dynamical vehicle model for high-level time-optimal motion planning with E-NMPC. Such a model extends the one described in [2]: we add a novel lateral speed prediction model, and a generalized formulation of the maximum performance G-G-v diagram.
2) A low-level neural feedforward steering controller is devised with an original physically explainable structure, inspired by the nonlinear laws of vehicle dynamics. In comparison with a general-purpose recurrent NN, the physical structure of our network enables the use of fewer learnable parameters, and it provides an increased generalization capability. Our feedforward neural network is complemented with a feedback steering controller, to further improve the modeling of the steering dynamics, while compensating for model and environment uncertainties.
3) A three-step automatic identification and learning procedure is developed, combining open- and closed-loop maneuvers to (a) identify/learn the high- and low-level control models, (b) parameterize the optimal G-G-v constraint for the E-NMPC formulation, and (c) capture the complex dependency of the lateral dynamics on the longitudinal acceleration.

The physics-driven architecture of the high- and low-level models results in an interpretable white-box planning and control framework. In comparison with deep learning, the proposed approach is less prone to overfitting, providing accurate control actions even when tested on racetracks never seen during training. Moreover, our artificial agent shows a good robustness to changes in the vehicle mass and load distribution. The physics-driven approach results in models and controllers with a low computational complexity, which makes it possible to run the agent on standard embedded computers, without the support of a GPU.

The proposed three-step automatic identification scheme is inspired by the way in which human race drivers progressively learn the vehicle dynamic limits, and how to reach them. After a first round of predefined open-loop maneuvers, two sessions of laps are performed on a given circuit (closed-loop maneuvers), to incrementally improve the ability of the artificial agent to drive a vehicle faster and faster. In a similar way, human professional drivers usually perform rounds of laps, in which they progressively improve their knowledge about a car, until they can drive it near its limits. Note that we use the generic term *identification*, even if our approach includes a mix of parameter identification, neural network training and iterative learning.

This paper is organized as follows. Section II presents the characteristics of the vehicle simulator to be controlled. Section III outlines the planning and control framework, from the high-level E-NMPC formulation (III-A) to the low-level steering and longitudinal controllers (III-B and III-D). Section IV describes the methods devised to identify and learn the high- and low-level control models. Section V compares the novel physics-driven neural steering controllers with traditional recurrent neural networks. Section VI analyzes the main results, while Section VII is dedicated to our conclusions and directions of future work.

II. VEHICLE SIMULATOR TO BE CONTROLLED

A. STRUCTURE OF THE VEHICLE SIMULATOR

In this section, we provide some insights into the vehicle simulator, whose dynamics need to be identified and controlled with minimum-time maneuvers.

The behavior of a vehicle near its handling limits is affected by factors that can be neglected in normal driving conditions. For example, tires work in the nonlinear region of the force-slip characteristics, the chassis motion (roll, pitch, heave) influences the vehicle dynamics and the aerodynamic effects, and the suspensions compliance has a non negligible effect. The model simulating the vehicle physics for minimum-time applications must be able to accurately describe the vehicle behavior close to the limits.

We implemented a vehicle simulator (VS) with a high-fidelity 14-degree-of-freedom multibody structure. The tire forces and moments are expressed with a Magic Formula 6.2 formulation [13]; the VS can be driven on three-dimensional roads [36]; and sophisticated powertrain and steering models are used. The VS model equations are symbolically manipulated to obtain an efficient formulation, enabling real-time hardware-in-the-loop (HIL) and driver-in-the-loop (DIL) simulations.

The VS is integrated into the driving simulation framework developed by AnteMotion Srl[4] (Fig. 1). The reader is referred to [37, Sections V-VI] for further implementation details about the VS.

We model a front-wheel-drive (FWD) sedan car, with a single electric motor and an open mechanical differential. Kinematic and compliance maps model the vehicle suspensions. Table 1 shows the main parameters of the VS.

[4]AnteMotion Srl website: https://antemotion.com.

FIGURE 1. Vehicle driving simulator in the Autonomous Driving Laboratory of the University of Trento (Italy). The simulator implements the presented VS model in a real-time hardware.

TABLE 1. Main parameters of the vehicle simulator.

Parameter	Symbol	Value
Vehicle total mass	m	1296 kg
Center of mass (CoM) height	h_G	0.285 m
Front axle distance from the CoM	L_1	1.23 m
Rear axle distance from the CoM	L_2	1.45 m
Wheelbase	L	2.68 m
Yaw inertia	I_z	1400 kg m^2
Track width (front)	W_1	1.525 m
Track width (rear)	W_2	1.518 m
Steering ratio	τ_d	20
Maximum torque at wheel	$T_{w_{\max}}$	1200 N m
Maximum motor power	$P_{\max}$	150 kW
Wheel inertia (around spin axis)	I_w	1.42 kg m^2

The presented control framework drives the VS with the following inputs: a signal $p \in [-1, 1]$ for the pedals position (throttle for $p > 0$, brake for $p < 0$), and the steering wheel angle δ_D.

For the purposes of this work, the internal structure of the vehicle simulator is considered to be partially unknown. The adverb *partially* stems from the assumption that only few vehicle parameters are available, namely the outer shape of the car and the set $\mathcal{A} = \{m, h_G, I_w, L_1, L_2\}$ (see Table 1). However, we wish to underline that these known quantities are solely used to ease the tuning of a speed-tracking PID controller (Section III-D), by means of an identified model-based design. If the parameters in $\mathcal{A}$ were not known, the speed-tracking controller could still be tuned with a model-free approach (e.g. iterative learning [38] or reinforcement learning [39]), or using the simplified longitudinal model that we proposed in [2]. Moreover, we will show that our control framework is robust to variations of the parameters in $\mathcal{A}$.

B. VEHICLE BEHAVIOR NEAR THE LIMITS

The vehicle simulator does not have an anti-lock braking system (ABS), nor a traction control (TC). This section motivates the difficulty of driving a vehicle without such control systems. We intentionally disable the ABS and the TC, since

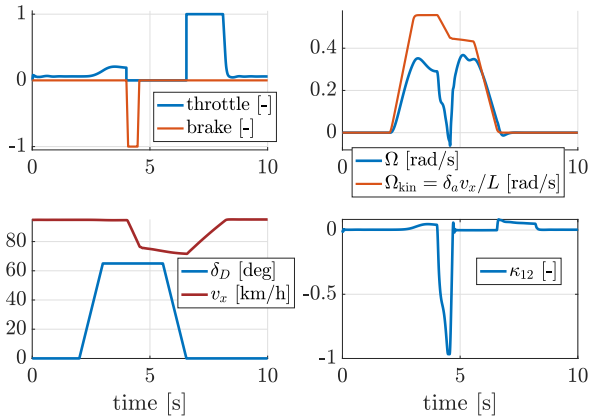


FIGURE 2. Combined steering-braking maneuver, performed at an initial speed $v_x = 95$ km/h and with a steering wheel angle $\delta_D = 65°$. The longitudinal slip for the inner front tire κ_{12} approaches -1 (wheel lock) when the vehicle brakes and steers simultaneously. A sharp decrease of yaw rate Ω happens during wheel lock, which corresponds to a strong reduction of the steering capabilities of the vehicle.

our aim is to develop an autonomous driver able to avoid the wheel lock and wheel spin conditions, which would harm the lateral controllability of the vehicle, and therefore result in a deterioration of performance. As depicted in Fig. 2, some wheels may lock during combined steering-braking maneuvers. Fig. 2 shows that the yaw rate (Ω) dynamics changes considerably when the longitudinal slip κ_{12} of the inner front tire approaches -1 (wheel lock). The kinematic yaw rate $\Omega_{\text{kin}} = \delta_a v_x / L$, frequently used by kinematic motion planners, is also shown for a comparison, with $\{\delta_a, v_x, L\}$ being the steering angle at the front wheels, the forward speed and the wheelbase. The actual yaw rate Ω is a lot smaller than Ω_{kin}, indicating that the vehicle cannot even negotiate a corner during wheel lock, and that the longitudinal acceleration a_x drastically changes the steering characteristics. Similarly, the driving wheels may spin if the vehicle steers and accelerates, due to the absence of a TC.

The previous example shows a pitfall of purely kinematic motion planners, which may yield infeasible reference trajectories for the low-level tracking controllers.

We devise an automatic identification scheme (Section IV) to ensure certain vehicle controllability requirements and avoid the wheel lock and spin conditions. We remark that maneuvers with excessive wheel longitudinal slip are usually not time-optimal, due to the reduction of lateral controllability of the vehicle.

III. CONTROL FRAMEWORK

Fig. 3 depicts the overall hierarchical motion planning and control framework, whose pseudo-code is given in Algorithm 1. Let us denote with $\hat{\mathrm{x}}_k$ the vector of states of the vehicle simulator at the time step k. Using the current state $\hat{\mathrm{x}}_k$ and the road geometry, the high-level E-NMPC motion planner performs the receding-horizon trajectory optimization every $f_{\text{MP}}^{-1} = 80$ ms (line 6 in Algorithm 1). The longitudinal

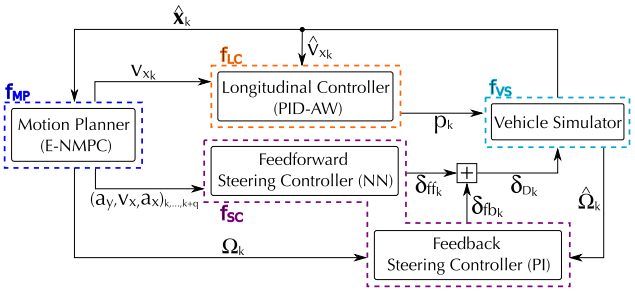


FIGURE 3. Hierarchical framework for motion planning and control.

Algorithm 1 Motion Planning and Control Framework

1: **Input:** $\hat{x}_0$, road_geom
2: $t_{step} \leftarrow 1/f_{VS}$
3: $k \leftarrow 0$
4: **for** $t = 0 : t_{step} : t_{end}$ **do**
5: **if** $(1/t) \bmod f_{MP} = 0$ **then**
6: $[a_y, v_x, a_x, \Omega]_{k,\ldots,k+q} \leftarrow$ E-NMPC($\hat{x}_k$, road_geom)
7: **end if**
8: **if** $(1/t) \bmod f_{LC} = 0$ **then**
9: $p_k \leftarrow$ PID-AW($v_{x_k}, \hat{v}_{x_k}$)
10: **end if**
11: **if** $(1/t) \bmod f_{SC} = 0$ **then**
12: $\delta_{ff_k} \leftarrow$ NN($[a_y, v_x, a_x]_{k,\ldots,k+q}$)
13: $\delta_{fb_k} \leftarrow$ PI($\Omega_k, \hat{\Omega}_k$)
14: $\delta_{D_k} \leftarrow \delta_{ff_k} + \delta_{fb_k}$
15: **end if**
16: **if** $(1/t) \bmod f_{VS} = 0$ **then**
17: $\hat{x}_{k+1} \leftarrow$ Vehicle_Simulator(p_k, δ_{D_k})
18: **end if**
19: $k \leftarrow k + 1$
20: **end for**

controller computes the requested (throttle/brake) pedal p_k every $f_{LC}^{-1} = 1$ ms, using the target E-NMPC speed v_{x_k} and the real vehicle speed $\hat{v}_{x_k}$ (line 9). The feedforward neural steering controller calculates the steering angle δ_{ff_k} every $f_{SC}^{-1} = 1$ ms, using windows of future E-NMPC predictions of lateral acceleration a_y, longitudinal speed v_x and longitudinal acceleration a_x (line 12). The target yaw rate Ω_k and the actual value $\hat{\Omega}_k$ are the inputs of the feedback steering controller, computing δ_{fb_k} (line 13). The requested steering wheel angle $\delta_{D_k} = \delta_{ff_k} + \delta_{fb_k}$ (line 14) is fed to the vehicle simulator. The vehicle simulator runs every $f_{VS}^{-1} = 1$ ms, taking as input the requested pedal p_k and steering wheel angle δ_{D_k}, and returning the vector of states $\hat{x}_{k+1}$ at the next time step (line 17).

A. HIGH-LEVEL MOTION PLANNER (E-NMPC)

1) DYNAMIC MODEL FOR E-NMPC

We extend the kineto-dynamical vehicle model presented in [2] for minimum-time E-NMPC, introducing the lateral velocity (v_y) dynamics:

$$\begin{cases} \dot{v}_x(t) = a_x(t) & \text{(1a)} \\ \tau_{a_x}\dot{a}_x(t) + a_x(t) = a_{x0}(t) & \text{(1b)} \\ \tau_\Omega(v_x(t))\dot{\Omega}(t) + \Omega(t) = \Omega_{0_s}(t) \cdot \Omega_{\max_s}(v_x(t)) & \text{(1c)} \\ \tau_{v_y}(v_x(t))\dot{v}_y(t) + v_y(t) = F_{v_y}(a_y(t), v_x(t), a_x(t)) & \text{(1d)} \end{cases}$$

The longitudinal acceleration a_x is given by the time derivative of the forward speed v_x (1a). a_x has a first order dynamic evolution (1b) with respect to the longitudinal control a_{x0}, and τ_{a_x} is the corresponding time constant.

$\Omega_{0_s} \in [-1, 1]$ is a scaled yaw rate control, and the weighting function $\Omega_{\max_s}(v_x(t))$ is computed as:

$$\begin{cases} v_{x_{PW}}(v_x) = \text{pPart}(v_x - v_{x_{th}}) + v_{x_{th}} & \text{(2a)} \\ \Omega_{\max_s}(v_x(t)) = \dfrac{a_{y_M}(v_x(t))}{v_{x_{PW}}(v_x(t))} & \text{(2b)} \end{cases}$$

where $a_{y_M}(v_x)$ is a polynomial function of v_x, providing the maximum lateral acceleration (Section III-A2). $v_{x_{PW}}(v_x)$ is a smooth piecewise linear function, yielding v_x for $v_x \geq v_{x_{th}}$, and $v_{x_{th}}$ for $v_x < v_{x_{th}}$ (thus avoiding a division by zero in (2b) if $v_x = 0$). pPart($\cdot$) in (2a) is a smooth function computing the positive part of its argument, defined in Appendix B. The first order model (1c) regulates the yaw rate Ω dynamics, with the time constant $\tau_\Omega(v_x)$ being a polynomial function of v_x (Section IV-A3.a).

a: NOVEL LATERAL SPEED PREDICTION MODEL

The major novelty in (1a) with respect to [2] is the prediction model (1d) for the lateral speed v_y. An accurate model for v_y is beneficial to correctly predict the transient trajectory curvature $\rho = (\dot{\beta} + \Omega)/v_G$, with $\beta = \text{atan}(v_y/v_x)$ being the chassis side slip angle and $v_G = \sqrt{v_x^2 + v_y^2}$.

To design (1d), we first consider the pure lateral dynamics, while the effect of a_x is modeled in a second stage.

When $a_x \approx 0$ and in almost steady-state conditions, the behavior of v_y can be expressed as a function of the lateral acceleration $a_y = \Omega v_x$ and v_x. Using the vehicle simulator (Section II), a sequence of low-frequency sine steering tests is performed at different values of speed, reaching the largest feasible lateral accelerations (more details about such maneuvers are given in Section IV-A3). The resulting quantities $\{a_y, v_x, v_y\}$ are plotted in Fig. 4(a), to show what we call nonlinear *lateral velocity diagram* (LVD). The LVD describes the almost steady-state behavior of v_y, as a function of a_y and v_x. We design an approximation model $v_y = H_{v_y}(a_y, v_x)$ for the LVD using polynomials:

$$\begin{cases} f_{a_{yi}}(v_x) = \sum_{j=0}^{4} f_{v_{x_{ji}}} v_x^j, \quad i \in \{1, 3, 5\} & \text{(3a)} \\ v_y = H_{v_y}(a_y, v_x) = f_{a_{y1}}(v_x) \cdot a_y + f_{a_{y3}}(v_x) \cdot a_y^3 + f_{a_{y5}}(v_x) \cdot a_y^5 & \text{(3b)} \end{cases}$$

The dependency of $H_{v_y}(a_y, v_x)$ on a_y is expressed as a 5^{th} order polynomial (3b), with only odd powers due to the diagram symmetry. The functions $f_{a_{yi}}(v_x)$ in (3b), $i \in \{1, 3, 5\}$,

are in turn polynomial functions of v_x (3a), with $f_{v_{xji}}$ being the tunable coefficients of the polynomial, $j \in \{0, 1, \ldots, 4\}$. The degrees of the polynomials in $H_{v_y}(\cdot)$ and $f_{a_{yi}}(\cdot)$ are optimized with experiments. The number of tunable parameters in the $H_{v_y}(\cdot)$ function is $3 \cdot 5 = 15$.

We proceed by extending the LVD to model the effect of the longitudinal acceleration a_x. Equation (3b) is augmented with additional terms, that multiply the functions $f_{a_{yi}}(v_x)$, $i \in \{1, 3, 5\}$, to consider the contribution of a_x:

$$\begin{aligned} v_y = F_{v_y}(a_y, v_x, a_x) = & f_{a_{y1}}(v_x) \cdot (1 + b_1 a_x + b_2 a_x^2) \cdot a_y \\ & + f_{a_{y3}}(v_x) \cdot (1 + b_3 a_x + b_4 a_x^2) \cdot a_y^3 \\ & + f_{a_{y5}}(v_x) \cdot (1 + b_5 a_x + b_6 a_x^2) \cdot a_y^5 \end{aligned} \tag{4}$$

with $\{b_1, b_2, \ldots, b_6\}$ being tunable parameters. Second-order polynomials of the type $1 + b_i a_x + b_{i+1} a_x^2$, $i \in \{1, 3, 5\}$, were found to be a good compromise between accuracy and computational complexity, to model the effect of a_x.

We design the function $F_{v_y}(\cdot)$ in (4) as an extension of $H_{v_y}(\cdot)$ in (3b), so that $F_{v_y}(a_y, v_x, a_x = 0) = H_{v_y}(a_y, v_x)$: the new terms of $F_{v_y}(\cdot)$ depending on a_x do not change the shape of the pure lateral LVD for $a_x = 0$.

Note that the model (4) could be equivalently rewritten with a neural network formalism, for example using the idea of polynomial neural networks recently proposed in [40].

The total number of tunable parameters in the augmented function $F_{v_y}(\cdot)$ is $3 \cdot (5 + 2) = 21$.

The nonlinear function $F_{v_y}(a_y, v_x, a_x)$ is integrated in the E-NMPC model (1d). The function $F_{v_y}(\cdot)$ captures the quasi steady-state behavior of v_y, while its transient evolution is modeled with the first-order dynamics (1d), with the time constant $\tau_{v_y}(v_x)$ being a polynomial function of v_x.

The identification of the functions $\{H_{v_y}(\cdot), \tau_{v_y}(\cdot), F_{v_y}(\cdot)\}$ is explained in Sections IV-A3 and IV-C. Fig. 4(b) plots the approximation of the lateral velocity diagram, using the identified model (1d).

The impact of the novel lateral speed model on the trajectory tracking performance will be discussed in Section VI-B.

The proposed formulation (1c,1d) of the yaw rate and lateral velocity dynamics is numerically efficient for E-NMPC, and it differs from the classical single-track model [41]: it accurately captures the nonlinear transient dynamics without requiring complex (and expensive to measure) tire and suspension parameters, and at the same time without resorting to generic deep neural networks. Conversely, the authors using single-track models for MPC, like [9], need to simplify the lateral tire forces and slip formulation to preserve the real-time feasibility, while requiring the previous knowledge or measurement of the tire parameters. Other authors, like [5], [6], and [23] adopted point-mass models for online MPC, but they did not model the yaw rate and the lateral speed dynamics.

Our model is kineto-dynamical, in the sense that it combines the kinematic equation (1a) with the equations (1c,1d), which implicitly model vehicle dynamic characteristics. Specifically, the shape of the lateral velocity diagram, modeled by $F_{v_y}(\cdot)$ in (1d), depends on the dynamics of a certain vehicle. Similarly, the functions $\tau_\Omega(\cdot)$ and $\tau_{v_y}(\cdot)$ in (1c,1d) capture the transient evolution of Ω and v_y, which in turn depends on the vehicle dynamics.

b: CURVILINEAR COORDINATES

The curvilinear coordinates $\{\zeta, n, \xi\}$ describe the vehicle pose with respect to the road center-line [42], [43]:

$$\begin{cases} \dot{\zeta}(t) = v_\zeta(t) = \dfrac{v_x(t)\cos(\xi(t)) - v_y(t)\sin(\xi(t))}{1 - n(t)\kappa_r(\zeta(t))} & \text{(5a)} \\ \dot{n}(t) = v_x(t)\sin(\xi(t)) + v_y(t)\cos(\xi(t)) & \text{(5b)} \\ \dot{\xi}(t) = \Omega(t) + \\ \quad - \kappa_r(\zeta(t))\left(\dfrac{v_x(t)\cos(\xi(t)) - v_y(t)\sin(\xi(t))}{1 - n(t)\kappa_r(\zeta(t))}\right) & \text{(5c)} \end{cases}$$

As depicted in Fig. 5, ζ is the curvilinear abscissa of the road middle-line, n is the lateral displacement of the vehicle center with respect to the middle-line, and the relative yaw angle is $\xi = \psi - \theta$, with ψ being the vehicle yaw angle and θ being the path heading angle. In (5a), $\kappa_r(\zeta)$ is the local path curvature.

2) E-NMPC PROBLEM

The E-NMPC is formulated as a minimum-time optimal control problem:

$$\begin{aligned} \min \ \boldsymbol{u} \in \mathcal{U}\Big(& \sum_{j=1}^{N_x} \boldsymbol{I}_j \big(\boldsymbol{x}_j(\zeta_i) - \boldsymbol{x}_{0_j}\big)^2 + \boldsymbol{F}_j \big(\boldsymbol{x}_j(\zeta_f) - \boldsymbol{x}_{f_j}\big)^2 \Big) \\ & + \int_{\zeta_i}^{\zeta_f} \frac{w_T}{v_\zeta(\zeta)} \mathrm{d}\zeta \end{aligned} \tag{6a}$$

$$\text{s.t} \begin{cases} \dfrac{\mathrm{d}\boldsymbol{x}(\zeta)}{\mathrm{d}\zeta} = \dfrac{\boldsymbol{f}(\boldsymbol{x}(\zeta), \boldsymbol{u}(\zeta))}{v_\zeta(\zeta)} & \text{(6b)} \\ \boldsymbol{b}(\boldsymbol{x}(\zeta_i), \boldsymbol{u}(\zeta_i)) = 0 & \text{(6c)} \\ \boldsymbol{c}(\boldsymbol{x}(\zeta), \boldsymbol{u}(\zeta)) \geq 0 & \text{(6d)} \end{cases}$$

Since the final travel time T is unknown, the independent variable of the problem is chosen to be the path curvilinear abscissa ζ, rather than time t. ζ ranges in the fixed and known interval $[\zeta_i, \zeta_f]$ [43].

