

AD-MPCC: Adaptive Differentiable Model Predictive Contouring Control for Autonomous Racing

Nam T. Nguyen^{1,†}, Binh Nguyen^{1,†}, Ahmad Amine²,
Thanh Vo-Duy³, Rahul Mangharam², and Truong X. Nghiem¹

Abstract—This paper presents Adaptive Differentiable Model Predictive Contouring Control (AD-MPCC), a framework for autonomous racing that integrates differentiable MPCC with online parameter estimation to handle varying road-surface conditions. For online parameter estimation, we leverage a parameterized Pacejka Magic Formula together with a regularized moving-horizon estimation scheme with exponentially decaying weights to capture road interactions and update parameters in real time. Furthermore, we propose a differentiable MPCC (Diff-MPCC) framework that enables optimal adjustment of objective weights based on predefined long-horizon performance costs. To implement Diff-MPCC for online objective weight adaptation, we propose a Pacejka-informed machine learning model that is trained in a supervised manner using data generated by Diff-MPCC to tune the objective weights. Simulation results demonstrate that AD-MPCC reliably ensures safety and achieves faster lap times compared to baseline controllers in both single-surface and multiple-surface scenarios.

I. INTRODUCTION

Autonomous racing pushes vehicles to operate at their physical limits while optimizing well-defined performance objectives such as lap-time minimization [1], making it a compelling testbed for advanced control methods, including imitation learning [2], reinforcement learning [3], or iterative learning control [4]. Model predictive contouring control (MPCC) [5], [6] has emerged as a standard control framework for autonomous racing. By jointly minimizing contouring and lag errors while maximizing progress along a reference path, MPCC naturally captures the trade-off between tracking accuracy and racing speed.

A growing body of work has sought to improve MPCC by replacing or augmenting the physics-based predictive model with data-driven models. Gaussian process models have been integrated with nominal dynamics in [7], [8], and neural networks have been used to compensate for unmodeled effects in [9], with both approaches demonstrating improved lap times over their physics-based counterparts [10]. However, the nonlinearity and nonconvexity of the learned components increase the complexity of the predictive model, complicating real-time deployment. In practice, the dominant source of

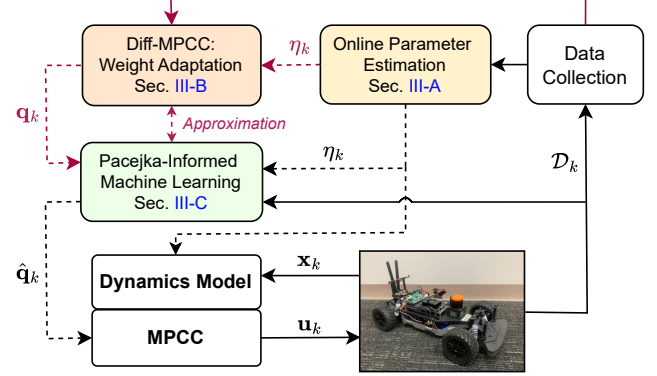


Fig. 1. The AD-MPCC architecture for autonomous racing. Purple lines and black lines represent offline training and real-time deployment, respectively. Dashed lines denote parameters and solid lines denote signals.

model uncertainty in racing arises from tire-road interactions, which are well characterized by the Pacejka tire model [11]. Rather than introducing learned residual models, directly estimating Pacejka parameters preserves simplicity and the interpretability of the vehicle dynamics based on physics. On multi-surface tracks, Pacejka parameters must be estimated in real time to ensure the predictive model’s accuracy.

In addition to online parameter estimation, another challenge in autonomous racing on multi-surface tracks is that changes in the dynamic model caused by updates of the Pacejka parameters require corresponding adaptation of the MPCC objective weights to maintain safety and consistent control performance. Differentiable MPC provides a principled mechanism for optimizing MPC objective weights by differentiating a performance cost through the control optimization problem, enabling gradient-based weight updates [12], [13]. Recent work has further enabled GPU-accelerated differentiable MPC solvers that scale to challenging driving tasks [14]. Extending this idea to MPCC, however, faces a fundamental difficulty. In standard MPC, the reference trajectory is fixed and the differentiation is straightforward. In MPCC, however, the reference point is determined implicitly through the optimized progress variable θ , coupling the objective weights to the reference trajectory and requiring higher-order derivatives of the reference path [5].

Addressing the above challenges, this paper presents the Adaptive Differentiable MPCC (AD-MPCC) framework that combines online Pacejka parameter estimation with a differentiable MPCC (Diff-MPCC) for autonomous racing under different road-surface conditions. Diff-MPCC provides gradients of the MPCC solution with respect to the objective

This material is based upon work supported by the National Science Foundation under Award No. 2514584.

[†]These authors contributed equally to this work.

¹Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL 32816, USA.

²Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104, USA.

³CTI Lab4EV, School