The integral term in (6a) is used to minimize the travel time, with the weight w_T. The dynamical constraint $\boldsymbol{f}$ in (6a) consists of the vehicle model equations (1a) and (5a). The model has $N_x = 6$ states, collected in $\boldsymbol{x} = \{v_x, a_x, \Omega, v_y, n, \xi\}$, while the controls $\boldsymbol{u}$ are $\{a_{x0}, \Omega_{0_s}\}$.

Soft initial conditions are set for the states $\{a_x, n\}$, using the terms in (6a) multiplying the vector of weights $\boldsymbol{I}$. Strict initial conditions are instead set for the states $\{v_x, \Omega, v_y, \xi\}$, with the equality constraints (6a).

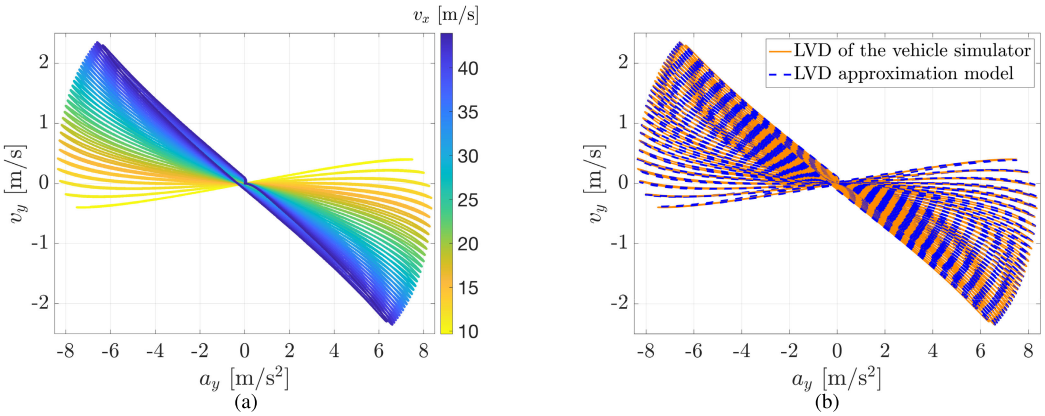


FIGURE 4. (a) Lateral velocity diagram (LVD) for the vehicle simulator, and (b) approximation of the diagram using the model (1d).

The inequality constraints in (6a) are the following:

$$\begin{cases} n(\zeta) - \dfrac{W}{2} \geq -\mathrm{M}_R(\zeta), \qquad n(\zeta) + \dfrac{W}{2} \leq \mathrm{M}_L(\zeta) & \text{(7a)} \\ a_{x0_{\min}} \leq a_{x0}(\zeta) \leq a_{x0_{\max}}, \qquad -1 \leq \Omega_{0_s}(\zeta) \leq 1 & \text{(7b)} \\ \mathrm{pSgn}\big(a_x(\zeta) - a_{x_{\mathrm{of}}}\big) \cdot S\big(a_x(\zeta), a_y(\zeta), v_x(\zeta), a_{x_{\mathrm{M}}}\big) + \\ \quad - \mathrm{nSgn}\big(a_x(\zeta) - a_{x_{\mathrm{of}}}\big) \cdot S\big(a_x(\zeta), a_y(\zeta), v_x(\zeta), a_{x_{\mathrm{m}}}\big) \leq 1 & \text{(7c)} \\ \left[\dfrac{\mathrm{absReg}(a_y(\zeta) + a_{y_c})}{a_{y_b}}\right]^{n_b} & \text{(7d)} \\ \quad + \left[\dfrac{\mathrm{absReg}(a_x(\zeta) + a_{x_c})}{a_{x_b}}\right]^{n_b} \geq 1 & \text{(7e)} \\ \left[\dfrac{\mathrm{absReg}(a_y(\zeta) - a_{y_c})}{a_{y_b}}\right]^{n_b} & \text{(7f)} \\ \quad + \left[\dfrac{\mathrm{absReg}(a_x(\zeta) + a_{x_c})}{a_{x_b}}\right]^{n_b} \geq 1 & \text{(7g)} \end{cases}$$

The road margins constraints are imposed in (7a), with W being the vehicle track width, and $\{M_R(\zeta), M_L(\zeta)\}$ being the distances from the circuit center line to the right and left boundaries.

A G-G-v diagram is defined with (7c,7e,7g), to constrain the accelerations $\{a_x, a_y\}$, as a function of v_x. (7c) has the superelliptic formulation of [2]: $\{\mathrm{pSgn}(\cdot), \mathrm{nSgn}(\cdot)\}$ are smooth functions – defined in Appendix B – enabling the use of superellipses with different shapes, for the combined acceleration and braking performance. The $S(\cdot)$ function in (7c) represents a speed-dependent superellipse:

$$S(a_x, a_y, v_x, a_{x_s}) = \left(\frac{\mathrm{absReg}(a_y)}{a_{y_{\mathrm{M}}}(v_x)}\right)^{n_g} + \left(\frac{\mathrm{absReg}\big(a_x - a_{x_{\mathrm{of}}}(v_x)\big)}{a_{x_s}(v_x)}\right)^{n_g} \quad (8)$$

In (7c) and (8), the functions $\{a_{x_{\mathrm{M}}}(\cdot), a_{x_{\mathrm{m}}}(\cdot), a_{y_{\mathrm{M}}}(\cdot), a_{x_{\mathrm{of}}}(\cdot)\}$ – defined as polynomials in v_x – and the exponent n_g are fitted to best model the G-G-v diagram, obtained experimentally. The smooth absolute value function $\mathrm{absReg}(\cdot)$ is defined in Appendix B. We introduce the additional superelliptic constraints (7e,7g) for a better modeling of the combined braking performance, which is conceptually similar to [23] and [44]. In (7e,7g), the only tunable parameter is the exponent n_b, while the parameters $\{a_{x_c}, a_{y_c}, a_{x_b}, a_{y_b}\}$ are determined with relations depending on the functions $\{a_{x_{\mathrm{m}}}(v_x), a_{y_{\mathrm{M}}}(v_x)\}$ in (7c,8).

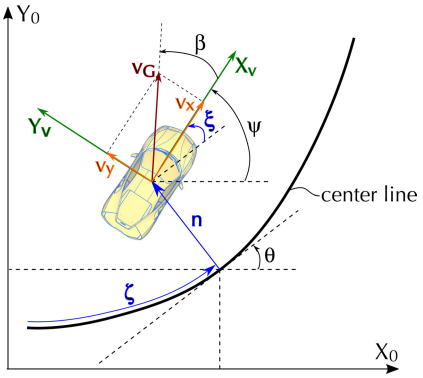


FIGURE 5. Curvilinear coordinates $\{\zeta, n, \xi\}$ and velocity components $\{v_x, v_y\}$ in the chassis reference frame.

An example of G-G-v constraint defined using (7c,7e,7g) is shown in Fig. 6. The G-G-v diagram implicitly contains the saturation effects due to tires, suspensions, aerodynamics and powertrain.

The automatic identification procedure of Section IV-B1 adapts the parameters of (7c,7e,7g) to the identified vehicle performance, and it determines if the additional constraints (7e,7g) need to be activated or not, depending on the identified characteristics of the vehicle to be controlled.

$\{a_{x0_{\min}}, a_{x0_{\max}}\}$ in (7b) are control bounds, derived from (7c).

The E-NMPC problems are solved online, with a long prediction horizon of 300 m (corresponding to more than 6.8 s). A finer discretization is performed for the first 5 m of the horizon, whose total number of mesh points is 346.

As in [2], an offline OC problem – named MLT (for minimum lap time) – is setup with the same structure and

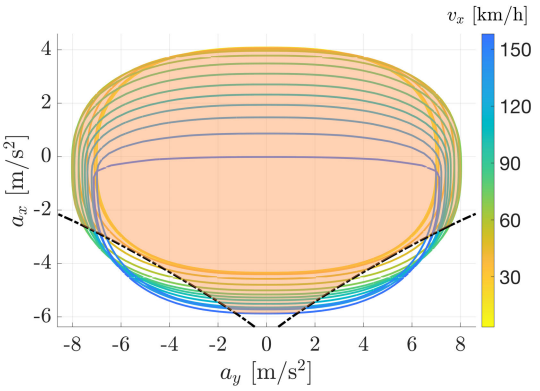


FIGURE 6. Example of G-G-v constraint. The colored superellipses (functions of v_x) are the base constraint (7c), while the two dash-dotted black curves are the additional constraints (7e,7g). The achievable accelerations are in the orange shaded volume.

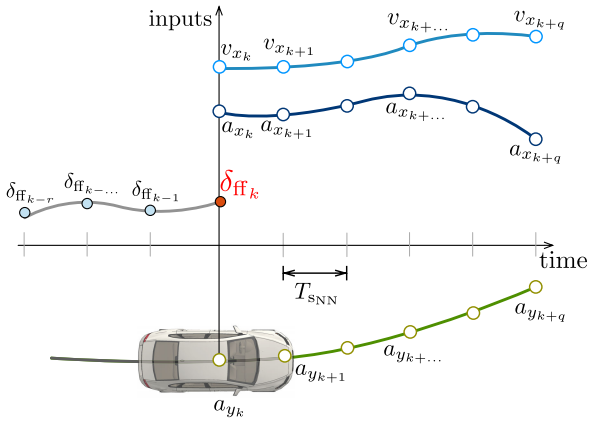


FIGURE 7. Schematic of the inputs and output of the neural steering controller. **Inputs:** future predicted values of lateral acceleration a_y, longitudinal speed v_x, longitudinal acceleration a_x, and past steering angles $\{\delta_{\mathrm{ff}_{k-r}}, \ldots, \delta_{\mathrm{ff}_{k-1}}\}$. **Output:** feedforward steering angle δ_{ff_k}.

vehicle model developed for E-NMPC. Using the MLT solution, a terminal cost is set for E-NMPC in (6a), with the term multiplying the vector of weights $\boldsymbol{F}$. Such terminal cost imposes *soft* final conditions for the states $\boldsymbol{x}$, which improves the NMPC stability [15], [16]. We underline that the use of the terminal cost does *not* imply tracking the MLT solution: in Section VI we will show that the prediction horizon is long enough to let the E-NMPC replan different time-optimal trajectories, to comply with the current initial conditions of the vehicle. Exploiting the MLT to formulate a terminal cost is only a means to help stabilizing the NMPC solutions, and it is a widely used technique (in the form of soft equality or inequality constraints) in the literature of MPC and autonomous racing [2], [5], [9], [23].

3) E-NMPC SOLVER

The solver Pins [45], [46], [47] is adopted to solve the online E-NMPC and offline MLT problems. Pins uses the Pontryagin minimum principle to derive a two-point boundary value problem (TP-BVP) and implement the indirect OC approach. The inequality constraints are handled with penalty and barrier functions, which are included in the cost functional of the OC problem. The solution of the TP-BVP yields the optimal states and co-states, while the optimal controls are obtained with a minimization problem.

For a given OC problem, Pins derives symbolically the necessary equations, and it automatically generates a C++ code, which is then used to compute a numerical solution.

We used Pins for online minimum-time E-NMPC in [2] and [7].

B. LOW-LEVEL FEEDFORWARD STEERING CONTROLLER (NN)

In this section, we illustrate an original physics-driven neural network controller (NNc) of the steering wheel angle. Such a neural network is physics-driven, meaning that its structure is designed by exploiting the nonlinear laws of vehicle dynamics. We will show that the physically explainable formulation allows the use of fewer trainable parameters, and it improves the accuracy and generalization potential. We get inspiration from the inverse neural models in [19], but the network structure is here specialized to model the coupled longitudinal-lateral nonlinear dynamics.

Fig. 7 provides a conceptual input/output schematic of the neural controller: NNc calculates the steering wheel angle δ_{ff_k} for the current time step k, combining as input a window of steering angles that it computed in the past $\{\delta_{\mathrm{ff}_{k-r}}, \ldots, \delta_{\mathrm{ff}_{k-1}}\}$, and windows of future E-NMPC predictions for the lateral and longitudinal accelerations $\{a_y, a_x\}$ and the longitudinal speed v_x.

The neural controller NNc is a discrete-time system – whose sampling time is named T_{sNN} – and it learns an inverse dynamic model of the type:

$$\delta_{\mathrm{ff}_k} = \mathcal{F}_N(\boldsymbol{a_y}_k, \boldsymbol{v_x}_k, \boldsymbol{a_x}_k) + \begin{bmatrix}\delta_{\mathrm{ff}_{k-r}}, & \ldots, & \delta_{\mathrm{ff}_{k-1}}\end{bmatrix} \begin{bmatrix}\mathrm{A}_1 \\ \vdots \\ \mathrm{A}_r\end{bmatrix} \tag{9}$$

The neural functional $\mathcal{F}_N$ is devised to learn the nonlinear vehicle dynamics, as will be described in the following subsections. The vectors $\boldsymbol{a_y}_k$, $\boldsymbol{v_x}_k$ and $\boldsymbol{a_x}_k$ contain the present and future E-NMPC predictions, with the subscript $k \in \mathbb{N}$ being the current time step, and $\mathrm{x}_k = \mathrm{x}(kT_{\mathrm{sNN}})$ indicating the quantity x at time $t = kT_{\mathrm{sNN}}$:

$$\begin{cases} \boldsymbol{a_y}_k = \left[a_{y_k}, \ldots, a_{y_{k+q}}\right] & \text{(10a)} \\ \boldsymbol{v_x}_k = \left[v_{x_k}, \ldots, v_{x_{k+q}}\right] & \text{(10b)} \\ \boldsymbol{a_x}_k = \left[a_{x_k}, \ldots, a_{x_{k+q}}\right] & \text{(10c)} \end{cases}$$

In the training phase, the availability of the future behavior is exploited in $\mathcal{F}_N$ to learn part of the dynamic response of the system, and the delays between the steering angle and the lateral vehicle motion. The structure of $\mathcal{F}_N$ is inspired by the theory of neuro-fuzzy local models [48], [49], and we design the local models by leveraging the prior knowledge

of the vehicle dynamic laws. We used a conceptually similar approach in [19], [50], and [51]. The learnable weights $\{A_1, \ldots, A_r\}$ in (9) are used to implement an auto-regressive linear layer, which models the effect of the past steering values $\{\delta_{ff_{k-r}}, \ldots, \delta_{ff_{k-1}}\}$ on the currently computed δ_{ff_k}.

NNc is designed in two incremental variants. The first variant, named NNc_{base}, models only the pure lateral nonlinear dynamics. The second variant, called NNc_{ext}, extends NNc_{base} by learning the effect of the longitudinal acceleration a_x on the lateral dynamics. The two networks are trained sequentially, as described in Sections IV-A4 and IV-C1.

1) PURE LATERAL DYNAMICS

This section illustrates the neural controller NNc_{base} of the pure lateral nonlinear dynamics.

The overall structure of NNc_{base} is depicted in Fig. 8(a), and can be written as:

$$\delta_{ff_k} = f_{st}\left(\sum_{j=1}^{n_v}\left(\bar{\boldsymbol{G}}(\boldsymbol{a_{y_k}}, \boldsymbol{v_{xk}}) \odot \bar{\boldsymbol{\phi}}_j(\boldsymbol{v_{xk}})\right)\begin{bmatrix} F_{j_1} \\ \vdots \\ F_{j_{q+1}} \end{bmatrix}\right) + \begin{bmatrix}\delta_{ff_{k-r}}, & \ldots, & \delta_{ff_{k-1}}\end{bmatrix}\begin{bmatrix} A_1 \\ \vdots \\ A_r \end{bmatrix} \tag{11}$$

where $\odot$ denotes the element-wise product between two vectors. The output of NNc_{base} is the steering wheel angle δ_{ff_k}, while the inputs are the vectors $\{\boldsymbol{a_{y_k}}, \boldsymbol{v_{xk}}\}$, given in (10a,10b) and containing the future predictions for $\{a_y, v_x\}$, and a window of past steering angles $\{\delta_{ff_{k-r}}, \ldots, \delta_{ff_{k-1}}\}$ computed by NNc_{base}.

We now provide a brief overview of the functional blocks in (11) and Fig. 8(a), which will then be described in detail.

NNc_{base} learns most of the steady-state behavior by processing the inputs $\{\boldsymbol{a_{y_k}}, \boldsymbol{v_{xk}}\}$ with the vector function $\bar{\boldsymbol{G}}(\boldsymbol{a_{y_k}}, \boldsymbol{v_{xk}})$, which combines local approximations of the handling diagram. $\bar{\boldsymbol{G}}(\boldsymbol{a_{y_k}}, \boldsymbol{v_{xk}})$ is a vector of $q+1$ steering angle future predictions. The fully connected layers F_j in Fig. 8(a), $j \in \{1, \ldots, n_v\}$, learn linear combinations of the $\bar{\boldsymbol{G}}(\boldsymbol{a_{y_k}}, \boldsymbol{v_{xk}})$ vector entries, modeling the transient dynamics and improving the steady-state estimates provided by $\bar{\boldsymbol{G}}(\boldsymbol{a_{y_k}}, \boldsymbol{v_{xk}})$. To learn the change of the transient lateral dynamics in different ranges of v_x, we use a number n_v of fully connected layers $F_j, j \in \{1, \ldots, n_v\}$. The inputs of each fully connected layer F_j are selectively activated in predefined regions of v_x, by means of the activation functions $\bar{\boldsymbol{\phi}}_j(\boldsymbol{v_{xk}})$ in (11). Each of the F_j layers, $j \in \{1, \ldots, n_v\}$, has a single neuron and $q+1$ weights, which are named $\{F_{j_1}, \ldots, F_{j_{q+1}}\}$ in (11). The function f_{st} in (11) is a static steering map, identified in a preliminary phase, relating the steering wheel angle with the average angle at the front wheels. The weights $\{A_1, \ldots, A_r\}$ of the auto-regressive layer A finally contribute to model part of the transient and steady-state behavior, by learning a linear combination of the previously computed steering angles $\{\delta_{ff_{k-r}}, \ldots, \delta_{ff_{k-1}}\}$.

The details of the NNc_{base} formulation (11) are now illustrated. Let us first present how NNc_{base} learns most of the steady-state lateral behavior, while the transient dynamics is introduced later.

We start by recalling the handling diagram (HD) [2], [52], which is a description of the steady-state nonlinear under/oversteering characteristic of the vehicle:

$$\delta - \rho L = \delta - \frac{a_y}{v_x^2} L = K_{us}(a_y, v_x) \tag{12}$$

The HD measures the mismatch between the real steering angle δ (average angle at the front wheels) and the kinematic steering angle ρL, with $\rho = a_y / v_x^2$ and L being respectively the trajectory curvature and the vehicle wheelbase. $K_{us}(\cdot)$ is a (nonlinear) understeering gradient function, providing the shape of the HD. Fig. 9 shows the HD for the vehicle simulator of this paper, obtained with a sequence of low-frequency sine steering tests, at different values of speed (such maneuvers are described with more detail in Section IV-A3).

The $K_{us}(\cdot)$ function in (12) describes the shape of the HD, and it depends, in general, on a_y and v_x [2]. Neglecting for the moment the dependency on v_x, the function $K_{us}(\cdot)$ can be locally linearized around a value of $|a_y| = a_{y0_i}$, so that the steering angle δ is locally given by:

$$\delta = \bar{g}_i(a_y, v_x) = \frac{a_y}{v_x^2} L + k_{1_i} \text{sign}(a_y) + k_{2_i}\left(a_y - a_{y0_i} \text{sign}(a_y)\right) \tag{13}$$

To approximate the nonlinear HD, a number n_y of local models $\bar{g}_i(\cdot)$ can be created, with the structure of (13). The centers a_{y0_i} of the local models, $i \in \{1, 2, \ldots, n_y\}$, are computed to equally partition the range of $|a_y|$ values in the training set. The learnable parameters $\{k_{1_i}, k_{2_i}\}$ define the local linearization. The sign$(\cdot)$ functions in (13) are introduced to model a symmetric steering behavior,[5] for positive and negative a_y.

The output of each $\bar{g}_i(\cdot)$ local model is multiplied by an activation function $\bar{\phi}_i(|a_y|)$, which activates $\bar{g}_i(\cdot)$ in the vicinity of its linearization point a_{y0_i}.

As shown in Fig. 10(a), the functions $\bar{\phi}_i(|a_y|)$ are piecewise linear in $|a_y|$, they are equal to 1 only for $|a_y| = a_{y0_i}$, and they partition the unity, i.e., $\sum_{i=1}^{n_y} \bar{\phi}_i(|a_y|) = 1$. As described in Section IV-A4, we use $n_y = 6$ local models, to learn the lateral inverse dynamics in different regions of $|a_y|$.

The outputs of all the products $\bar{g}_i(a_y, v_x) \cdot \bar{\phi}_i(|a_y|)$, $i \in \{1, \ldots, n_y\}$, are finally summed, to obtain an approximation of the handling diagram over the entire range of a_y:

$$\delta = \bar{G}(a_y, v_x) = \sum_{i=1}^{n_y} \bar{g}_i(a_y, v_x)\bar{\phi}_i(|a_y|) \tag{14}$$

[5] If the vehicle behaved differently for $a_y > 0$ and $a_y < 0$, the sign$(\cdot)$ functions could be removed, and extra local models could be defined for $a_y < 0$.

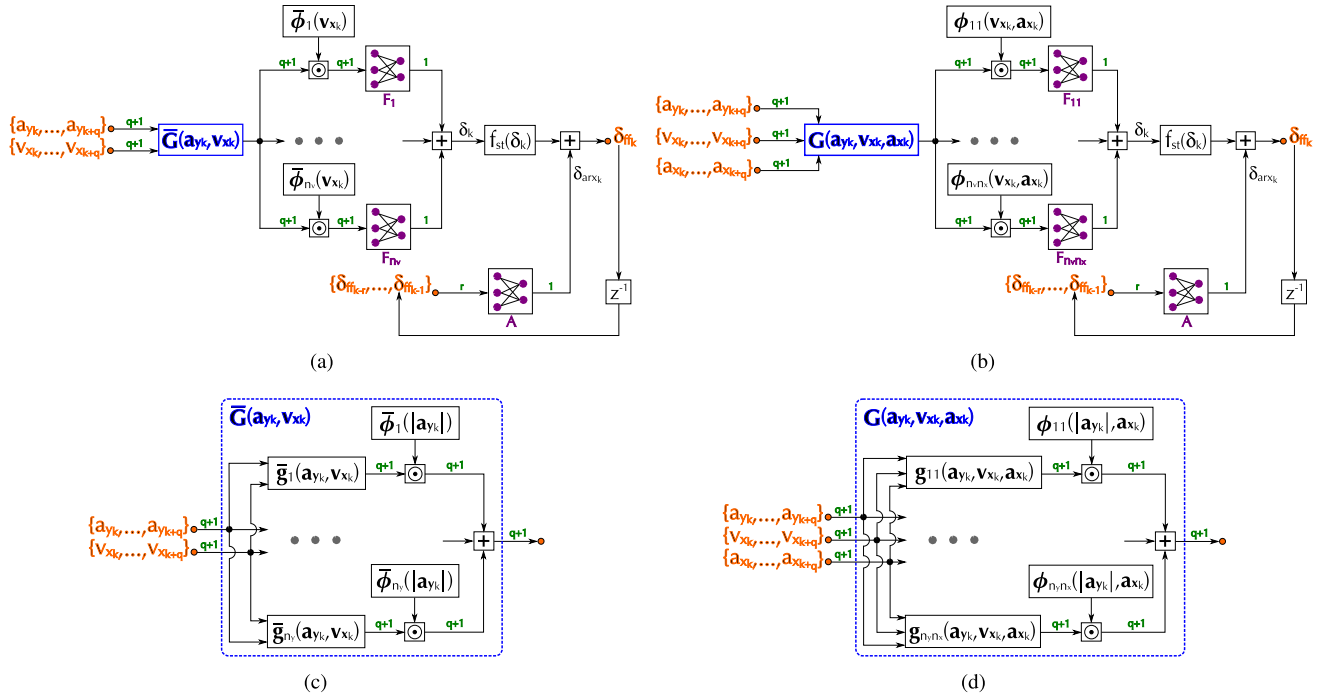


FIGURE 8. Neural controller of the steering wheel angle: (a) basic implementation to model the pure lateral inverse dynamics (NNc_{base}); (b) extended version to consider the coupling of the longitudinal-lateral dynamics (NNc_{ext}); (c) expanded view of the $\bar{G}(a_{y_k}, v_{x_k})$ threading layer of NNc_{base}; (d) expanded view of the $G(a_{y_k}, v_{x_k}, a_{x_k})$ threading layer of NNc_{ext}. The parameters and numbers in green indicate the size of the propagated signals.

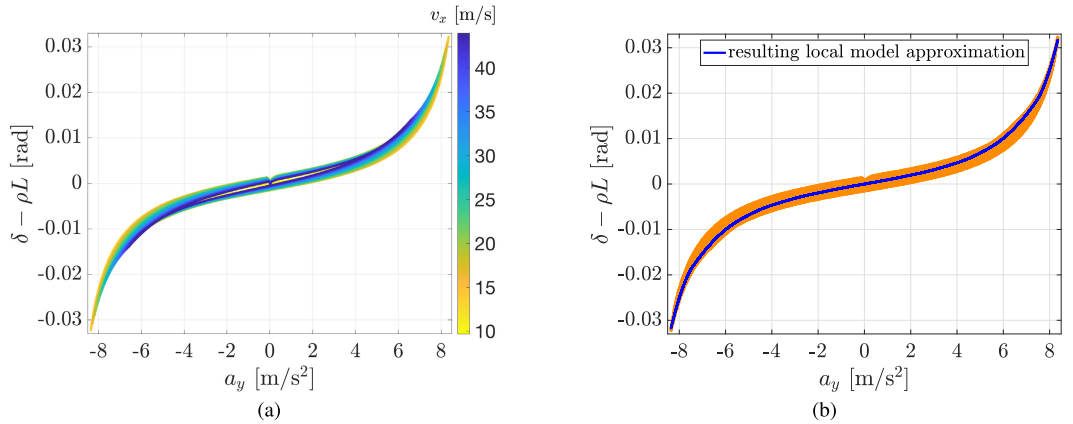


FIGURE 9. (a) Nonlinear handling diagram for the vehicle simulator, and (b) local model approximation of the diagram using (14).

The resulting HD approximation (14) is plotted in Fig. 9(b). Note that other local models can be created in different ranges of v_x, to approximate even better the shape of the HD. However, such additional local models were not found to improve significantly the overall model accuracy, and they are therefore not included in the formulation.

Equation (14) provides the steering angle (at the front wheels) in almost steady-state conditions, for given values of a_y and v_x.

We now discuss the integration of (14) inside the neural controller NNc_{base} (11). Since NNc_{base} receives as input the vectors $\{\boldsymbol{a}_{y_k}, \boldsymbol{v}_{\boldsymbol{x}k}\}$ of future predictions, the first step is to extend the local models $\bar{g}_i(a_y, v_x)$ in (13) and the function $\bar{G}(a_y, v_x)$ in (14) with their corresponding vector versions $\bar{\boldsymbol{g}}_i(\boldsymbol{a}_{y_k}, \boldsymbol{v}_{\boldsymbol{x}k})$ and $\bar{\boldsymbol{G}}(\boldsymbol{a}_{y_k}, \boldsymbol{v}_{\boldsymbol{x}k})$:

$$\begin{cases} \bar{\boldsymbol{g}}_i(\boldsymbol{a}_{y_k}, \boldsymbol{v}_{\boldsymbol{x}k}) = \boldsymbol{a}_{y_k} \odot (\boldsymbol{v}_{\boldsymbol{x}k})^{\odot -2} L + \mathrm{k}_{1_i} \mathrm{sign}(\boldsymbol{a}_{y_k}) \\ \qquad\qquad + \mathrm{k}_{2_i} \left(\boldsymbol{a}_{y_k} - a_{y0_i} \mathrm{sign}(\boldsymbol{a}_{y_k})\right) \end{cases} \tag{15a}$$

$$\bar{\boldsymbol{G}}(\boldsymbol{a}_{y_k}, \boldsymbol{v}_{\boldsymbol{x}k}) = \sum_{i=1}^{n_y} \left(\bar{\boldsymbol{g}}_i(\boldsymbol{a}_{y_k}, \boldsymbol{v}_{\boldsymbol{x}k}) \odot \bar{\boldsymbol{\phi}}_i(|\boldsymbol{a}_{y_k}|) \right) \tag{15b}$$

Note that, being $\boldsymbol{a}$ and $\boldsymbol{b}$ two generic vectors, we use the notation $\boldsymbol{a} \odot \boldsymbol{b}$ to indicate an element-wise product between $\boldsymbol{a}$ and $\boldsymbol{b}$, while $\boldsymbol{a}^{\odot x}$ raises each element of $\boldsymbol{a}$ to the power x.

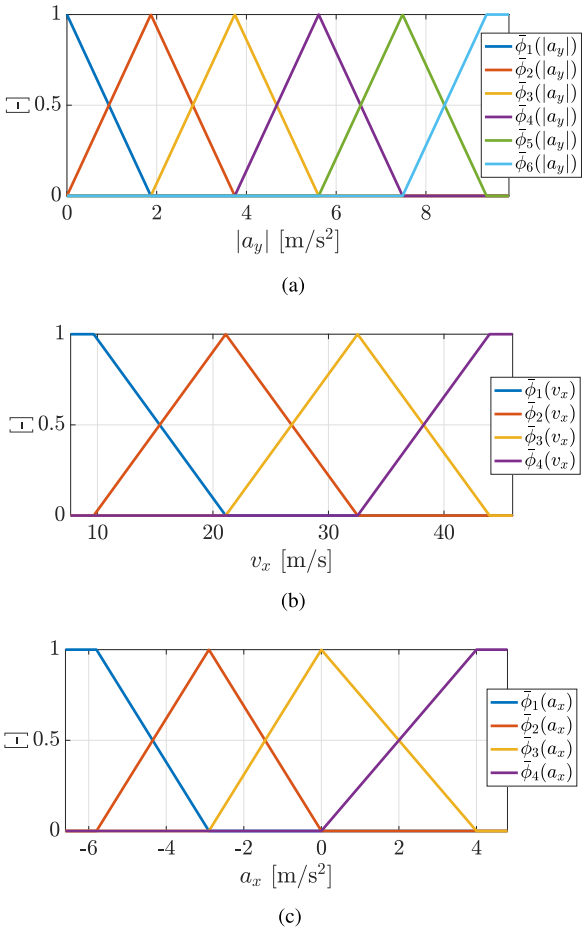


FIGURE 10. Shape of the activation functions: (a) $\bar{\phi}_i(|a_y|)$, $i \in \{1, \ldots, n_y\}$, (b) $\bar{\phi}_j(v_x)$, $j \in \{1, \ldots, n_v\}$, and c) $\bar{\phi}_l(a_x)$, $l \in \{1, \ldots, n_x\}$.

The term $\boldsymbol{a_{y_k}} \odot (\boldsymbol{v_{xk}})^{\odot -2}$ in (15a) is therefore the vector representation of $\frac{a_y}{v_x^2}$ in (12).

The activation functions $\bar{\boldsymbol{\phi}}_i(|\boldsymbol{a_{y_k}}|)$ in (15b) are the vector versions of the scalar $\bar{\phi}_i(|a_y|)$ in (14). The input of $\bar{\boldsymbol{\phi}}_i(\cdot)$ is the vector $|\boldsymbol{a_{y_k}}| = \left[|a_{y_k}|, \ldots, |a_{y_{k+q}}|\right]$ of E-NMPC predictions for the lateral acceleration:

$$\bar{\boldsymbol{\phi}}_i(|\boldsymbol{a_{y_k}}|) = \left[\bar{\phi}_i(|a_{y_k}|), \ldots, \bar{\phi}_i(|a_{y_{k+q}}|)\right] \tag{16}$$

with $i \in \{1, 2, \ldots, n_y\}$. The same scalar function $\bar{\phi}_i(\cdot)$ is used for each element of the $\bar{\boldsymbol{\phi}}_i(\cdot)$ vector in (16).

An element-wise product is performed in (15b) between the output vector of each local model $\bar{\boldsymbol{g}}_i(\cdot)$ and the corresponding activation vector function $\bar{\boldsymbol{\phi}}_i(|\boldsymbol{a_{y_k}}|)$, which activates the entries of $\bar{\boldsymbol{g}}_i(\cdot)$ in the neighborhood of the linearization point $a_{y_{0_i}}$. The resulting vector function $\bar{\boldsymbol{G}}(\cdot)$ in (15b) is the sum of all the locally-activated models. $\bar{\boldsymbol{G}}(\cdot)$ is used as a threading layer in the neural network NNc_{base} of Fig. 8(a), and an expanded view of $\bar{\boldsymbol{G}}(\cdot)$ is depicted in Fig. 8(c).

Since $\bar{\boldsymbol{G}}(\cdot)$ is simply the vector version of $\bar{G}(\cdot)$ in (14), the output of $\bar{\boldsymbol{G}}(\boldsymbol{a_{y_k}}, \boldsymbol{v_{xk}})$ is a vector of steering angle predictions in almost steady-state conditions, for the given vectors $\boldsymbol{a_{y_k}}$ and $\boldsymbol{v_{xk}}$.

The entries of the vectors $\boldsymbol{a_{y_k}}$ and $\boldsymbol{v_{xk}}$ in (10a,10b) range from the $k-$th present time step to the $(k+q)-$th future step. As a consequence, the vector $\bar{\boldsymbol{G}}(\boldsymbol{a_{y_k}}, \boldsymbol{v_{xk}})$ has in turn $q+1$ elements, pertaining to the present and future predictions. The fully connected layers F_j in Fig. 8(a), with $j \in \{1, \ldots, n_v\}$, are used to learn linear combinations of the $q+1$ predictions computed by $\bar{\boldsymbol{G}}(\cdot)$, thus modeling part of the transient lateral dynamics.[6] To consider the dependency of the lateral dynamics on the forward speed v_x, we use a number n_v of fully connected layers $\{\text{F}_1, \ldots, \text{F}_{n_v}\}$, which learn the transient dynamics in predefined ranges of v_x. The activation vector functions $\bar{\boldsymbol{\phi}}_j(\boldsymbol{v_{xk}})$ are used in (11) to create n_v local models of the transient dynamics, by selectively activating the inputs of each fully connected layer in predefined ranges of v_x. The vector functions $\bar{\boldsymbol{\phi}}_j(\boldsymbol{v_{xk}})$ are given in (17), and their scalar versions $\bar{\phi}_j(v_x)$, $j \in \{1, \ldots, n_v\}$, are plotted in Fig. 10(b). As described in Section IV-A4, in this work we use $n_v = 4$ local models to capture the lateral dynamics in different ranges of v_x.

$$\bar{\boldsymbol{\phi}}_j(\boldsymbol{v_{xk}}) = \left[\bar{\phi}_j(v_{x_k}), \ldots, \bar{\phi}_j(v_{x_{k+q}})\right] \tag{17}$$

As shown in Fig. 8(a), the scalar outputs of all the F_j layers are summed, $j \in \{1, \ldots, n_v\}$, yielding a first prediction of the steering angle δ_k at the front wheels, for the current time step k. Since the purpose of NNc_{base} is to compute a steering angle δ_{ff_k} to be applied at the steering wheel, the static map $f_{\text{st}}(\cdot)$ between δ_k and δ_{ff_k} – identified with preliminary static tests[7] – is applied to δ_k.

An auto-regressive linear layer, named A in Fig. 8(a), learns the effect of the past steering values $\{\delta_{\text{ff}_{k-r}}, \ldots, \delta_{\text{ff}_{k-1}}\}$ on the currently computed δ_{ff_k}, thus contributing to model the steering dynamics. The block z^{-1} is used in Fig. 8(a) to indicate that a one-step delayed version of the network output is recursively fed back to the input. The weights of the auto-regressive A layer are called $\{\text{A}_1, \ldots, \text{A}_r\}$ in (11). The output of the auto-regressive branch is named δ_{arx_k}, and it is summed to the estimate $f_{\text{st}}(\delta_k)$ produced by the combination of the local models. The overall output of the NNc_{base} neural model is therefore $\delta_{\text{ff}_k} = f_{\text{st}}(\delta_k) + \delta_{\text{arx}_k}$.

We provide more details about the training of the network NNc_{base} in Section IV-A4.

2) COMBINED LONGITUDINAL-LATERAL DYNAMICS

In this section, we extend the NNc_{base} neural controller to consider the effect of the longitudinal acceleration a_x on the inverse lateral dynamics. The extended model, named NNc_{ext}

[6] In a preliminary design phase, we even tried to incorporate some past history of the quantities a_y and v_x in the vectors $\boldsymbol{a_{y_k}}$ and $\boldsymbol{v_{xk}}$. However, the network accuracy improvement derived from the past history resulted to be negligible, which is why $\boldsymbol{a_{y_k}}$ and $\boldsymbol{v_{xk}}$ contain only the present and future predictions.

[7] Naming $\{\delta_{11}, \delta_{12}\}$ the steering angles at the front right and front left wheels, δ can be approximated with the average angle $(\delta_{11} + \delta_{12})/2$ at the front wheels. δ_{11} and δ_{12} are assumed to be measurable, but only in a preliminary identification phase.

and depicted in Fig. 8(b), is designed as:

$$\delta_{\mathrm{ff}_k} = f_{\mathrm{st}}\Bigg(\sum_{j=1}^{n_v}\sum_{l=1}^{n_x}\Big(\boldsymbol{G}(\boldsymbol{a_{y}}_k, \boldsymbol{v_x}_k, \boldsymbol{a_x}_k)\odot \boldsymbol{\phi}_{jl}(\boldsymbol{v_x}_k, \boldsymbol{a_x}_k)\Big) \cdot \begin{bmatrix} \mathrm{F}_{jl_1} \\ \vdots \\ \mathrm{F}_{jl_{q+1}} \end{bmatrix}\Bigg) + \begin{bmatrix} \delta_{\mathrm{ff}_{k-r}}, & \ldots, & \delta_{\mathrm{ff}_{k-1}} \end{bmatrix} \begin{bmatrix} \mathrm{A}_1 \\ \vdots \\ \mathrm{A}_r \end{bmatrix} \tag{18}$$

The differences between $\mathrm{NNc_{ext}}$ and $\mathrm{NNc_{base}}$ are threefold. First, $\mathrm{NNc_{ext}}$ uses the additional input vector $\boldsymbol{a_x}_k$, given in (10c) and containing the future predicted values of a_x. Second, the vector function $\bar{\boldsymbol{G}}(\boldsymbol{a_{y}}_k, \boldsymbol{v_x}_k)$ of $\mathrm{NNc_{base}}$ is extended with the new $\boldsymbol{G}(\boldsymbol{a_{y}}_k, \boldsymbol{v_x}_k, \boldsymbol{a_x}_k)$, which has a physics-driven formulation to model the contribution of a_x. Third, a higher number of local models and fully connected layers is used to learn the transient dynamics, in predefined ranges of v_x and a_x.

Let us start the illustration of (18) from the vector function $\boldsymbol{G}(\cdot)$, whose expanded structure is depicted in Fig. 8(d):

$$\boldsymbol{G}(\boldsymbol{a_{y}}_k, \boldsymbol{v_x}_k, \boldsymbol{a_x}_k) = \sum_{i=1}^{n_y}\sum_{l=1}^{n_x} \boldsymbol{g}_{il}(\boldsymbol{a_{y}}_k, \boldsymbol{v_x}_k, \boldsymbol{a_x}_k) \odot \boldsymbol{\phi}_{il}(|\boldsymbol{a_{y}}_k|, \boldsymbol{a_x}_k) \tag{19}$$

The function $\boldsymbol{G}(\cdot)$ is conceived as an extension of $\bar{\boldsymbol{G}}(\cdot)$ in (15b). The new $\boldsymbol{G}(\cdot)$ combines the local models $\boldsymbol{g}_{il}(\cdot)$, each of which is activated in the vicinity of a point $\{|a_y|, a_x\} = \{a_{y0_i}, a_{x0_l}\}$, by means of the activation vector functions $\boldsymbol{\phi}_{il}(|\boldsymbol{a_{y}}_k|, \boldsymbol{a_x}_k)$:

$$\boldsymbol{\phi}_{il}(|\boldsymbol{a_{y}}_k|, \boldsymbol{a_x}_k) = \bar{\boldsymbol{\phi}}_i(|\boldsymbol{a_{y}}_k|)\odot \bar{\boldsymbol{\phi}}_l(\boldsymbol{a_x}_k) \tag{20}$$

with $i \in \{1, \ldots, n_y\}$ and $l \in \{1, \ldots, n_x\}$. The functions $\bar{\boldsymbol{\phi}}_i(|\boldsymbol{a_{y}}_k|)$, whose number is $n_y = 6$, are the same used for $\mathrm{NNc_{base}}$ in (15b), while we adopt $n_x = 4$ functions $\bar{\boldsymbol{\phi}}_l(\boldsymbol{a_x}_k)$ to subdivide the operating range of a_x. The scalar versions of the activation functions, namely $\bar{\phi}_i(|a_y|)$ and $\bar{\phi}_l(a_x)$, are plotted in Fig. 10(a) and 10(c). The total number of local models in $\boldsymbol{G}(\cdot)$ is $n_y \cdot n_x$.

The models $\boldsymbol{g}_{il}(\boldsymbol{a_{y}}_k, \boldsymbol{v_x}_k, \boldsymbol{a_x}_k)$ in (19), $i \in \{1, \ldots, n_y\}$ and $l \in \{1, \ldots, n_x\}$, are an augmented version of the $\bar{\boldsymbol{g}}_i(\boldsymbol{a_{y}}_k, \boldsymbol{v_x}_k)$ in (15a), which were devised as local models of the handling diagram for the pure lateral dynamics. The derivation of the new models $\boldsymbol{g}_{il}(\cdot)$ is instead inspired by a local approximation of a nonlinear double-track vehicle model, around a generic quasi steady-state condition, in which $\{a_y, a_x\} = \{a_{y0_i}, a_{x0_l}\}$. We illustrate the technical details about the $\boldsymbol{g}_{il}(\cdot)$ local model derivation in Appendix A. The augmented model $\boldsymbol{g}_{il}(\cdot)$ contains additional terms, to learn the coupled longitudinal-lateral dynamics, and it is here written in scalar form, starting from (48) in Appendix A:

$$\begin{aligned} g_{il}(a_y, v_x, a_x) &= \frac{a_y}{v_x^2}L + \mathrm{k}_{y1_i}\mathrm{sign}(a_y) + \mathrm{k}_{y2_i}\Big(a_y - a_{y0_i}\mathrm{sign}(a_y)\Big) + \mathrm{k}_{y3_i}\mathrm{k}_{x1_l} \\ &\times \Big(a_y - (a_{y0_i} + \mathrm{k}_{y4_i})\mathrm{sign}(a_y)\Big)\Big(\mathrm{k}_{x2_l} + a_x - a_{x0_l}\Big) \\ &\times \Bigg[1 + \mathrm{k}_{y5_i}\Big(a_y - a_{y0_i}\mathrm{sign}(a_y)\Big) + \mathrm{k}_{x3_l} \\ &\times (a_x - a_{x0_l}) + \mathrm{k}_{x4_l}(a_x - a_{x0_l})^2 \\ &+ \mathrm{k}_{y6_i}\mathrm{k}_{x5_l}\Big(a_y - a_{y0_i}\mathrm{sign}(a_y)\Big)(a_x - a_{x0_l})\Bigg] \end{aligned} \tag{21}$$

The function $\boldsymbol{g}_{il}(\boldsymbol{a_{y}}_k, \boldsymbol{v_x}_k, \boldsymbol{a_x}_k)$ is the vector version of (21), here not written for brevity. Each of the $\boldsymbol{g}_{il}(\cdot)$ local models, $i \in \{1, \ldots, n_y\}$ and $l \in \{1, \ldots, n_x\}$, has $6 + 5 = 11$ learnable parameters, namely the set $\{k_{y1_i}, \ldots, k_{y6_i}, k_{x1_l}, \ldots, k_{x5_l}\}$. As stated in Appendix A, the subscripts y and x are used in the parameters $k_{y\_}$ and $k_{x\_}$ to symbolically separate the 6 terms related to the lateral dynamics $\{k_{y1_i}, \ldots, k_{y6_i}\}$ from the 5 terms of the longitudinal dynamics $\{k_{x1_l}, \ldots, k_{x5_l}\}$. Such a symbolic separation decreases the number of learnable parameters in the $\boldsymbol{G}(\cdot)$ function, which equals to $6n_y + 5\,n_x$ instead of $(6 + 5)n_y n_x$, without a significant decrease in the accuracy. Like in the case of the basic local model (13), we use the sign functions in (21) to model a symmetric lateral behavior, for positive and negative a_y.

By combining the local models $\boldsymbol{g}_{il}(\cdot)$, the function $\boldsymbol{G}(\cdot)$ in (19) computes a vector of present and future steering angle predictions. Following the idea already used for the $\mathrm{NNc_{base}}$ network, the fully connected layers $\{\mathrm{F}_{11}, \ldots, \mathrm{F}_{n_v n_x}\}$ in Fig. 8(b) are adopted to learn linear combinations of the $\boldsymbol{G}(\boldsymbol{a_{y}}_k, \boldsymbol{v_x}_k, \boldsymbol{a_x}_k)$ vector elements. The use of the fully connected layers allows the network to model the transient dynamics, and it improves the (quasi) steady-state estimates returned by $\boldsymbol{G}(\cdot)$. The network uses a number $n_v \cdot n_x$ of fully connected layers $\{\mathrm{F}_{11}, \ldots, \mathrm{F}_{n_v n_x}\}$, to learn the change of the lateral dynamics in n_v and n_x predefined regions of respectively v_x and a_x.

Like in the case of $\mathrm{NNc_{base}}$, each fully connected layer F_{jl} has one neuron and $q + 1$ weights, which are named $\{\mathrm{F}_{jl_1}, \ldots, \mathrm{F}_{jl_{q+1}}\}$ in (18), with $j \in \{1, \ldots, n_v\}$ and $l \in \{1, \ldots, n_x\}$. The inputs of the fully connected layers are locally activated by the vector functions $\boldsymbol{\phi}_{jl}(\boldsymbol{v_x}_k, \boldsymbol{a_x}_k)$ in (18), which partition the domain of v_x and a_x:

$$\boldsymbol{\phi}_{jl}(\boldsymbol{v_x}_k, \boldsymbol{a_x}_k) = \bar{\boldsymbol{\phi}}_j(\boldsymbol{v_x}_k)\odot \bar{\boldsymbol{\phi}}_l(\boldsymbol{a_x}_k) \tag{22}$$

The shapes of the scalar activation functions $\bar{\phi}_j(v_x)$ and $\bar{\phi}_l(a_x)$ are shown in Fig. 10.

Referring to Fig. 8(b), the rest of the $\mathrm{NNc_{ext}}$ network does not differ from $\mathrm{NNc_{base}}$: the outputs of the $\{\mathrm{F}_{11}, \ldots, \mathrm{F}_{n_v n_x}\}$ layers are summed, yielding the steering angle estimate δ_k. The static steering map $f_{\mathrm{st}}(\cdot)$ is then applied to δ_k, and the result $f_{\mathrm{st}}(\delta_k)$ is summed to the output δ_{arx_k} of the autoregressive layer A.

More details about the training and parameterization of the network $\mathrm{NNc_{ext}}$ are given in Section IV-C1.

We remark that the neural models $\mathrm{NNc_{base}}$ and $\mathrm{NNc_{ext}}$ enable the incremental learning of the pure and combined nonlinear lateral dynamics. If no data were available in the nonlinear handling regions, the networks could still be used to learn the linear steering behavior. Additionally, the physics-

driven structure prevents the network from yielding fancy outputs, in situations for which it was not trained.

C. LOW-LEVEL FEEDBACK STEERING CONTROLLER (PI)

The neural steering controller previously presented in Section III-B operates in feedforward, i.e., it uses the E-NMPC target trajectories to compute the control action δ_{ff_k}. However, model mismatches and external disturbances (e.g. local friction small changes) may still result in undesirable trajectory tracking errors. A PI feedback steering controller is developed to compensate the yaw rate tracking error $\Omega_{e_k} = \Omega_k - \hat{\Omega}_k$, with $\{\Omega_k, \hat{\Omega}_k\}$ being the target and the real yaw rate values for the time step k. As shown in Fig. 3, the steering feedback correction δ_{fb_k} is added to the δ_{ff_k} computed by the feedforward controller. The overall steering wheel angle is $\delta_{D_k} = \delta_{\text{ff}_k} + \delta_{\text{fb}_k}$. Section IV-B2 discusses the PI tuning method, by means of iterative learning.

D. LOW-LEVEL LONGITUDINAL CONTROLLER (PID-AW)

1) CONTROLLER DESIGN

This section deals with the design of the low-level longitudinal speed-tracking controller, whose purpose is to compute the throttle and brake pedal signals to track the high-level E-NMPC speed trajectories.

The longitudinal controller is implemented with a PID logic, which is designed with a model-based approach. We develop a longitudinal dynamic model, and we identify its parameters with a set of maneuvers. The longitudinal model is then linearized using a list of steady-state speed values, around which local PID controllers are designed, with target specifications for the closed-loop system. The PID gains are then linearly interpolated as a function of the vehicle speed (gain scheduling). The pedal control saturation is handled with the back-calculation anti-windup (AW) method.

2) LONGITUDINAL DYNAMIC MODEL

In this section, we outline the longitudinal dynamic model that we use to design the speed-tracking controller. Most of the parameters of the presented dynamic model can be identified, as described in Section IV-A1. However, to ease the identification process, we assume the prior knowledge of the parameters in the set $\mathcal{A} = \{m, h_G, I_w, L_1, L_2\}$ (Table 1). As already discussed in Section II-A, the term *partially-unknown* vehicle is used only because we assume to know the parameter set $\mathcal{A}$. The quantities in $\mathcal{A}$ are used only to perform a model-based design of the speed-tracking PID controller. However, in Section VI-C2 we will show that our framework is robust to variations of the parameters in $\mathcal{A}$.

We extend the longitudinal dynamic model of [2], including the wheel rotational dynamics:

$$\begin{cases} m\dot{v}_x = 2(F_{x_1} + F_{x_2}) - c_0 - c_v v_x - c_a v_x^2 & \text{(23a)} \\ \tau_{\tilde{a}_x}\dot{\tilde{a}}_x + \tilde{a}_x = \dot{v}_x & \text{(23b)} \\ I_{w_i}\dot{\omega}_i = -F_{x_i} r_{w_i} + T_{w_i}(p, \omega_i), \quad i \in \{1, 2\} & \text{(23c)} \end{cases}$$

The states of the model are $\{v_x, \tilde{a}_x, \omega_1, \omega_2\}$, while the control is the pedal $p \in [-1, 1]$ ($p > 0$ for throttle, $p < 0$ for brake). In (23a), m is the vehicle mass, $\{F_{x_1}, F_{x_2}\}$ are the longitudinal forces for a single front and rear tire, while $\{c_0, c_v, c_a\}$ are friction and drag parameters. Equation (23b) provides an estimate of the longitudinal acceleration $\tilde{a}_x$, which is a delayed version of $\dot{v}_x$ (with a small time constant $\tau_{\tilde{a}_x}$). As shown next, the estimate $\tilde{a}_x$ is useful to compute the vertical loads F_{z_i} and the longitudinal load transfer in (25a). Equation (23c) describes the wheel rotational dynamics, with $\{\omega_1, \omega_2\}$ being the angular rates of the front and rear wheels. In this section, the subscript $i \in \{1, 2\}$ is used to refer to the front and rear wheels, respectively. $\{I_{w_i}, r_{w_i}, T_{w_i}(\cdot)\}$, $i \in \{1, 2\}$, are the wheel rotational inertia, the rolling radius, and the wheel torque functions.

A simplified Pacejka model is adopted for the tire forces:

$$F_{x_i} = D_{x_i} \sin\Big(C_x \text{atan}\big(B_{x_i}\kappa_i\big)\Big), \quad i \in \{1, 2\} \qquad (24)$$

where:

$$\begin{cases} F_{z_i} = \frac{1}{2}\big(mg\frac{L_{\bar{i}}}{L} + C_{D_i}v_x^2 + (-1)^i m\tilde{a}_x\frac{h_G}{L}\big) & \text{(25a)} \\ \text{d}f_{z_i} = \frac{F_{z_i}}{F_{z_{0i}}} - 1 & \text{(25b)} \\ D_{x_i} = F_{z_i}\big(p_{D_{x1}} + p_{D_{x2}}\text{d}f_{z_i}\big), \quad C_x = p_{C_{x1}} & \text{(25c)} \\ K_{xk_i} = F_{z_i}\big(p_{K_{x1}} + p_{K_{x2}}\text{d}f_{z_i}\big) & \text{(25d)} \\ B_{x_i} = \frac{K_{xk_i}}{C_x D_{x_i}} = \frac{p_{K_{x1}} + p_{K_{x2}}\text{d}f_{z_i}}{p_{C_{x1}}\big(p_{D_{x1}} + p_{D_{x2}}\text{d}f_{z_i}\big)} & \text{(25e)} \\ \kappa_i = \frac{\omega_i r_{w_i} - v_x}{v_{\text{den}_i}(v_x, \omega_i)} & \text{(25f)} \end{cases}$$

with $i \in \{1, 2\}$ indicating the front and rear axles; $\bar{i} = 1$ if $i = 2$, and $\bar{i} = 2$ if $i = 1$. F_{z_i} are the vertical tire loads, while $\{g, h_G, C_{D_i}, F_{z_{0i}}\}$ are respectively the gravitational acceleration, the center of mass (CoM) height from ground, the aerodynamic downforce coefficients and the static vertical loads. The parameters in the set tire_par $= \{p_{D_{x1}}, p_{D_{x2}}, p_{C_{x1}}, p_{K_{x1}}, p_{K_{x2}}\}$ are assumed to be equal for the front and rear tires. $v_{\text{den}_i}(v_x, \omega_i)$ in the expression of κ_i contains a smooth low-speed correction.

The vehicle simulator to be controlled (Section II) is FWD. The driving front wheel torque $T_{w_{d1}}$ is computed as a function of the pedal p and ω_1, using (26b). The dependency of the powertrain torque curve on $\{p, \omega_1\}$ is modeled with the polynomial relations in (26a,26b) (we use polynomials of order P_q=P_m=4). The wheel braking torques $\{T_{wb1}, T_{wb2}\}$ for the front and rear wheels are calculated with (26c), as a function of p and $\{\omega_1, \omega_2\}$. The maximum braking torques are named $\{f_{b_1}, f_{b_2}\}$, while $\{\text{pPart}(\cdot), \text{nPart}(\cdot)\}$ are smooth differentiable functions (defined in Appendix B), extracting the positive and negative part of their arguments. The total wheel torques

$\{T_{w_1}, T_{w_2}\}$ are given by (26d,26e).

$$\begin{cases} f_{m_i}(p) = \sum_{q=0}^{P_q} f_{m_{iq}} p^q, \quad i \in \{0, 1, \ldots, P_m\} & \text{(26a)} \\ T_{w_{d1}}(p, \omega_1) = \text{pPart}(p) \cdot \left(\sum_{i=0}^{P_m} f_{m_i}(p) \cdot \omega_1^i \right) & \text{(26b)} \\ T_{w_{bi}}(p, \omega_i) = -\text{nPart}(p) \cdot f_{b_i}, \quad i \in \{1, 2\} & \text{(26c)} \\ T_{w_1}(p, \omega_1) = T_{w_{d1}}(p, \omega_1) + T_{w_{b1}}(p, \omega_1) & \text{(26d)} \\ T_{w_2}(p, \omega_2) = T_{w_{b2}}(p, \omega_2) & \text{(26e)} \end{cases}$$

The identification of the model parameters (see Table 2 for the complete list) is described in Section IV-A1.

IV. AUTOMATIC IDENTIFICATION/LEARNING OF THE MODELS FOR CONTROL

In this section, we outline the identification/learning of the high- and low-level models presented in Section III.

A three-round automatic identification scheme is developed, as depicted in Fig. 11. We define our approach as an *asynchronous closed-loop identification*, in which the accuracy of the control models and the estimate of the maximum performance improve from the first to the third identification rounds. We remark that the proposed scheme is fully automatic, i.e., it does not require manual tuning.

The reader can find a video demonstration of the automatic three-round identification scheme at the following link: `https://www.youtube.com/watch?v=xQ_T96IjGP8`.

Note that the generic term *identification* is used for the entire framework, even if our approach contains a mix of parameter identification, neural network training and iterative learning.

In the first round, *open-loop* maneuvers are devised to identify the longitudinal and the pure lateral nonlinear dynamics, and to build an initial estimate of the maximum performance. In the second round, the artificial agent uses the first-round controllers to drive the vehicle in *closed-loop* along a given circuit. The results are employed to improve the first-round estimate of the achievable performance, and to tune the steering feedback controller. Finally, in the third round, the artificial agent drives the vehicle with the second-round controllers, collecting new data to learn the effect of the longitudinal acceleration on the lateral dynamics.

Our three-round automatic identification scheme is inspired by the way used by human professional drivers to incrementally learn how to reach the vehicle dynamic limits. Starting from a preliminary knowledge of the vehicle dynamic behavior (corresponding to our round n°1), professional drivers typically perform sessions of laps to progressively improve their ability to drive the vehicle faster and faster (rounds n°2 and n°3).

A. FIRST IDENTIFICATION ROUND

The first identification round consists of the following *open-loop* maneuvers:

(f1) Identification of the top speed and the steering maps.
(f2) Identification of the longitudinal model and tuning the longitudinal controller (III-D and IV-A1).

TABLE 2. Identification of the longitudinal dynamic model.

Type	Identified quantities	Specifications	
		pedal p	v_{x_0}—v_{x_F}
Zero pedal	$\{r_{w_1}, r_{w_2}, c_0, c_v, c_a\}$	0	$v_{\max}$—0
Max throttle	$\{C_{D_1}, \text{tire\_par}\}$	1	0—$v_{\max}$
Max throttle+brake	$\{C_{D_1}, C_{D_2}, \text{tire\_par}\}$	1→-1	0—0
Throttle sweep	function $T_{w_{d1}}(p, \omega_1)$	sweep >0	0—$v_{\max}$
Constant brake	$\{f_{b_1}, f_{b_2}\}$	const <0	$v_{\max}$—0

TABLE 3. Evaluation of the high- and low-level models with test sets.

Longitudinal dynamic model to tune the PID-AW controller	
RMS($\tilde{v}_x$)	1.86 km/h
Yaw rate prediction model for E-NMPC	
RMS($\tilde{\Omega}$)	$5.1 \cdot 10^{-3}$ rad/s
Lateral speed prediction model for E-NMPC	
RMS($\tilde{v}_y$)	$3.50 \cdot 10^{-2}$ m/s
Feedforward neural steering controller (NNc_{ext})	
RMS($\tilde{\delta}_{\text{ff}}$)	1.36 deg

(f3) Identification of the maximum lateral performance (IV-A2).
(f4) Estimation and fitting of the G-G-v diagram (7c), using the pure longitudinal, lateral and combined open-loop tests which we described in [2].
(f5) Fit the functions $\{\tau_\Omega(v_x), H_{v_y}(a_y, v_x), \tau_{v_y}(v_x)\}$ of the yaw rate and lateral speed prediction models for E-NMPC (III-A1), using sine steering tests (IV-A3.a).
(f6) Train the feedforward neural steering controller NNc_{base} (III-B1 and IV-A4).

1) IDENTIFICATION OF THE LONGITUDINAL DYNAMICS

In this section, we briefly describe the identification process for the longitudinal dynamic model of Section III-D. We remark that such a model is employed only to design the low-level speed-tracking controller.

The parameters of the longitudinal model are identified by minimizing the prediction error of (23a), with the maneuvers in Table 2. Table 2 provides the values of pedal p, initial and final speed $\{v_{x_0}, v_{x_F}\}$ for each test.

The model validation is shown in Fig. 12. Table 3 reports the root mean squared (RMS) value for $\tilde{v}_x = \hat{v}_x - v_x$, with $\{v_x, \hat{v}_x\}$ being the predicted and the real speed signals. Note that the longitudinal dynamic model is employed only to ease the tuning of the longitudinal controller, hence a model mismatch may have a moderate impact on the final speed tracking performance.

2) IDENTIFICATION OF THE MAXIMUM LATERAL PERFORMANCE

Before proceeding with the identification of the pure and combined lateral dynamics, it is useful to know the maximum steering angles that can be applied to the vehicle, to maximize the performance without exceeding the stability limits.

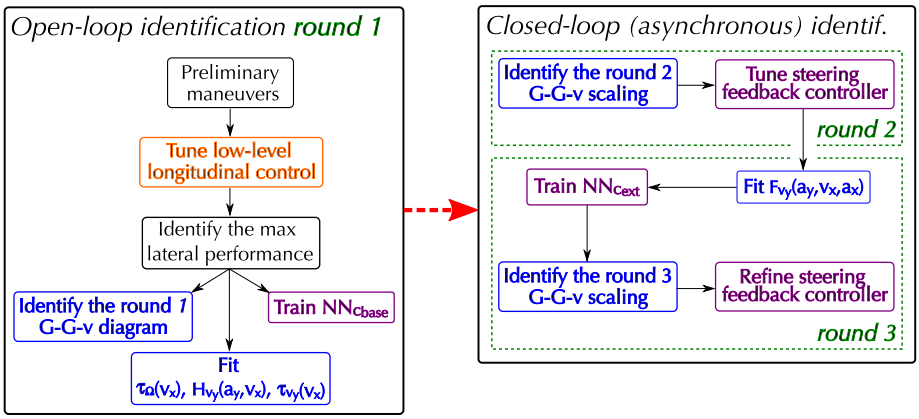


FIGURE 11. Three-step identification scheme. The same colors are used as in Fig. 3, to classify the identification steps: blue for the high-level E-NMPC, purple for the low-level steering controllers, and orange for the low-level longitudinal controller.

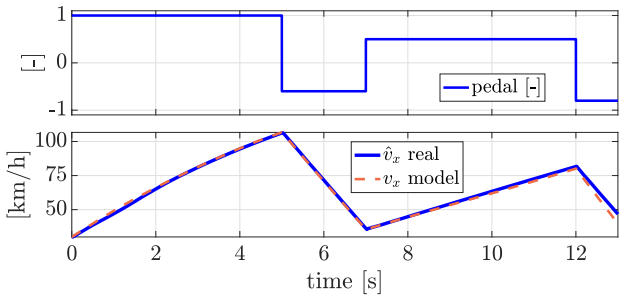


FIGURE 12. Validation of the longitudinal dynamic model.

Following [2], the maximum lateral performance is identified with constant-speed ramp steer tests, by recording the steering angle values above which no significant increase in the lateral acceleration a_y is obtained.

The ramp steer maneuvers are repeated with a list of N_R (constant) speed values, ranging up to the maximum speed $v_{\max}$. The v_x-dependent steering angle limits are defined as the ones producing lateral accelerations $a_y(v_x) = (1 - \Psi) \cdot a_{y_t}(v_x)$, with $a_{y_t}(v_x)$ being the maximum achievable a_y value, for a given v_x. A safety margin $\Psi = 0.06$ is kept in order to enable possible local violations of the learned performance constraints (G-G-v diagram (7c)-(7g)), which may occur to compensate a model mismatch during closed-loop control.

A dataset, hereafter named $\mathcal{T}$, is created to store the N_R speed values and the corresponding steering angle limits.

3) IDENTIFICATION OF THE DYNAMIC MODEL FOR E-NMPC

This section deals with the identification of the dynamic model for E-NMPC, presented in Section III-A1. We focus the description on the yaw rate and lateral speed prediction models (1c)-(1d).

a: YAW RATE PREDICTION MODEL

The identification of the yaw rate prediction model (1c) for E-NMPC is performed following the method that we proposed in [2].

Constant-speed sine steer tests are carried out with the vehicle simulator, using a low steering frequency, and choosing the steer amplitudes and speed values from the identified dataset $\mathcal{T}$ (Section IV-A2). Such sine steer tests represent almost steady-state conditions, and the handling diagram (HD) resulting from the maneuvers is shown in Fig. 9. The HD is modeled with the nonlinear function $K_{\text{us}}(a_y, v_x)$ in (12), which we here define as a (bilinear) spline that interpolates the HD points.

In the yaw rate dynamic model (1c) for E-NMPC, we design the function $\tau_\Omega(v_x)$ as a second order polynomial[8] in v_x. To optimize the parameters of the function $\tau_\Omega(v_x)$, we need to execute the same sine steer tests with the vehicle simulator and then with (1c). In the identification phase, the latter NMPC model (1c) is reformulated as:

$$\tau_\Omega(v_x) \cdot \dot{\Omega} + \Omega = \Omega_{\text{ss}} \tag{27}$$

In (27), Ω_{ss} represents a steady-state yaw rate request. For the sine steer maneuvers, Ω_{ss} can be approximated using the HD equation (12) and the $K_{\text{us}}(\cdot)$ spline, substituting the steady-state relation $\rho = \Omega_{\text{ss}}/v_x$:

$$\delta - \frac{\Omega_{\text{ss}}}{v_x} L = K_{\text{us}}(a_y, v_x) \longrightarrow \Omega_{\text{ss}} = v_x \frac{\left(\delta - K_{\text{us}}(a_y, v_x)\right)}{L} \tag{28}$$

The function $\tau_\Omega(v_x)$ is fitted by minimizing the prediction error of (27)-(28), using the same sine steering tests described before. We remark that (28) is used only in the identification phase, to fit $\tau_\Omega(v_x)$, but (28) is not part of the E-NMPC model.

The double lane change maneuvers of Fig. 13 are employed to validate the E-NMPC yaw rate model. Table 3 reports the RMS value for $\tilde{\Omega} = \hat{\Omega} - \Omega$, where $\{\Omega, \hat{\Omega}\}$ are the predicted and the real yaw rate signals.

[8]Polynomials with higher order do not improve significantly the model accuracy.

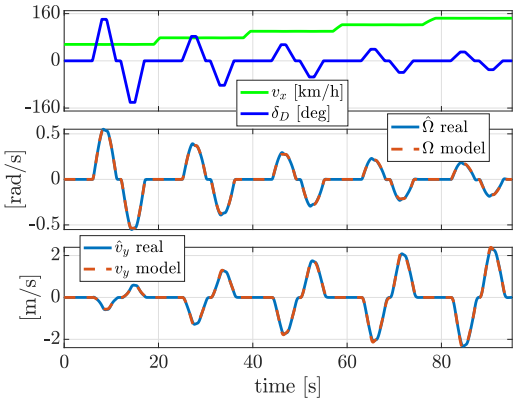


FIGURE 13. Validation of the yaw rate (Ω) and lateral speed (v_y) prediction models for E-NMPC, with double lane change maneuvers.

b: LATERAL SPEED PREDICTION MODEL

The identification of the lateral speed prediction model (1d) for E-NMPC is performed in two steps:

1) Fitting the nonlinear map $H_{v_y}(\cdot)$ in (3a), with the same set of constant-speed sine steering tests adopted for the yaw rate prediction model. Fig. 4 depicts the resulting $a_y - v_x - v_y$ lateral velocity diagram.
2) Fitting the polynomial function $\tau_{v_y}(v_x)$ by minimizing the prediction error of (1d),[9] using the same maneuvers of point 1. The model is validated using the double lane change maneuvers of Fig. 13.

4) TRAINING THE FEEDFORWARD STEERING CONTROLLER NNc_{base}

This section discusses the parameterization and the training method for the neural steering controller NNc_{base}, presented in Section III-B1.

The neural network NNc_{base} is a discrete-time dynamical system, whose sampling time is $T_{\text{sNN}} = 0.05$ s. The input vectors $\boldsymbol{a_{y_k}}$ and $\boldsymbol{v_{xk}}$ contain $q+1 = 15$ samples, covering of $(q+1)T_{\text{sNN}} = 0.75$ s of future E-NMPC predictions. The auto-regressive branch in the network linearly combines $r = 15$ past steering angle values computed by NNc_{base}, thus keeping track of $rT_{\text{sNN}} = 0.75$ s of past history. The number of local models to learn the inverse lateral dynamics in predefined ranges of $|a_y|$ and v_x is respectively $n_y = 6$ and $n_v = 4$. The best values for the parameters $\{n_y, n_v\}$ are selected with a grid-search approach, and the best model is the one that maximizes the Akaike's information criterion (AIC, Section V) [48], to trade-off accuracy against number of parameters.

The centers $a_{y_{0_i}}$ and $v_{x_{0_j}}$ of the piecewise linear activation functions $\bar{\phi}_i(|a_y|)$ and $\bar{\phi}_j(v_x)$, with $i \in \{1, \ldots, n_y\}$ and $j \in \{1, \ldots, n_v\}$, are chosen to evenly subdivide the ranges of $|a_y|$ and v_x values measured in the training dataset (Fig. 10).

Training Method and Dataset: The neural model NNc_{base} is trained using supervised learning, where the objective is to minimize the mean square error (MSE) between the predicted steering wheel angles δ_{ff} and the actual values $\hat{\delta}_{\text{ff}}$, obtained during training maneuvers. To generate the training, validation and test datasets, open-loop sinusoidal steering maneuvers are performed at constant speed values. The operating range is divided into a list of equally spaced vehicle speed values, and the sinusoidal steering tests are repeated for each speed value. The amplitudes of the steering sinusoids are determined based on the previously identified steering limits (Section IV-A2), which correspond to the maximum achievable lateral accelerations and are dependent on the vehicle speed. The datasets used for training, validation and testing contain respectively 10, 3 and 2 sinusoidal steering maneuvers, carried out at different values of speed, steering frequencies and steering amplitudes. The attached video provides a brief overview of this process.

The number of learnable parameters for the NNc_{base} model is $2n_y + (q+1)n_y + r = 87$.

The neural networks are trained in Tensorflow, using the Keras API and the Nadam optimization method [53].

B. SECOND IDENTIFICATION ROUND

As outlined in Section IV-A, the first identification round consists of only open-loop maneuvers, aimed at obtaining a preliminary estimate of the maximum performance, and at identifying/learning the pure longitudinal and lateral dynamics.

However, the first-round estimate of the G-G-v diagram (point (f4) in Section IV-A) for the E-NMPC problem may not ensure the closed-loop controllability of the vehicle, since the peak acceleration values recorded with open-loop tests may be feasible only for a short time, and wheel lock may occur (see Section II-B). In the second round, the artificial agent controls the vehicle in *closed-loop*, along a given circuit, using the first-round control models. The closed-loop results are used for the following operations:

(s1) Maximize the G-G-v constraint for E-NMPC: using as initial condition the first-round G-G-v diagram with a down-scaled longitudinal performance, iteratively enlarge the achievable acceleration envelope, as long as the following closed-loop requirements (c1)-(c3) are satisfied.
  (c1) Wheel lock/spin conditions are avoided: the tire longitudinal slips must not exceed given thresholds, above which the vehicle performance would degrade (the vehicle has no ABS and no traction control, see II-B).
  (c2) The difference between the target E-NMPC lateral acceleration a_y and the actual value $\hat{a}_y$ is always below a certain threshold, to ensure a limited model mismatch.[10]

[9] To identify $\tau_{v_y}(v_x)$, the function $F_{v_y}(\cdot)$ is replaced with $H_{v_y}(\cdot)$ in (1d).

[10] In the second identification round, the vehicle is controlled with the NNc_{base} neural network, which does not model the effect of a_x on the steering dynamics. Hence, $|a_y - \hat{a}_y|$ may reach high values if $|a_x|$ is large, leading to mismatches between the real and the predicted trajectories. Such an issue is attenuated in the third identification round, in which the NNc_{ext} controller is used.

(c3) The difference between the lateral coordinate n_{MLT} of the MLT solution (Section III-A2) and the actual value n is always limited.[11]

(s2) Tune the feedback steer controller (III-C) using iterative learning (IV-B2).

1) MAXIMIZING THE G-G-V CONSTRAINT

As outlined in the previous paragraph, the second identification round begins by creating a new estimate of the G-G-v constraint for E-NMPC, aimed at ensuring certain closed-loop requirements while maximizing the vehicle performance.

We devise an automatic iterative procedure, that uses as initial guess a down-scaled version of the first-round G-G-v diagram, and iteratively enlarges it while ensuring the requirements (c1)-(c3). The initial guess is built by reducing the longitudinal accelerations of the first-round G-G-v diagram: the v_x-dependent functions $\{a_{x_{\text{M}}}(v_x), a_{x_{\text{m}}}(v_x)\}$, modeling the longitudinal performance in the G-G-v diagram (7c) and fitted in the first round, are here multiplied by the factors $\{\eta_1, \eta_2\} \in (0, 1]$, whose initial value is 0.4.

Algorithm 2 is a simplified pseudo-code for the G-G-v scaling scheme: the vehicle is controlled in closed-loop, and the scaling factors $\{\eta_1, \eta_2\}$ of the longitudinal performance are iteratively increased if the (c1)-(c3) criteria are met for an entire lap (lines 6-12). Moreover, the algorithm can selectively limit the combined steering-braking accelerations (without degrading the pure braking performance), by activating the G-G[12] constraints (7e)-(7g) and optimizing their exponent n_b[13] (lines 15-16). The reduction of the combined braking performance increases with n_b. However, to limit the non-convexity of the resulting G-G-v constraint, an upper bound for n_b is used during the identification. The resulting second-round G-G-v diagram is shown in Fig. 6.

2) TUNING THE FEEDBACK STEERING CONTROLLER

This section explains the tuning method for the PI feedback steering controller, described in Section III-C. Since the vehicle to be controlled is partially-unknown, the PI controller gains are optimized with a model-free technique. The iterative learning tuning (ILT) method of [38] is preferred to other model-free approaches (e.g. [54], [55]), since it requires no assumptions about the plant model, and it can handle generic cost functions. Each learning iteration consists of the closed-loop control of the vehicle simulator for one lap, along a circuit, to collect new data and update the PI parameters.[14] Using ILT, the PI gains $\boldsymbol{k} = [k_P, k_I]$ for the $i+1$-th iteration

Algorithm 2 Maximizing the G-G-v Constraint

1: **Input:** initial guess for η_1, η_2, n_b
2: **Output:** $\eta_{1_{\text{opt}}}, \eta_{2_{\text{opt}}}, n_{b_{\text{opt}}}$
3: $\eta_{1_{\text{sel}}}, \eta_{2_{\text{sel}}}, n_{b_{\text{sel}}}, \eta_{1_{\text{opt}}}, \eta_{2_{\text{opt}}}, n_{b_{\text{opt}}} \leftarrow \varnothing$
4: **while** true **do**
5: **if** ($\eta_{1_{\text{opt}}} \neq \varnothing$ **and** $\eta_{2_{\text{opt}}} \neq \varnothing$) **then break end if**
6: **if** (c1) and (c2) and (c3) satisfied **then**
7: **if** $\eta_{1_{\text{opt}}} = \varnothing$ **then**
8: $\eta_1 \leftarrow$ increase_acceleration(η_1)
9: **end if**
10: **if** $\eta_{2_{\text{opt}}} = \varnothing$ **then**
11: $[\eta_2, n_b] \leftarrow$ increase_braking(η_2, n_b)
12: **end if**
13: $\eta_{1_{\text{sel}}} \leftarrow \eta_1, \eta_{2_{\text{sel}}} \leftarrow \eta_2, n_{b_{\text{sel}}} \leftarrow n_b$
14: **else**
15: **if** (c1) or (c2) or (c3) violated while braking **then**
16: $[\eta_2, n_b] \leftarrow$ reduce_braking(η_2, n_b)
17: **else**
18: $\eta_1 \leftarrow$ reduce_acceleration(η_1)
19: **end if**
20: **end if**
21: **if** $\eta_1 = \eta_{1_{\text{sel}}}$ **then** $\eta_{1_{\text{opt}}} \leftarrow \eta_1$ **end if**
22: **if** $\eta_2 = \eta_{2_{\text{sel}}}$ **and** $n_b = n_{b_{\text{sel}}}$ **then**
23: $\eta_{2_{\text{opt}}} \leftarrow \eta_2, n_{b_{\text{opt}}} \leftarrow n_b$
24: **end if**
25: **end while**

are computed as:

$$\boldsymbol{k}_{i+1} = \boldsymbol{k}_i - \boldsymbol{\gamma}_i \cdot J_i(\boldsymbol{k}_i) \tag{29}$$

In (29), the cost function $J_i(\boldsymbol{k}_i)$ is a weighted sum of the lap time and the RMS values of the yaw rate tracking error Ω_e, the control action δ_{fb}, and its variations $\Delta\delta_{\text{fb}}$. In order to ensure the decrease of the cost function, the relation $|1 - \frac{\mathrm{d}J(\boldsymbol{k}_i)}{\mathrm{d}\boldsymbol{k}}\boldsymbol{\gamma}_i| < 1$ must hold [38]. The gradient $\frac{\mathrm{d}J(\boldsymbol{k}_i)}{\mathrm{d}\boldsymbol{k}}$ is estimated with the secant method in [56], using the results of the current and previous iterations. The magnitude of the learning gains $\boldsymbol{\gamma}_i$ is computed as $|\boldsymbol{\lambda}\frac{\mathrm{d}J(\boldsymbol{k}_i)}{\mathrm{d}\boldsymbol{k}}^{-1}|$, with $\boldsymbol{\lambda} \in (0, 1)$ being fixed scaling factors. The signs for the two components of $\boldsymbol{\gamma}_i$ are selected with additional sub-iterations, to evaluate the best descent direction (M2 search method in [38]). If no sign combination for $\boldsymbol{\gamma}_i$ reduces the cost, the step size $|\boldsymbol{\gamma}_i|$ is decreased with a linesearch approach. The ILT terminates when certain exit conditions are met.

C. THIRD IDENTIFICATION ROUND

In the third identification round, the second-round models are employed for the closed-loop vehicle control, along the same circuit used in the second round. The collected data are used to model the effect of a_x on the lateral dynamics, in the following way:

(t1) Identify the dependency of the v_y dynamics on a_x, by fitting the parameters $\{b_1, b_2, \ldots, b_6\}$ of $F_{v_y}(\cdot)$ in (4), minimizing the prediction error of (1d). The RMS

[11] If $|n_{\text{MLT}} - n|$ is large (mainly due to model mismatches), the lap time increases and a very sub-optimal solution may be found. Note, however, that our E-NMPC problem is *not* formulated to track the MLT solution, but rather to plan minimum-time trajectories on a long horizon (300 m).

[12] The term G-G is used instead of G-G-v to emphasize that the constraints (7e)-(7g) do not depend on the vehicle speed v_x.

[13] In the constraints (7e)-(7g), only n_b is optimized. The other parameters $\{a_{x_c}, a_{y_c}, a_{x_b}, a_{y_b}\}$ are instead computed as functions of $\{\eta_2 a_{x_{\text{m}}}, a_{y_{\text{M}}}\}$.

[14] During the ILT, the full control scheme of Fig. 3 is used, and only the gains of the steering feedback controller are updated.

TABLE 4. Identification results for the G-G-v constraint maximization, during the second and third identification rounds.

Round n°	Identified parameters			
	η_1	η_2	(7d)-(7e) enabled	n_b
2	0.78	0.46	yes	1.1
3	0.80	0.54	yes	0.9

of $\tilde{v}_y = \hat{v}_y - v_y$ for the validation set is listed in Table 3, where $\{v_y, \hat{v}_y\}$ are the predicted and the real lateral velocity signals.

(t2) Train the feedforward steering controller NNc_{ext} (see III-B2 and the dedicated subsection IV-C1 for details).

(t3) Refine the G-G-v diagram and the feedback steering controller, with the methods in IV-B1 and IV-B2.

Additional identification rounds would not result in a performance improvement, meaning that the parameters of the NNc_{ext} network and of the feedback steering controller have already been optimized to model the inverse lateral dynamics, and that a further expansion of the G-G-v diagram would lead to a violation of the (c1)-(c3) conditions.

Table 4 reports the results of the G-G-v identification scheme. A quantitative analysis of the vehicle performance in the last two identification rounds is offered in Section VI (Table 6).

1) TRAINING THE FEEDFORWARD STEERING CONTROLLER NNc_{ext}

This section describes the parameterization and training of the neural steering controller NNc_{ext}, outlined in Section III-B2.

NNc_{ext} extends the structure of NNc_{base}, and it shares with NNc_{base} the same sampling time $T_{\text{SNN}} = 0.05$ s, the number of samples in the input vectors $q + 1 = 15$ and in the autoregressive branch $r = 15$, the number of local models $n_y = 6$ to model the inverse lateral dynamics in predefined ranges of $|a_y|$.

Inside NNc_{ext}, the function $\boldsymbol{G}(\cdot)$ in (19) has additional $n_x = 4$ local models to learn the dynamics in selected regions of a_x, and the total number of local models in $\boldsymbol{G}(\cdot)$ is $n_y n_x = 24$. As stated in Section III-B2, a symbolic distinction is performed for the parameters of the local models $g_{il}(\cdot)$ in (21), $i \in \{1, \ldots, n_y\}$ and $l \in \{1, \ldots, n_x\}$, so that the total number of learnable parameters in $\boldsymbol{G}(\cdot)$ is $6n_y + 5\,n_x$.

As shown in Fig. 10(c), the centers $\{a_{x_{0_1}}, a_{x_{0_2}}\}$ of the activation functions $\{\bar{\phi}_1(a_x), \bar{\phi}_2(a_x)\}$ equally partition the range of measured negative a_x values, $a_{x_{0_3}}$ is placed in $a_x = 0$, while $a_{x_{0_4}}$ equals the maximum recorded a_x.

The same activation functions $\bar{\phi}_l(a_x)$, $l \in \{1, \ldots, n_x\}$, are used, together with the functions $\bar{\phi}_j(v_x)$, $j \in \{1, \ldots, n_v\}$, to create a number $n_x n_v$ of local fully connected layers F_{jl}, through which the transient lateral dynamics is learned.

Training Method and Dataset: The neural model NNc_{ext} is trained using the same supervised learning method and algorithm as NNc_{base} (Section IV-A4). During the third identification round, the second-round controllers (including NNc_{base}) are employed to drive the vehicle in closed-loop, on the same circuit as in the second round. The data collected in the third round are used to train NNc_{ext}, as visually described in the attached video. Specifically, telemetry data from 6 laps are stored: 4 for training, 1 for validation, and 1 for testing. To introduce some variability into the dataset, more aggressive or conservative trajectories are planned by the E-NMPC motion planner, via down-scaling the G-G-v constraint or increasing τ_{a_x} in (1b).

Since NNc_{ext} extends the structure of the previously trained NNc_{base}, the optimized parameters of NNc_{base} are used as initial guesses[15] for the training of NNc_{ext}.

The NNc_{ext} network has $6n_y + 5\,n_x + (q+1)n_v n_x + r = 311$ learnable parameters.

The RMS of the prediction error $\tilde{\delta}_{\text{ff}} = \hat{\delta}_{\text{ff}} - \delta_{\text{ff}}$ on the test set is reported in Table 3.

V. PHYSICS-DRIVEN NEURAL CONTROLLERS VERSUS GENERIC RECURRENT NETWORKS

The novel physics-driven networks NNc_{base} and NNc_{ext} are benchmarked against a generic recurrent NN, named gRNN. Similarly to NNc_{ext}, gRNN computes the steering wheel angle δ_{ff_k} using the windows of future predictions for $\{a_y, v_x, a_x\}$. The structure of gRNN is the cascade of a basic recurrent layer (with 30 internal states and a tanh activation function) and a fully connected layer (with one neuron and a linear activation function). The total number of learnable parameters for gRNN is 1051.

The two networks are compared using the RMS of the prediction error $\tilde{\delta}_{\text{ff}} = \hat{\delta}_{\text{ff}} - \delta_{\text{ff}}$ and the Akaike's information criterion (AIC) [48], which is defined as:

$$\text{AIC} = N_t \cdot \log\Big(\text{MSE}_t(\tilde{\delta}_{\text{ff}})\Big) + 2\,n_p \tag{30}$$

where N_t is the number of samples in the test set, $\text{MSE}_t(\tilde{\delta}_{\text{ff}})$ is the mean squared error in testing, and n_p is the number of learnable parameters of the model. When using the AIC to compare different models, the one with the highest |AIC| (absolute value) represents the best trade-off between accuracy and complexity (number of parameters).

A. COMPARING NNc_{base} WITH A GENERIC RECURRENT NETWORK

The physics-driven network NNc_{base} is here compared with the benchmark generic recurrent NN (gRNN) previously described. The same training set and hyperparameters are used to compare the two neural networks. Since NNc_{base} is designed to model the pure lateral inverse dynamics, the training and test datasets consist of a sequence of constant-speed sine steering maneuvers, each of which is carried out with a different speed and sine steering amplitude (Section IV-A4). The performance metrics on the test set (never used

[15] Specifically, we use the trained parameters of the $\bar{\boldsymbol{G}}(\cdot)$ function and fully connected F_j layers, $j \in \{1, \ldots, n_y\}$, of NNc_{base} as initial guesses for NNc_{ext}.

TABLE 5. Benchmarking the novel physics-driven models NNc_{base} and NNc_{ext} against a generic recurrent network (gRNN), on different test datasets. The proposed NNc_{base} and NNc_{ext} have fewer parameters than the benchmark gRNN, and they have a better generalization capacity when tested on new data. They yield a lower RMS value of the steering angle prediction error, and a higher |AIC| score, indicating a better trade-off between model complexity and accuracy.

Model name	N° parameters	RMS($\tilde{\delta}_{ff}$)	|AIC|
Pure lateral dynamics (sine-steer maneuvers)			
gRNN (benchmark)	1051	1.794 deg	71889.3
NNc_{base}	87	0.298 deg	109612.1
Combined longitudinal-lateral dynamics (Valencia circuit)			
gRNN (benchmark)	1051	2.517 deg	85393.1
NNc_{ext}	311	1.360 deg	104181.3

during training) are listed in Table 5. The results on a portion of the test set are depicted in Fig. 14 (left column plots).

Despite the higher number of parameters (1051 vs 87), the benchmark gRNN has a lower accuracy with respect to NNc_{base}, which results in a lower |AIC|.

B. COMPARING NNc_{ext} WITH A GENERIC RECURRENT NETWORK

In a second phase, gRNN is used as a benchmark for NNc_{ext}, which models the combined inverse lateral dynamics. The two networks are trained with the same hyperparameters, on a dataset collected by autonomously driving the vehicle simulator along the Valencia circuit (Spain), using the NNc_{base} steering controller (Section IV-C1). The test set is generated in the same way, but, in comparison with the training set, the high-level E-NMPC is forced to plan more conservative or aggressive maneuvers (by changing the scaling factors of the G-G-v constraint).

Referring to Table 5, the physics-driven NNc_{ext} network yields a lower RMS error with respect to the benchmark gRNN. The |AIC| value is higher for NNc_{ext}, due to the lower RMS error and the lower amount of learnable parameters (311 vs 1051). Fig. 14 (right column) shows the results on a portion of the test set.

The general-purpose internal structure of the benchmark gRNN leads to reduced modeling and generalization capability, necessitating larger training sets for reliable predictions on new test data. In contrast, the internal structures of NNc_{base} and NNc_{ext} draw inspiration from vehicle dynamics theory, enabling the use of fewer physically interpretable parameters and enhancing learning potential.

VI. SIMULATION RESULTS

In this section, we show the results obtained by the artificial agent, to control the vehicle simulator of Section II on two different circuits.

A video demonstration of the automatic identification scheme and the motion planning and control results is available at: `https://www.youtube.com/watch?v=xQ_T96IjGP8`.

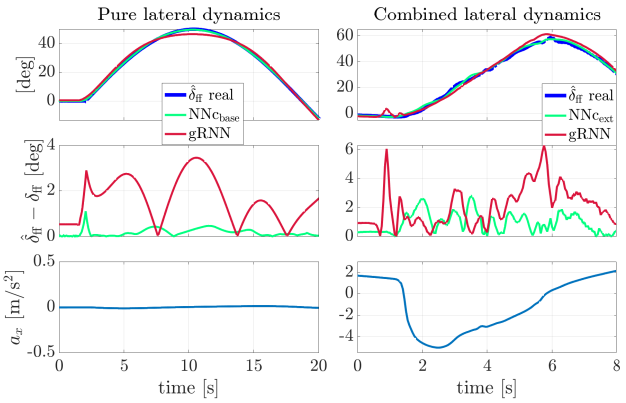


FIGURE 14. Benchmarking the novel neural models NNc_{base} and NNc_{ext} against a generic recurrent network (gRNN), to calculate the steering wheel angle δ_{ff}. The plots show portions of the test datasets for the pure lateral dynamics ($a_x \approx 0$, left column) and for the combined lateral dynamics ($a_x \neq 0$, right column). The NNc_{base} and NNc_{ext} exhibit a better generalization capacity than the benchmark gRNN, when tested on new data.

The second and third identification rounds (Sections IV-B and IV-C) are carried out on the Valencia circuit (Spain). Fig. 15, 16 and 17 show the online planning and control results, at the end of the third round. The offline MLT problem solution, computed with the same parameters and dynamic model used for E-NMPC (Section III-A2), is used as a benchmark for the comparison.

Fig. 15(b) and 16(a) show a close matching between the planned (orange) and executed (blue) profiles of forward and lateral speed $\{v_x, v_y\}$, yaw rate Ω, longitudinal and lateral accelerations $\{a_x, a_y\}$, proving the effectiveness of the low-level longitudinal and steering controllers.

The small differences between the planned and real lateral speed v_y indicate that the lateral speed prediction model for E-NMPC (Section IV-A3.b) is accurate enough, even in the presence of longitudinal accelerations. An accurate high-level prediction and low-level tracking of Ω, v_y, and v_x are beneficial for a close tracking of the desired vehicle trajectory, planned online with E-NMPC.

Leveraging the very long prediction horizon (Section III-A2), the E-NMPC state trajectories and vehicle path (Fig. 15) are close to the MLT solution (which is calculated on the entire lap). However, we underline that the E-NMPC problems are *not* formulated to track the MLT reference: as will be discussed in Sections VI-A and VI-C2, the online minimum-time E-NMPC solutions can locally differ from the MLT, since the E-NMPC can replan the time-optimal trajectory given the current initial states. The zoomed views n.2-3 of Fig. 15(a) focus on turn 8: in comparison with the MLT solution, the artificial agent plans and executes a wider trajectory in the entry phase (zoom-2), to then perform a sort of *late-apex* maneuver [57] in the corner exit phase (zoom-3). More details about the importance of replanning will be given in Section VI-A.

Fig. 17 shows the steering wheel angle δ_D, which is the sum of the feedforward angle δ_{ff} computed by the neural

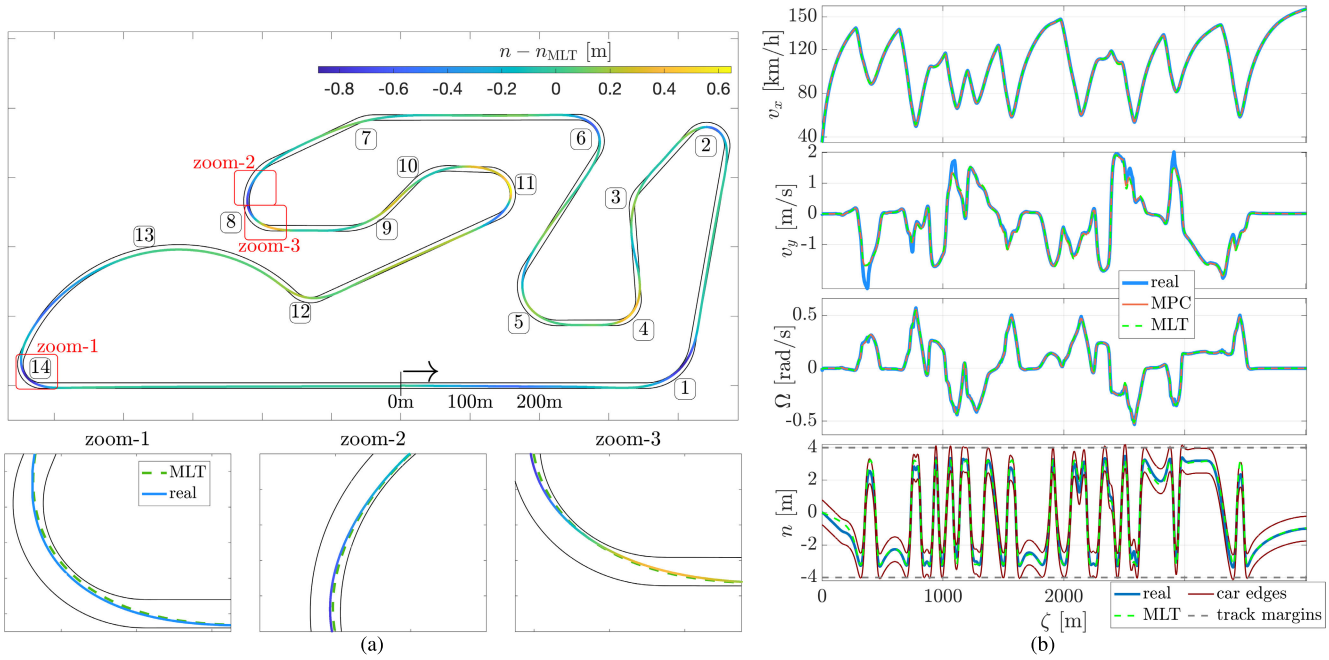


FIGURE 15. Results on the Valencia circuit: (a) comparison of the trajectories executed by the artificial agent ("real", colored solid line) and computed by the offline OC problem ("MLT", green dashed line). A color map is used to highlight the difference $n - n_{\text{MLT}}$, with n and n_{MLT} being the lateral coordinates of the executed trajectory and the MLT solution. (b) Comparison of the planned (orange), executed (blue) and benchmark ("MLT", green dashed) linear velocities $\{v_x, v_y\}$, yaw rate Ω and lateral coordinates n.

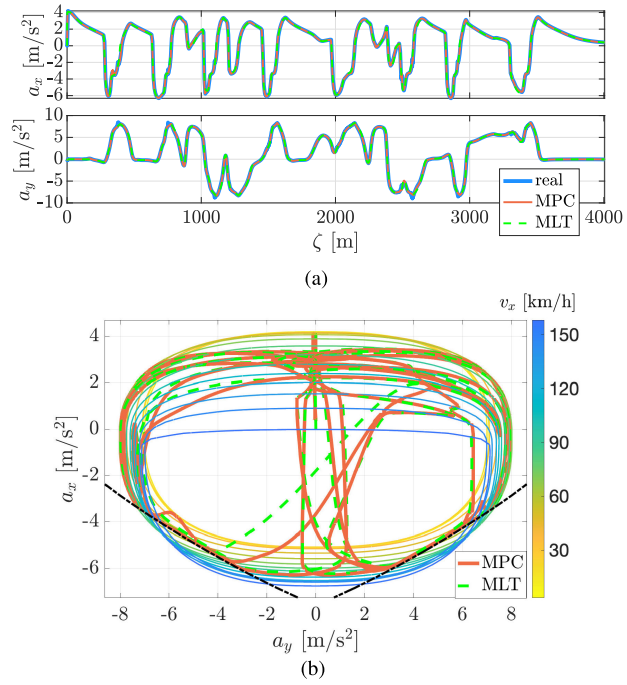


FIGURE 16. (a) Longitudinal and lateral accelerations, and (b) G-G-v diagram, Valencia circuit.

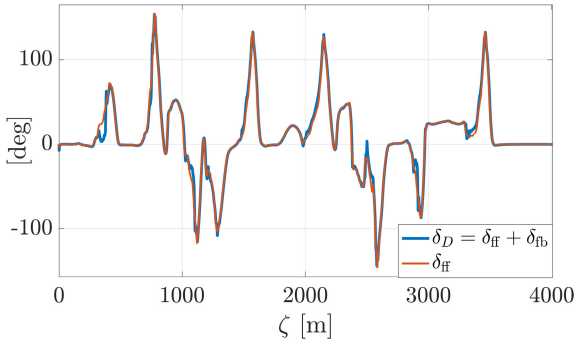


FIGURE 17. Steering wheel angle δ_D computed by the artificial agent on the Valencia circuit. A large fraction of δ_D is provided by NNc_{ext}, which returns the feedforward angle δ_{ff}, while the feedback steering controller injects some quicker corrections δ_{fb}.

steering controller NNc_{ext}, and the feedback quantity δ_{fb}. A large fraction of δ_D is provided by the feedforward angle δ_{ff}, meaning that NNc_{ext} learns a sufficiently accurate model of the combined inverse lateral dynamics. The feedback steering controller injects some quicker corrections, and compensates part of the local inaccuracies of NNc_{ext}. Moreover, the feedback steering correction also helps compensating for unmodeled disturbances or changes in the vehicle dynamics, e.g. due to the variation of some vehicle parameters (Section VI-C2).

As depicted in Fig. 16(b), the vehicle is driven up to the limits of the identified G-G-v constraint, exploiting the maximum identified pure and combined longitudinal-lateral accelerations.

Table 6 compares the lap times obtained by the artificial agent and by the offline MLT problem. At the end of the third identification round, the artificial agent achieves a lap time $T_{\text{MPC}} = 141.363$ s, driving the vehicle simulator in

TABLE 6. Results obtained by the artificial agent, at the end of the identification rounds n.2 and n.3. After training and testing the framework on the Valencia circuit, the agent robustness is evaluated on a new racetrack.

Round n°	lap time online MPC	lap time offline MLT
Training and testing on the same circuit (Valencia)		
2	142.767 s	141.139 s
3	141.363 s	
Testing on a new circuit (Adria)		
–	109.648 s	109.473 s

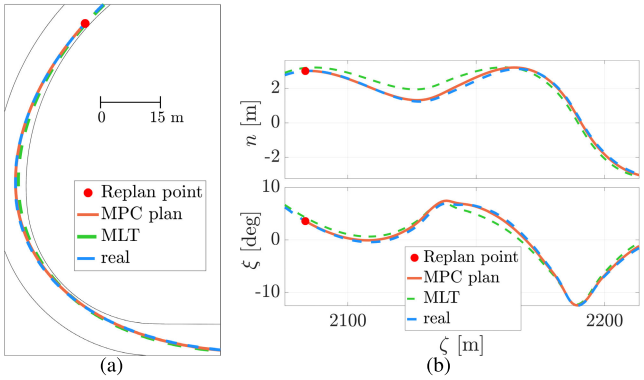


FIGURE 18. Trajectory replanned by the agent at the turn n.8 of the Valencia circuit. Due to a slightly different initial relative yaw angle ξ with respect to the MLT solution, the agent replans a new time-optimal maneuver (orange solid line), which is then accurately executed (blue dashed line). (a) vehicle path, (b) lateral coordinate n and relative yaw angle ξ.

closed-loop. The MLT problem sets a slightly lower lap time $T_{\text{MLT}} = 141.139$ s. The gap $T_{\text{MPC}} - T_{\text{MLT}} = 224$ ms is mainly due to tracking errors of the low-level steering and longitudinal controllers, which force the E-NMPC to replan new minimum-time trajectories, locally different from the MLT (Fig. 15(a)).

We may also argue that, since the MLT problem does not use the complex multibody dynamic model of the vehicle simulator to be controlled, the MLT lap time may be overestimated. It is out of the scope of this paper to formulate and solve an MLT problem with the exact vehicle model to be controlled.

Since the vehicle does not have an ABS nor a TC (Section II-B), the planned accelerations should produce minimum-time maneuvers without inducing partial wheel lock or spin conditions, which would decrease the vehicle stability and increase the model mismatches. To ensure the prescribed controllability requirements ((c1)-(c3) in IV-B) and maximize the achievable performance, during the identification rounds n°2 and n°3 our automatic procedure (IV-B1) computes the best scaling for the first-round G-G-v diagram, returning the factors $\{\eta_1, \eta_2, n_b\}$ (Table 4).

In the second round, the E-NMPC planner is more cautious, to satisfy the conditions (c1)-(c3) despite the incomplete knowledge of the vehicle dynamics: the identified parameters $\{\eta_1, \eta_2\}$ are lower and the lap time is higher, with respect to the third round.

In the third identification round, the dependency upon the longitudinal acceleration a_x is introduced in the low-level steering controller NNc_{ext} and in the high-level lateral speed model. Such augmented models yield an improved prediction accuracy, which enables the use of larger factors $\{\eta_1, \eta_2\}$ and results in a lower lap time (Table 4 and 6).

The entire motion planning and control framework is developed in C++ code, and a MacBook Pro machine with a 2.6 GHz 6-Core Intel i7 processor is used. Solving the E-NMPC problems with Pins takes an average CPU time of 20 ms. While a new E-NMPC solution is being computed, the low-level controllers use the solution of the previous E-NMPC step. The execution of the low-level longitudinal and steering controllers requires on average less than 18 μs. We underline that no GPU is used to run the neural networks, whose computational complexity is reduced through the physics-driven formulation. The overall planning and control structure can therefore run online, with the execution rates given in Section III.

A. IMPORTANCE OF REPLANNING

The ability to quickly replan and execute time-optimal trajectories is a key characteristic of professional drivers [57], which is not achievable with a pure tracking of a fixed minimum-time trajectory, as highlighted in [23]. The zoomed views in Fig. 15(a) show that, in certain corners, the artificial agent plans and then executes different maneuvers with respect to the MLT, to comply with the real initial states of the vehicle in the corner entry phases. Fig. 18 depicts the trajectory replanned at the turn n.8 of the Valencia circuit. As the vehicle enters the corner (red dot in Fig. 18), the deviations of the lateral coordinate n and the relative yaw angle ξ (Fig. 5) from the MLT solution are $\Delta n = n_{\text{MLT}} - n = 3.2 - 3.0 = 0.2$ m and $\Delta\xi = \xi_{\text{MLT}} - \xi = 4.3 - 3.6 = 0.7$ deg. The shape of the time-optimal trajectory is particularly sensitive to the initial relative yaw angle ξ: on account of $\Delta\xi$, the E-NMPC plans a new slightly wider maneuver (orange solid line) in the corner entry phase, which then becomes a sort of late-apex maneuver towards the exit phase. As depicted in Fig. 18(a), the agent starts replanning quite soon, before entering the corner, and then accurately executes the newly planned maneuver throughout the turn (blue dashed line). Given the real vehicle initial conditions, the MLT path (green dashed line) would lead to a violation of the G-G-v constraint, which is the reason why the E-NMPC changes the minimum-time trajectory.

The deviations of the executed path from the MLT solution should therefore *not* be seen as MLT path tracking errors, but rather as different trajectories replanned online by the agent, to consider the real vehicle state at each replanning step. In a similar way, the authors of [57] show that expert race drivers use different trajectories, depending on the current situation and on their driving styles, yet achieving similar final lap times.

The replanning capability of our E-NMPC is improved by the long prediction horizon (300 m) and the high number of mesh points (346), in comparison with many related literature papers, like [6], [8], [9], and [27]. The use of such a long horizon is enabled by the novel kineto-dynamical vehicle model for E-NMPC, and by the effectiveness of the OC solver Pins (Section III-A3).

B. IMPACT OF THE LATERAL SPEED MODEL

In this section, we analyze the benefit brought by the novel lateral speed v_y prediction model for E-NMPC on the trajectory tracking performance.

We focus our analysis on the corners n.9 and n.10 of the Valencia circuit, and the main results are plotted in Fig. 19.

As shown in Fig. 19(a), when the v_y prediction model is *not* used in the E-NMPC formulation (*i.e.*, $F_{v_y} = 0$ in (1d)), the trajectory executed by the artificial agent (red line) has visible deviations from the MLT benchmark, since the maneuvers planned with E-NMPC become more difficult to be tracked by the low-level steering controllers. Such a tracking difficulty can be explained with Fig. 19(b): when the v_y model is not used, the lateral velocity planned by the high-level E-NMPC is zero on the entire horizon (not plotted), but the maneuver executed by the agent produces a non-zero v_y evolution (red line). The lateral velocity and the chassis side slip angle $\beta = \text{atan}(v_y/v_x)$ have an impact on the transient trajectory curvature, given by $\rho = (\dot{\beta} + \Omega)/\sqrt{v_x^2 + v_y^2}$. If the E-NMPC model assumes that $\dot{\beta} = v_y = 0$, but the real vehicle executes the maneuver with $v_y \neq 0$, then the planned trajectory curvature will be different from the real one. The executed trajectory is, in certain points of Fig. 19(a), remarkably different from the offline MLT, and the E-NMPC decreases the forward speed v_x to cope with the large mismatch.

If instead the E-NMPC predicts v_y with the proposed model, the executed vehicle and state trajectories (blue lines in Fig. 19) are closer to the offline MLT, and the mismatch between planning and execution decreases. When the v_y model is used, the maneuver time is lower by 0.65 s, which is a remarkable difference.

In the autonomous racing literature papers [5], [6], [23], point-mass models were used for online MPC, but the yaw rate and the lateral speed dynamics were not modeled. In this paper, we instead develop sufficiently accurate and numerically efficient models of the yaw rate and lateral speed dynamics for E-NMPC, while preserving a kineto-dynamical formulation that does not require the prior knowledge of the tire or suspension parameters.

C. ROBUSTNESS ANALYSIS

1) TESTING ON NEW CIRCUITS

The robustness of the identification and control framework is first evaluated by driving the vehicle along a new circuit, never used in the identification process. The Adria racetrack (Italy) is chosen, and the main results are shown in Fig. 20.

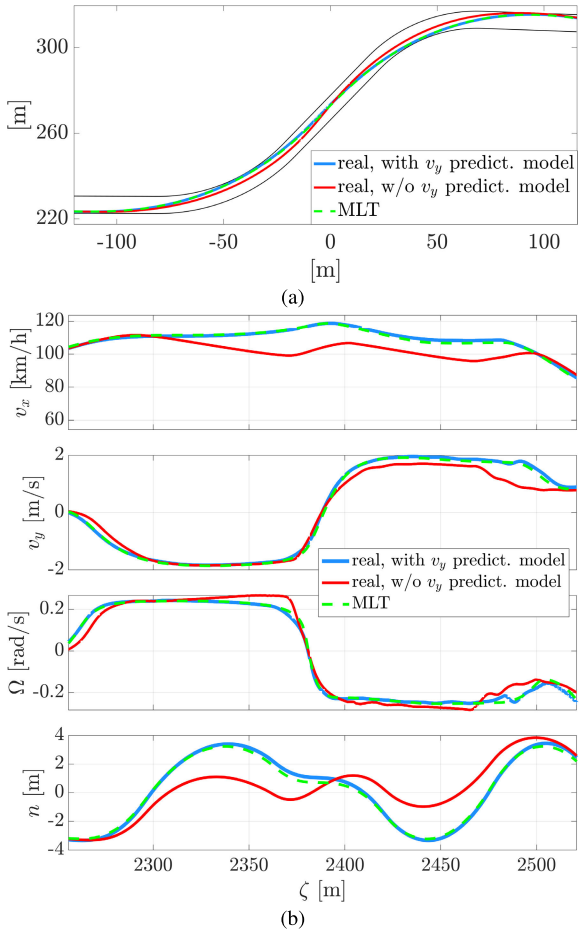


FIGURE 19. Comparison of trajectories executed by the artificial agent, when the v_y prediction model for E-NMPC is used (blue line) or not (red line), on the corners n.9 and n.10 of the Valencia circuit: (a) vehicle trajectories, and (b) linear velocities $\{v_x, v_y\}$, yaw rate Ω and lateral coordinate n of the vehicle. When the v_y prediction model is used, the executed trajectories (blue lines) are a lot closer to the MLT benchmark, and the maneuver time is lower by 0.65 s.

The planned and executed vehicle and state trajectories are close to the MLT benchmark solution, and the G-G-v constraint is exploited in its entirety (Fig. 21). As shown in Fig. 20(a), in certain corners the agent replans wider or narrower trajectories, depending on the current vehicle states. The lap time with online closed-loop control is $T_{\text{MPC}} = 109.648$ s, while the MLT solution is $T_{\text{MLT}} = 109.473$ s. The small lap time difference (175 ms) – mainly caused by low-level tracking errors – suggests that the framework learned vehicle dynamic characteristics that are independent of the racetrack. Such a robustness with respect to the circuit layout should not be taken for granted: neural networks with generic internal structures or non-parametric approaches (e.g. Gaussian process regression) may need large training sets, to provide robust predictions on test data that differ significantly from the training scenarios. Conversely, the physics-based formulations of the control models is less prone to overfitting, and it shows an improved generalization capability.

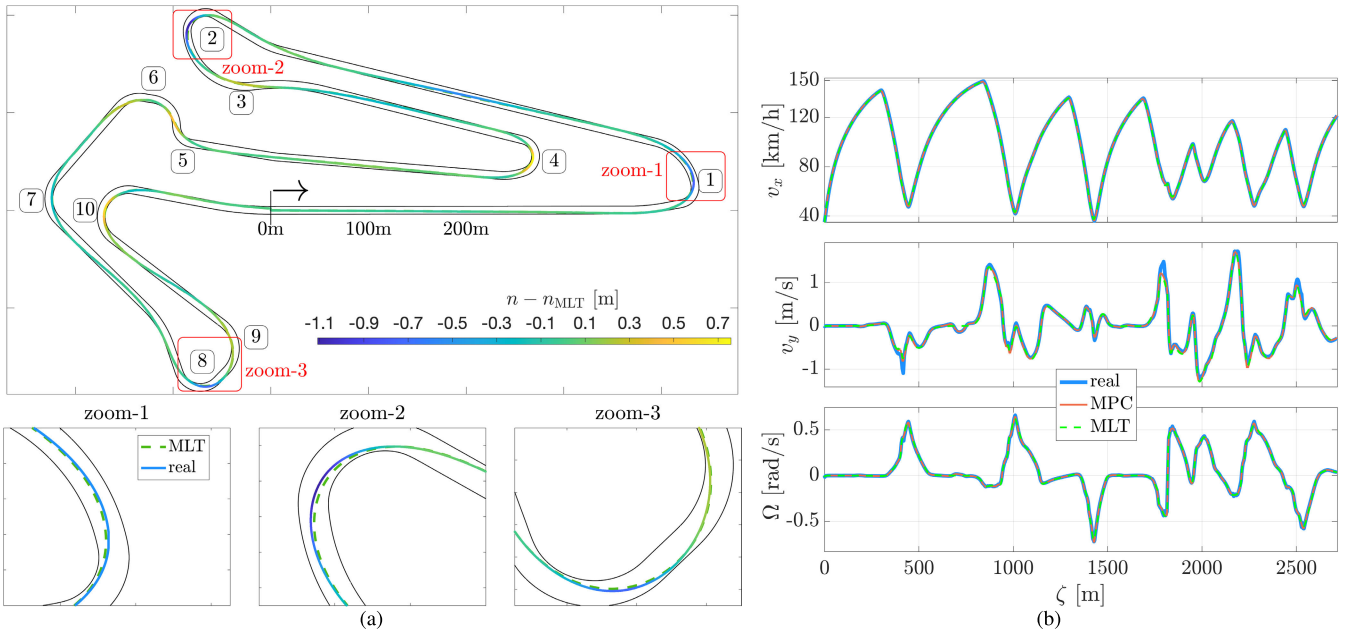


FIGURE 20. Results on the Adria circuit, never used in the identification and learning phases: (a) comparison of the trajectories executed by the artificial agent ("real", colored solid line) and computed by the offline OC problem ("MLT", green dashed line). A color map is used to highlight the difference $n - n_{\text{MLT}}$, with n and n_{MLT} being the lateral coordinates of the executed trajectory and the MLT solution. (b) Comparison of the planned (orange), executed (blue) and benchmark ("MLT", green dashed) linear velocities $\{v_x, v_y\}$ and yaw rate Ω.

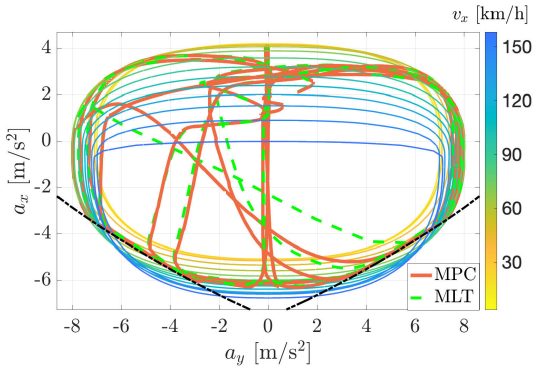


FIGURE 21. Resulting G-G-v diagram on the Adria circuit.

TABLE 7. Analyzing the effect of a change in the vehicle parameters m, h_G, L_1 on the lap time T_{MPC}, obtained by the artificial agent on the Valencia circuit. In each scenario, the changed parameters are underlined.

Test ID	m [kg]	h_G [m]	L_1 [m]	T_{MLT} [s]	T_{MPC} [s]
1 (nomin.)	1296	0.285	1.23	141.139	141.363
2	1096	0.285	1.23	140.582	141.112
3	1496	0.285	1.23	141.612	142.124
4	1296	0.240	1.23	-	141.148
5	1296	0.330	1.23	-	141.864
6	1296	0.285	1.08	-	141.322
7	1296	0.285	1.38	-	142.097

2) CHANGING THE VEHICLE PARAMETERS

We now analyze the robustness of the planning and control framework to a change of some vehicle parameters. Specifically, we focus on the parameters $\{m, h_G, L_1\}$, being respectively the vehicle mass, the center of mass height from ground, and the longitudinal distance from the center of mass to the front axle. These parameters were assumed to be known in advance (Section II-A), since they eased the model-based design of the speed-tracking controller, but we now show that the framework is sufficiently robust to changes of such quantities.

The parameter-variation analyses are reported in Table 7, where we underline the parameters that we change in each test scenario. The high- and low-level control models are not modified with respect to the nominal conditions (scenario n.1 in Table 7). The planning and control framework is therefore unaware of the parameter variations.

A change in the parameters m, h_G and L_1 affects the whole vehicle behavior: for example, it results in different dynamics of $\{v_x, v_y, \Omega\}$, it modifies the shape of the handling and lateral velocity diagrams, and it impacts on the load transfers.

We analyze the effect of the parameters variation on the lap time T_{MPC} (Table 7, last column) obtained by the artificial agent, and on the vehicle trajectory, shown in Fig. 22 and 23. The lap time T_{MLT} of the benchmark offline MLT problem is recomputed only for the two extreme scenarios in which the vehicle mass is changed.[16]

[16]The change in the vehicle mass affects the longitudinal and lateral dynamics of the MLT vehicle model, and the G-G-v constraint. Such parameters and constraints of the MLT model are re-identified, using the automatic procedure of Section IV, but they are used only to recompute the benchmark MLT problem. The models used by the artificial agent are instead not modified with respect to the nominal case.

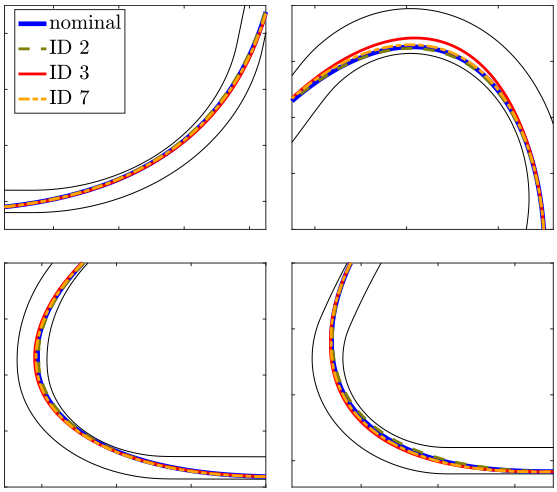


FIGURE 22. Comparison of the vehicle trajectories obtained in the nominal conditions and in the scenarios n.2, 3, 7 (Table 7), in which the vehicle mass m and the front/rear load distribution are changed. The plots show only the corners in which the differences are more visible, namely the corners n.1 (top left), n.2 (top right), n.8 (bottom left) and n.14 (bottom right) of the Valencia circuit.

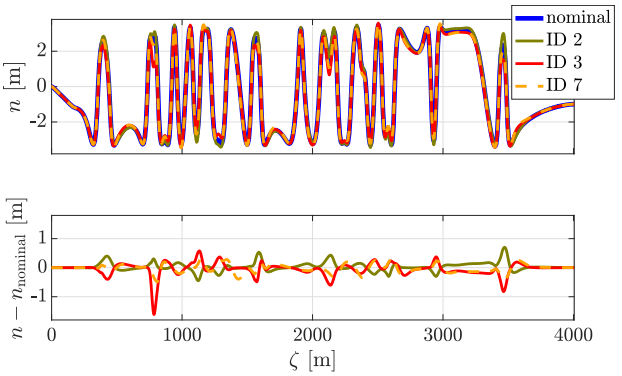


FIGURE 23. Lateral coordinate of the vehicle center: comparison of the solutions obtained in the nominal conditions and in the parameter-variation scenarios n.2, 3, 7 (Table 7), Valencia circuit.

In the scenarios n.2 and 3 of Table 7, the vehicle sprung mass is varied by 200 kg. If m is decreased (case n.2), the vehicle becomes more responsive, and it can quickly recover possible speed-tracking delays or other mismatches. The lap time T_{MPC} reached by the artificial agent is lower (by 0.25 s) than the nominal case n.1. The deviation of T_{MPC} from the offline MLT lap time T_{MLT} is around 0.5 s, which is slightly higher than the nominal case. However, considering that the artificial agent is not re-trained for the new conditions, the result shows an acceptable robustness of the agent to a decrease in the mass. Conversely, if m is increased by 200 kg (case n.3), the desired longitudinal accelerations computed by the E-NMPC may be locally unreachable for the low-level control, the impacts on the lateral dynamics are not negligible, and the lap time is higher (by 0.75 s) than the nominal case. Fig. 22 and 23 show that, as explained in Section VI-A, the high-level E-NMPC problem locally changes the planned time-optimal trajectories, to compensate different initial yaw angles, in the entry phases of some corners. Specifically, when the mass is increased, the yaw angles of the executed maneuvers are locally slightly different from the planned values, which forces the agent to replan wider trajectories in certain corners (Fig. 22), to minimize the maneuver time and satisfy the G-G-v constraint, while considering the current initial vehicle states. Nonetheless, the framework is still capable of controlling the vehicle, and the difference $T_{\mathrm{MPC}} - T_{\mathrm{MLT}}$ is around 0.5 s.

In the scenarios n.4 and 5, we vary the center of mass height h_G by 0.045 m. A higher h_G leads to larger longitudinal (and lateral) load transfers, which harms the traction performance in a front-wheel-drive vehicle. The lap time is consequently higher (by 0.5 s) than the nominal case, but the vehicle can still be controlled. On the other hand, lowering h_G (case n.4) stabilizes the vehicle, leading to low trajectory tracking errors and better lap times.

Finally, the scenarios n.6 and 7 involve changing the front/rear load distribution of the vehicle, with the parameter L_1 being varied by 0.15 m. When L_1 is decreased (case n.6), the vehicle gets more understeering, and the vertical loads on the front tires increase, which improves the traction potential. The lap time is very similar to the nominal case. Conversely, an increase in L_1 moves more vertical load to the rear axle, which decreases the traction and steering capabilities of the front tires. The lap time is therefore higher (by 0.7 s) than the nominal case, but the trajectory deviations are limited (Fig. 23).

The analyses reported in this section show that the motion planning and control framework is sufficiently robust to tolerate unknown variations in the vehicle mass and load distribution (h_G and L_1). The low-level feedback steering and longitudinal controllers compensate for part of the unmodeled dynamics, while the high-level E-NMPC locally adapts the planned trajectories to the current initial conditions. We underline that we do *not* force the E-NMPC solver to track a pre-computed fixed trajectory, which lets the E-NMPC compute new optimal solutions, in the presence of mismatches or disturbances.

VII. CONCLUSION AND FUTURE WORK

This paper develops physics-driven analytical and neural models, which are integrated in a hierarchical online minimum-time planning and control scheme for a partially-unknown vehicle simulator. A kineto-dynamical E-NMPC model for high-level planning is extended with a novel lateral speed predictor. A feedforward neural network with a physics-driven structure is devised for the low-level steering control, in combination with a feedback regulator. A low-level speed-tracking PID controller is tuned with an identified longitudinal dynamic model. A three-step automatic identification procedure is presented, to identify and learn the high- and low-level models, optimize the shape of a G-G-v constraint for E-NMPC, and learn the dependency of the lateral dynamics on the longitudinal acceleration.

Simulations are carried out with a partially-unknown vehicle simulator, whose dynamics are identified and learned using the devised three-step scheme. The results obtained with online motion planning and control are very similar to the offline MLT solution, used as a benchmark. The learned achievable performance (G-G-v envelope) is exploited up to the limits. A close matching exists between the planned and executed vehicle velocities and accelerations, indicating that the steering and longitudinal controllers effectively track the high-level optimal trajectories. Finally, it is shown that the framework is sufficiently robust, when driving on circuits never used in the learning process, and even when changing the vehicle mass and load distribution. The generalization and adaptation capability of the framework is mainly provided by the physics-driven structure of the control models, the feedback regulators, and the long E-NMPC planning horizon.

Future work will be applying the proposed framework to a real RC vehicle prototype. Testing on a real car will allow us to further optimize the identification and learning phases, by reducing the number of open- and closed-loop maneuvers to be carried out. We will also investigate the framework's robustness to measurement noise and communication delays. Moreover, the high- and low-level models will be further improved, to deal with changing road adherence conditions.

APPENDIX A
DERIVATION OF THE LOW-LEVEL FEEDFORWARD STEERING CONTROLLER

In this appendix, we provide more details about the formulation of the neural steering controller NNc_{ext}, which was introduced in Section III-B2.

The analysis here presented aims at finding some insightful physics-driven relations, to inspire the design of the neural controller NNc_{ext}. Our final goal is to extend the neural network NNc_{base}, described in Section III-B1, by considering the coupled effects of the lateral accelerations a_y and the longitudinal ones a_x. Specifically, we augment the local model (13), which was derived as a local linearization of the handling diagram for the pure lateral dynamics, with additional terms that depend on both a_y and a_x. The augmented local models are formulated by extending the analysis of [52, Chapter 7]: we perform local approximations of the nonlinear lateral dynamics of a double-track vehicle model, around generic quasi steady-state conditions.

Throughout this section, the notation X_{ij} will be used to indicate the generic quantity X for the right ($j = 1$) or left ($j = 2$) tires, of the front ($i = 1$) or rear ($i = 2$) axles.

Let us begin by recalling that the tire lateral slip angles α_{ij} can be derived from a kinematic analysis (see e.g. [41]):

$$\begin{cases} \alpha_{11} \approx \alpha_{12} = \alpha_1 = \arctan\left(-\dfrac{v_y}{v_x} - L_1\rho + \delta\right) \\ \alpha_{21} \approx \alpha_{22} = \alpha_2 = \arctan\left(-\dfrac{v_y}{v_x} + L_2\rho\right) \end{cases} \tag{31}$$

where $\{L_1, L_2\}$ are the distances from the center of mass to the front and rear axles, $\rho = a_y/v_x^2$ is the trajectory curvature, and δ is the steering angle at the front wheels (the vehicle is front-wheel steering). It is assumed that the lateral slips of the tires within the same axle are approximately equal, so that $\{\alpha_1, \alpha_2\}$ indicate the lateral slips of the front and rear tires. Performing a first-order approximation of the arctan$(\cdot)$ function, it is possible to rewrite (31) as:

$$\delta - \rho L = \delta - \frac{a_y}{v_x^2} L = \alpha_1 - \alpha_2 \tag{32}$$

with $L = L_1 + L_2$.

To augment the local model (13), our target is now to derive a relation between the lateral slips $\{\alpha_1, \alpha_2\}$ and the accelerations $\{a_y, a_x\}$, in the vicinity of an approximation point with $\{a_y = a_{y_0}, a_x = a_{x_0}\}$.

We start from the lateral nonlinear dynamics of a double-track vehicle model in a quasi steady-state condition:

$$\begin{cases} m a_{y_0} = F_{y_{11_0}} + F_{y_{12_0}} + F_{y_{21_0}} + F_{y_{22_0}} & \text{(33a)} \\ I_z \dot{\Omega} = 0 = (F_{y_{11_0}} + F_{y_{12_0}})L_1 - (F_{y_{21_0}} + F_{y_{22_0}})L_2 & \text{(33b)} \end{cases}$$

where $F_{y_{ij}}$ indicates the lateral tire forces for the right ($j = 1$) and left ($j = 2$) tires of the front ($i = 1$) and rear ($i = 2$) axles, while I_z and $\dot{\Omega}$ are the vehicle yaw inertia and the yaw acceleration. (33a) and (33b) are respectively the lateral and yaw rate Newton equations[17] in a quasi steady-state point $\mathcal{W}$, for which $a_y = a_{y_0}$, $a_x = a_{x_0}$, $F_{y_{ij}} = F_{y_{ij_0}}$, $\dot{\Omega} \approx 0$. (33a,33b) can be equivalently rewritten as:

$$\begin{cases} m a_{y_0} \dfrac{L_2}{L} = F_{y_{11_0}} + F_{y_{12_0}} & \text{(34a)} \\ m a_{y_0} \dfrac{L_1}{L} = F_{y_{21_0}} + F_{y_{22_0}} & \text{(34b)} \end{cases}$$

Before performing a local approximation of the lateral force model, we introduce an ellipse of adherence [52] to express the effect of the longitudinal tire forces $F_{x_{ij}}$ on the lateral ones $F_{y_{ij}}$:

$$F_{y_{ij}} = \bar{F}_{y_{ij}}(\alpha_i, F_{z_{ij}}) \sqrt{1 - \left(\frac{\Delta F_{x_{ij}}}{\mu_{x_{ij}}^{\max} F_{z_{ij}}}\right)^2} \tag{35}$$

where $\bar{F}_{y_{ij}}(\cdot)$ is the pure lateral tire force, which is a nonlinear function of the tire lateral slip α_i and of the vertical tire load $F_{z_{ij}}$. The maximum pure longitudinal tire force is $\mu_{x_{ij}}^{\max} F_{z_{ij}}$, while $\Delta F_{x_{ij}} = F_{x_{ij}} - F_{x_{ij_0}}$ is the variation of $F_{x_{ij}}$ with respect to the steady-state value $F_{x_{ij_0}}$. The variation $\Delta F_{x_{ij}}$ derives from a reformulation of the pure longitudinal dynamics (23a):

$$\begin{aligned} m a_x &= \sum_{i=1}^{2}\sum_{j=1}^{2} F_{x_{ij}} - c_0 - c_v v_x - c_a v_x^2 \\ &= \sum_{i=1}^{2}\sum_{j=1}^{2} \left(F_{x_{ij_0}} + \Delta F_{x_{ij}}\right) - c_0 - c_v v_x - c_a v_x^2 \\ &= \sum_{i=1}^{2}\sum_{j=1}^{2} \Delta F_{x_{ij}} \end{aligned} \tag{36}$$

[17] Note that (33b) could be complicated with the addition of other minor contributions, like the tire self-aligning torques, and the analysis of this section would still hold.

where the last step in (36) assumes that the sum of the $F_{x_{ij_0}}$ forces compensates the friction and drag terms. Moreover, it is assumed that $\Delta F_{x_{ij}}$ has a larger effect on the lateral dynamics, with respect to $F_{x_{ij_0}}$, which explains the use of $\Delta F_{x_{ij}}$ in (35). Starting from (36), the terms $\Delta F_{x_{ij}}$ can be expressed as a function of a_x:

$$\begin{cases} \Delta F_{x_{1j}} = \Gamma_p \Gamma_{d_j} m a_x \\ \Delta F_{x_{2j}} = (1-\Gamma_p)\Gamma_{d_j} m a_x \end{cases} \tag{37}$$

with $j \in \{1, 2\}$ indicating the right and left tires. $\{\Gamma_p, \Gamma_{d_j}\} \in (0, 1)$ are the traction/brake force distribution between respectively the front-rear axles and the right-left tires. Recalling (25a), the vertical tire loads $F_{z_{ij}}$ appearing in (35) depend in turn on a_x. The lateral force model (35) can therefore be rewritten by replacing the square root term with a nonlinear function of a_x, here named $G_{x_{ij}}(a_x)$:

$$F_{y_{ij}} = \bar{F}_{y_{ij}}(\alpha_i, F_{z_{ij}}) \cdot G_{x_{ij}}(a_x) \tag{38}$$

We locally approximate the combined lateral force model (38), around the quasi steady-state $\mathcal{W}$: the pure lateral force model $\bar{F}_{y_{ij}}(\alpha_i, F_{z_{ij}})$ and $G_{x_{ij}}(a_x)$ are linearized in the neighborhood of respectively α_{i_0} and a_{x_0}. The approximate local model becomes:

$$F_{y_{ij}} = \Big(Y_{ij} + C_{y_{ij}}(\alpha_i - \alpha_{i_0})\Big) \cdot \Big(b_{ij} + c_{ij}(a_x - a_{x_0})\Big) \tag{39}$$

where $\{Y_{ij}, b_{ij}, c_{ij}\}$ are constant parameters defining the local approximation. The parameter $C_{y_{ij}}$ in (39) can be seen as a generalized cornering stiffness, in that it is the local slope of the nonlinear pure lateral force model, for a generic α_{i_0}. Following [52, Chapter 7], the cornering stiffnesses depend approximately linearly on the vertical loads $F_{z_{ij}}$, and such a linear dependency can be assumed to hold also for $C_{y_{ij}}$:

$$C_{y_{ij}} = C_{y_{i0}} + \frac{\mathrm{d}C_{y_i}}{\mathrm{d}F_z} \cdot \left((-1)^{j-1}\Delta F_{z_{Yi}} + (-1)^i \frac{\Delta F_{z_X}}{2}\right) \tag{40}$$

where $C_{y_{i0}}$ is the nominal value of $C_{y_{ij}}$ in the quasi steady-state condition $\mathcal{W}$, $\frac{\mathrm{d}C_{y_i}}{\mathrm{d}F_z}$ is the vertical load sensitivity, $\{\Delta F_{z_{Yi}}, \Delta F_{z_X}\}$ are the variations of the lateral and longitudinal load transfers with respect to the point $\mathcal{W}$. In the vicinity of the quasi steady-state $\mathcal{W}$, the load transfer variations $\Delta F_{z_{Yi}}$ and ΔF_{z_X} are directly proportional to respectively $(a_y - a_{y_0})$ and $(a_x - a_{x_0})$ [41, Chapter 6]:

$$\begin{cases} \Delta F_{z_{Yi}} = \eta_{y_i}(a_y - a_{y_0}) \\ \Delta F_{z_X} = m\dfrac{h_G}{L}(a_x - a_{x_0}) \end{cases} \tag{41}$$

with η_{y_i} being a proportionality factor that depends on the vehicle characteristics, and h_G being the center of mass height. The expression of $C_{y_{ij}}$ in (40) can be therefore be written as a function of $(a_y - a_{y_0})$ and $(a_x - a_{x_0})$.

In the quasi steady-state $\mathcal{W}$, the relations (34a, 34b) hold, and (39) becomes $F_{y_{ij_0}} = Y_{ij} b_{ij}$. In the neighborhood of $\mathcal{W}$, (34a) and (39) can be used to write the following:

$$\begin{aligned} m(a_y - a_{y_0})\frac{L_2}{L} &= F_{y_{11}} + F_{y_{12}} - (F_{y_{11_0}} + F_{y_{12_0}}) \\ &= \sum_{j=1}^{2} \Big(Y_{1j} + C_{y_{1j}}(\alpha_1 - \alpha_{1_0})\Big)\Big(b_{1j} + c_{1j}(a_x - a_{x_0})\Big) + \\ &\quad - Y_{1j} b_{1j} \end{aligned} \tag{42}$$

The previous equation can be solved for α_1:

$$\alpha_1 = \frac{m(a_y - a_{y_0})\frac{L_2}{L} + \alpha_{1_0} D_1 - \sum_{j=1}^{2} Y_{1j} c_{1j}(a_x - a_{x_0})}{D_1} \tag{43}$$

where the term D_1 is given by:

$$D_1 = \sum_{j=1}^{2} C_{y_{1j}}\big(b_{1j} + c_{1j}(a_x - a_{x_0})\big) \tag{44}$$

To obtain a relation for α_1 that depends only on a_y, a_x and constant terms, we can now substitute the expressions (40) and (41) for $C_{y_{1j}}$ in (43). The resulting equation can be written as:

$$\begin{cases} B_{\alpha_1} = \tilde{q}_{y_1} + \tilde{q}_{y_2}(a_y - a_{y_0}) + \tilde{q}_{x_1}(a_x - a_{x_0}) \\ \quad + \tilde{q}_{x_2}(a_x - a_{x_0})^2 + \\ \quad + \tilde{q}_{y_3}\tilde{q}_{x_3}(\tilde{q}_{y_4} + a_y - a_{y_0})(\tilde{q}_{x_4} + a_x - a_{x_0}) & \text{(45a)} \\ A_{\alpha_1} = \tilde{q}_{y_5}(a_y - a_{y_0}) + \tilde{q}_{x_5}(a_x - a_{x_0}) \\ \quad + \tilde{q}_{x_6}(a_x - a_{x_0})^2 + \tilde{q}_{y_6}\tilde{q}_{x_7}(a_y - a_{y_0})(a_x - a_{x_0}) & \text{(45b)} \\ \alpha_1 = B_{\alpha_1}/(1 + A_{\alpha_1}) & \text{(45c)} \end{cases}$$

where the new parameters $\{\tilde{q}_{y_1}, \ldots, \tilde{q}_{y_6}\}$ and $\{\tilde{q}_{x_1}, \ldots, \tilde{q}_{x_7}\}$ are introduced to collect the combinations of the constant terms in (43). The subscripts y and x are used in the parameters $\tilde{q}_{y\_}$ and $\tilde{q}_{x\_}$ to preserve the separation of the terms related to the lateral and longitudinal dynamics. The formulation (45c) is a local model for α_1, with $\{a_y, a_x\}$ in the vicinity of $\{a_{y_0}, a_{x_0}\}$.

The denominator term $1 + A_{\alpha_1}$ in (45c) can be manipulated using the first-order Taylor approximation $(1 + A_{\alpha_1})^{-1} \approx 1 - A_{\alpha_1}$, which holds for a small A_{α_1} (i.e., for $\{a_y, a_x\}$ in the vicinity of $\{a_{y_0}, a_{x_0}\}$). The expression of α_1 in (45c) becomes $B_{\alpha_1}(1 - A_{\alpha_1})$, which can be written as:

$$\begin{cases} \alpha_1 = q_{y_1} + q_{y_2}(a_y - a_{y_0}) + q_{y_3} q_{x_1}(q_{y_4} + a_y + -a_{y_0}) \\ \quad \times (q_{x_2} + a_x - a_{x_0})\Big[1 + q_{y_5}(a_y - a_{y_0}) \\ \quad + q_{x_3}(a_x - a_{x_0}) + q_{x_4}(a_x - a_{x_0})^2 + \\ \quad + q_{y_6} q_{x_5}(a_y - a_{y_0})(a_x - a_{x_0})\Big] + Q_{\alpha_1} & \text{(46a)} \\ Q_{\alpha_1} = q_{y_7}(a_y - a_{y_0})^2 + \sum_{i=1}^{4} q_{x_{5+i}}(a_x - a_{x_0})^i & \text{(46b)} \end{cases}$$

where the parameters $\{q_{y_1}, \ldots, q_{y_7}\}$ and $\{q_{x_1}, \ldots, q_{x_9}\}$ in turn contain combinations of the $\{\tilde{q}_{y_1}, \ldots, \tilde{q}_{y_6}\}$ and $\{\tilde{q}_{x_1}, \ldots, \tilde{q}_{x_7}\}$ used in (45a). The Q_{α_1} term in (46a) is isolated because,

as discussed next, it does not improve the accuracy of the final model, and it will therefore be neglected.

The analysis outlined above can be reapplied for the case of the rear tires, starting from (34b), and a relation of the same type of (46a) will be found for α_2 as well. In the case of α_2, the parameters $\{q_{y_1}, \dots, q_{y_7}, q_{x_1}, \dots, q_{x_9}\}$ used in (46a) for α_1 need to be replaced with a different set, which we call $\{r_{y_1}, \dots, r_{y_7}, r_{x_1}, \dots, r_{x_9}\}$:

$$\begin{cases} \alpha_2 = r_{y_1} + r_{y_2}(a_y - a_{y_0}) + r_{y_3} r_{x_1}(r_{y_4} + a_y + -a_{y_0}) \\ \quad \times (r_{x_2} + a_x - a_{x_0})\Big[1 + r_{y_5}(a_y - a_{y_0}) \\ \quad + r_{x_3}(a_x - a_{x_0}) + r_{x_4}(a_x - a_{x_0})^2 \\ \quad + r_{y_6} r_{x_5}(a_y - a_{y_0})(a_x - a_{x_0})\Big] + Q_{\alpha_2} \end{cases} \tag{47a}$$

$$Q_{\alpha_2} = r_{y_7}(a_y - a_{y_0})^2 + \sum_{i=1}^{4} r_{x_{5+i}}(a_x - a_{x_0})^i \tag{47b}$$

To augment the local model (13), initially conceived for the pure lateral dynamics, the resulting formulation (46a)-(47a) for the tire lateral slips $\{\alpha_1, \alpha_2\}$ can now be substituted in (32), which yields the following:

$$\begin{aligned} &\delta - \frac{a_y}{v_x^2} L \\ &= \alpha_1 - \alpha_2 = \mathrm{k}_{y_1} \mathrm{sign}(a_y) + \mathrm{k}_{y_2}\Big(a_y - a_{y_0} \mathrm{sign}(a_y)\Big) \\ &\quad + \mathrm{k}_{y_3} \mathrm{k}_{x_1}\Big(a_y - (a_{y_0} + \mathrm{k}_{y_4}) \mathrm{sign}(a_y)\Big)\Big(\mathrm{k}_{x_2} + a_x - a_{x_0}\Big) \\ &\quad \times \Big[1 + \mathrm{k}_{y_5}\Big(a_y - a_{y_0} \mathrm{sign}(a_y)\Big) + \mathrm{k}_{x_3}(a_x - a_{x_0}) \\ &\quad + \mathrm{k}_{x_4}(a_x - a_{x_0})^2 + \mathrm{k}_{y_6} \mathrm{k}_{x_5}\Big(a_y - a_{y_0} \mathrm{sign}(a_y)\Big)(a_x - a_{x_0})\Big] \end{aligned} \tag{48}$$

where $\{k_{y_1}, \dots, k_{y_6}, k_{x_1}, \dots, k_{x_5}\}$ is the set of parameters describing the local model. Such parameters can be seen as functions of the $\{q_{y_1}, \dots, q_{y_6}, q_{x_1}, \dots, q_{x_5}\}$ and $\{r_{y_1}, \dots, r_{y_6}, r_{x_1}, \dots, r_{x_5}\}$ used for α_1 and α_2. Similarly to what was done in the basic local model (13), the sign functions are used in (48) to model a symmetric lateral behavior, for $a_y > 0$ and $a_y < 0$. Note that the contributions of the $\{Q_{\alpha_1}, Q_{\alpha_2}\}$ terms in (46a)-(47a) are neglected in (48), since they do not significantly improve the accuracy of (48). More specifically, the terms of (46a)-(47a) involving powers of a_x improve the accuracy only when multiplied by an expression in a_y. Such a result has a physical explanation: since (48) is used to compute the steering angle δ, if the vehicle has no steering biases then the model should return $\delta = 0$ when the desired a_y is zero, for any value of a_x.

The local model formulation (48) is an extension of (13), in that it allows modeling the effect of the longitudinal acceleration a_x on the lateral dynamics, in a neighborhood of $\{a_{y_0}, a_{x_0}\}$. The derivation of (48) is physics-driven, in the sense that we started from the double-track model equations and exploited the laws of vehicle dynamics. The resulting formulation, when used inside the neural controller $\mathrm{NNc}_{\mathrm{ext}}$ of Section III-B2, has fewer parameters and higher accuracy with respect to black-box generic neural networks.

We remark that the model (48) could be further extended by adding other terms, that derive from a generalization of our analysis. For example, higher-order approximations can be performed in (39) and (40). However, we found that such additional terms do not improve significantly the model accuracy, meaning that the formulation (48) already represents a good trade-off between accuracy and complexity.

APPENDIX B
AUXILIARY FUNCTIONS

This section defines the auxiliary smooth functions adopted to define the high and low-level control models.

$$\begin{cases} \mathrm{signReg} := x \mapsto \sin(\arctan(x/h_1)) \\ \mathrm{pSgn} := x \mapsto \dfrac{\mathrm{signReg}(x) + 1}{2} \\ \mathrm{nSgn} := x \mapsto \dfrac{\mathrm{signReg}(x) - 1}{2} \\ \mathrm{pPart} := x \mapsto x \cdot \dfrac{\tanh(x/h_2) + 1}{2} \\ \mathrm{nPart} := x \mapsto x \cdot \dfrac{\tanh(x/h_2) - 1}{2} \\ \mathrm{absReg} := x \mapsto x \cdot \mathrm{signReg}(x) \end{cases} \tag{49}$$

In (49), $\{h_1, h_2\}$ are factors influencing the smoothness of the functions and their derivatives, and they are selected in a preliminary modeling stage.

ACKNOWLEDGMENT

The authors would like to thank Gastone Pietro Papini Rosati and Mauro Da Lio for the fruitful discussions and critical reviews of the manuscript. They would also grateful to Luca Gasbarro, Leone Pasquato, and Giammarco Valenti, from AnteMotion Srl company, for their support in the development of the presented framework.

REFERENCES

[1] J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam, "Autonomous vehicles on the edge: A survey on autonomous vehicle racing," *IEEE Open J. Intell. Transp. Syst.*, vol. 3, pp. 458–488, 2022.

[2] M. Piccinini, M. Larcher, E. Pagot, D. Piscini, L. Pasquato, and F. Biral, "A predictive neural hierarchical framework for on-line time-optimal motion planning and control of black-box vehicle models," *Vehicle Syst. Dyn.*, vol. 61, no. 1, pp. 83–110, Jan. 2023.

[3] G. Hartmann, Z. Shiller, and A. Azaria, "Competitive driving of autonomous vehicles," *IEEE Access*, vol. 10, pp. 111772–111783, 2022.

[4] M. Rokonuzzaman, N. Mohajer, S. Nahavandi, and S. Mohamed, "Model predictive control with learned vehicle dynamics for autonomous vehicle path tracking," *IEEE Access*, vol. 9, pp. 128233–128249, 2021.

[5] T. Novi, A. Liniger, R. Capitani, and C. Annicchiarico, "Real-time control for at-limit handling driving on a predefined path," *Vehicle Syst. Dyn.*, vol. 58, no. 7, pp. 1007–1036, Jul. 2020.

[6] F. Altché, P. Polack, and A. de La Fortelle, "High-speed trajectory planning for autonomous vehicles using a simple dynamic model," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2017, pp. 1–7.

[7] E. Pagot, M. Piccinini, and F. Biral, "Real-time optimal control of an autonomous RC car with minimum-time maneuvers and a novel kineto-dynamical model," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 2390–2396.

[8] A. Wischnewski, T. Herrmann, F. Werner, and B. Lohmann, "A tube-MPC approach to autonomous multi-vehicle racing on high-speed ovals," *IEEE Trans. Intell. Vehicles*, vol. 8, no. 1, pp. 368–378, Jan. 2023.

[9] J. L. Vázquez, M. Bruhlmeier, A. Liniger, A. Rupenyan, and J. Lygeros, "Optimization-based hierarchical motion planning for autonomous racing," 2020, *arXiv:2003.04882*.

[10] U. Rosolia and F. Borrelli, "Learning how to autonomously race a car: A predictive control approach," *IEEE Trans. Control Syst. Technol.*, vol. 28, no. 6, pp. 2713–2719, Nov. 2020.

[11] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3363–3370, Oct. 2019.

[12] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optim. Control Appl. Methods*, vol. 36, no. 5, pp. 628–647, Sep. 2015.

[13] H. Pacejka, *Tire and Vehicle Dynamics*, 3rd ed. Amsterdam, The Netherlands: Elsevier, 2012.

[14] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Vehicles*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[15] L. Grune and J. Pannek, *Nonlinear Model Predictive Control: Theory and Algorithms*. Berlin, Germany: Springer, 2013.

[16] T. Faulwasser, L. Grune, and M. A. Muller, "Economic nonlinear model predictive control," *Found. Trends Syst. Control*, vol. 5, no. 1, pp. 1–98, 2018.

[17] R. Verschueren, M. Zanon, R. Quirynen, and M. Diehl, "Time-optimal race car driving using an online exact Hessian based nonlinear MPC algorithm," in *Proc. Eur. Control Conf. (ECC)*, Jun. 2016, pp. 141–147.

[18] K. Berntorp and F. Magnusson, "Hierarchical predictive control for ground-vehicle maneuvering," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2015, pp. 2771–2776.

[19] M. Da Lio, R. Dona, G. P. R. Papini, F. Biral, and H. Svensson, "A mental simulation approach for learning neural-network predictive control (in self-driving cars)," *IEEE Access*, vol. 8, pp. 192041–192064, 2020.

[20] F. Biral, E. Bertolazzi, D. Bortoluzzi, and P. Bosetti, "Development and testing of an autonomous driving module for critical driving conditions," in *Proc. ASME Int. Mech. Eng. Congr. Expo.*, Jan. 2008, pp. 361–370.

[21] M. Veneri and M. Massaro, "A free-trajectory quasi-steady-state optimal-control method for minimum lap-time of race vehicles," *Vehicle Syst. Dyn.*, vol. 58, no. 6, pp. 933–954, Jun. 2020.

[22] F. Altche, P. Polack, and A. de La Fortelle, "A simple dynamic model for aggressive, near-limits trajectory planning," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 141–147.

[23] J. K. Subosits and J. C. Gerdes, "From the racetrack to the road: Real-time trajectory replanning for autonomous driving," *IEEE Trans. Intell. Vehicles*, vol. 4, no. 2, pp. 309–320, Jun. 2019.

[24] J. Betz, T. Betz, F. Fent, M. Geisslinger, A. Heilmeier, L. Hermansdorfer, T. Herrmann, S. Huch, P. Karle, M. Lienkamp, B. Lohmann, F. Nobis, L. Ögretmen, M. Rowold, F. Sauerbeck, T. Stahl, R. Trauth, F. Werner, and A. Wischnewski, "TUM autonomous motorsport: An autonomous racing software for the Indy Autonomous Challenge," *J. Field Robot.*, vol. 40, no. 4, pp. 783–809, Jun. 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.22153

[25] J. K. Subosits and J. C. Gerdes, "Impacts of model fidelity on trajectory optimization for autonomous vehicles in extreme maneuvers," *IEEE Trans. Intell. Vehicles*, vol. 6, no. 3, pp. 546–558, Sep. 2021.

[26] L. Hewing, A. Liniger, and M. N. Zeilinger, "Cautious NMPC with Gaussian process dynamics for autonomous miniature race cars," in *Proc. Eur. Control Conf. (ECC)*, Jun. 2018, pp. 1341–1348.

[27] N. A. Spielberg, M. Brown, and J. C. Gerdes, "Neural network model predictive motion control applied to automated driving with unknown friction," *IEEE Trans. Control Syst. Technol.*, vol. 30, no. 5, pp. 1934–1945, Sep. 2022.

[28] N. D. Bianco, E. Bertolazzi, F. Biral, and M. Massaro, "Comparison of direct and indirect methods for minimum lap time optimal control problems," *Vehicle Syst. Dyn.*, vol. 57, no. 5, pp. 665–696, May 2019.

[29] S. E. Samada, V. Puig, and F. Nejjari, "Robust TS-ANFIS MPC of an autonomous racing electrical vehicle considering the battery state of charge," *IEEE/ASME Trans. Mechatronics*, vol. 28, no. 2, pp. 656–667, Apr. 2023.

[30] X. Jin, J. Wang, X. He, Z. Yan, L. Xu, C. Wei, and G. Yin, "Improving vibration performance of electric vehicles based on in-wheel motor-active suspension system via robust finite frequency control," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 2, pp. 1631–1643, Feb. 2023.

[31] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 2, pp. 712–733, Feb. 2021.

[32] G. Devineau, P. Polack, F. Altché, and F. Moutarde, "Coupled longitudinal and lateral control of a vehicle using deep learning," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 642–649.

[33] S. Löckel, J. Peters, and P. van Vliet, "A probabilistic framework for imitating human race driver behavior," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2086–2093, Apr. 2020.

[34] N. A. Spielberg, M. Brown, N. R. Kapania, J. C. Kegelman, and J. C. Gerdes, "Neural network vehicle models for high-performance automated driving," *Sci. Robot.*, vol. 4, no. 28, Mar. 2019, Art. no. eaaw1975.

[35] M. D. Lio, D. Bortoluzzi, and G. P. R. Papini, "Modelling longitudinal vehicle dynamics with neural networks," *Vehicle Syst. Dyn.*, vol. 58, no. 11, pp. 1675–1693, 2019.

[36] D. Stocco and E. Bertolazzi, "ACME: A small 3D geometry library," *SoftwareX*, vol. 16, Dec. 2021, Art. no. 100845.

[37] D. Stocco, M. Larcher, and E. Bertolazzi, "A novel approach for real-time tire/ground contact modeling," *Multibody Syst. Dyn.*, to be published.

[38] J.-X. Xu and D. Huang, "Optimal tuning of PID parameters using iterative learning approach," in *Proc. IEEE 22nd Int. Symp. Intell. Control*, Mar. 2007, pp. 226–231.

[39] D. G. McClement, N. P. Lawrence, J. U. Backström, P. D. Loewen, M. G. Forbes, and R. B. Gopaluni, "Meta-reinforcement learning for the tuning of PI controllers: An offline approach," *J. Process Control*, vol. 118, pp. 139–152, Oct. 2022.

[40] G. G. Chrysos, S. Moschoglou, G. Bouritsas, J. Deng, Y. Panagakis, and S. Zafeiriou, "Deep polynomial neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 8, pp. 4021–4034, Aug. 2022.

[41] M. Guiggiani, *The Science of Vehicle Dynamics: Handling, Braking, and Ride of Road and Race Cars*. Berlin, Germany: Springer, 2018.

[42] 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 Syst. Dyn.*, vol. 31, no. 2, pp. 113–135, Feb. 1999.

[43] R. Lot and F. Biral, "A curvilinear abscissa approach for the lap time optimization of racing vehicles," *IFAC Proc. Volumes*, vol. 47, no. 3, pp. 7559–7565, 2014.

[44] F. Biral, E. Bertolazzi, and L. Da Mauro, "The optimal manoeuvre," in *Modelling, Simulation and Control of Two-Wheeled Vehicles*. Hoboken, NJ, USA: Wiley, 2014, ch. 5, pp. 119–154. [Online]. Available: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118536391.ch5, doi: 10.1002/9781118536391.ch5.

[45] E. Bertolazzi, F. Biral, and M. Da Lio, "Symbolic-numeric efficient solution of optimal control problems for multibody systems," *J. Comput. Appl. Math.*, vol. 185, no. 2, pp. 404–421, Jan. 2006.

[46] E. Bertolazzi, F. Biral, and M. D. Lio, "Real-time motion planning for multibody systems: Real life application examples," *Multibody Syst. Dyn.*, vol. 17, nos. 2–3, pp. 119–139, Apr. 2007.

[47] F. Biral, E. Bertolazzi, and P. Bosetti, "Notes on numerical methods for solving optimal control problems," *IEEJ J. Ind. Appl.*, vol. 5, no. 2, pp. 154–166, 2016.

[48] O. Nelles, *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*. Berlin, Germany: Springer, 2001.

[49] O. Nelles, A. Fink, and R. Isermann, "Local linear model trees (LOLIMOT) toolbox for nonlinear system identification," *IFAC Proc. Volumes*, vol. 33, no. 15, pp. 845–850, Jun. 2000.

[50] M. Da Lio and M. Piccinini, "Estimation of vehicle lateral velocity using neural networks with novel custom structure informed by kinematic principles," *IEEE Trans. Intell. Transp. Syst.*, to be published.

[51] E. Pagot, M. Piccinini, E. Bertolazzi, and F. Biral, "Real-time planning and tracking of complex parking maneuvers with an indirect optimal control approach," *IEEE Trans. Autom. Sci. Eng.*, to be published.

[52] M. Abe, *Vehicle Handling Dynamics: Theory and Application*. Amsterdam, The Netherlands: Elsevier, 2009.

[53] T. Dozat, "Incorporating Nesterov momentum into Adam," in *Proc. ICLR*, 2016, pp. 1–4.

[54] N. J. Killingsworth and M. Krstić, "PID tuning using extremum seeking: Online, model-free performance optimization," *IEEE Control Syst. Mag.*, vol. 26, no. 1, pp. 70–79, Feb. 2006.

[55] M. C. Campi, A. Lecchini, and S. M. Savaresi, "Virtual reference feedback tuning: A direct method for the design of feedback controllers," *Automatica*, vol. 38, no. 8, pp. 1337–1346, Aug. 2002.
[56] J.-X. Xu and Y. Tan, *Linear and Nonlinear Iterative Learning Control* (Lecture Notes in Control and Information Sciences), vol. 291. Springer-Verlag, Jan. 2003.
[57] J. C. Kegelman, L. K. Harbott, and J. C. Gerdes, "Insights into vehicle trajectories at the handling limits: Analysing open data from race car drivers," *Vehicle Syst. Dyn.*, vol. 55, no. 2, pp. 191–207, Feb. 2017.

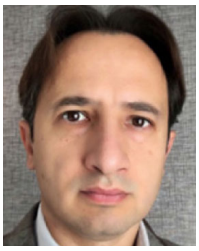
MATTIA PICCININI (Member, IEEE) received the B.Sc. degree (cum laude) in industrial engineering and the M.Sc. degree (cum laude) in mechatronics engineering from the University of Trento, Italy, in 2017 and 2019, respectively, where he is currently pursuing the Ph.D. degree. From March 2022 to June 2022, he was a Visiting Ph.D. Student with Universität der Bundeswehr München, Munich, Germany.

His research interests include motion planning, control, and state estimation methods for racing autonomous vehicles.

SEBASTIANO TADDEI received the bachelor's degree in industrial engineering from the University of Trento, in 2020, and the double master's degree in mechatronics engineering from the University of Trento and in ICT innovation from Aalto University, in 2022. He is currently pursuing the Ph.D. degree in the DAUSY National Ph.D. Program with the University of Trento coordinated by the Politecnico di Bari. His research interests include automated driving for both urban and racing applications, as well as vehicle dynamics and modeling.

MATTEO LARCHER received the master's degree in mechatronics engineering from the University of Trento, where he is currently pursuing the Ph.D. degree in vehicle dynamic models for real-time applications. His research interests include multibody modeling of mechanical systems and hardware-in-the-loop simulations.

MATTIA PIAZZA received the master's degree in mechatronics engineering from the University of Trento, Italy, where he is currently pursuing the Ph.D. degree. He is involved in autonomous driving and optimization for racing and urban scenarios. He received a research fellowship grant to work on minimum-time optimal control problems for racing vehicles.



FRANCESCO BIRAL received the master's degree in mechanical engineering from the University of Padova, Italy, and the Ph.D. degree in mechanism and machine theory from the University of Brescia, Italy, in 2000, with a focus on minimum lap time of racing vehicles with the use of optimal control.

He is currently an Associate Professor with the Department of Industrial Engineering, University of Trento. He has 15 years of experience in the development and validation of ADAS and AD functions, both for cars and PTWs, gained in several European and industrial funded research projects. His research interests include symbolic and numerical multi-body dynamics and optimization, constrained optimal control, mainly in the field of vehicle dynamics with special focus on intelligent vehicles and optimal maneuver for racing vehicles.

Open Access funding provided by 'Università degli Studi di Trento' within the CRUI CARE Agreement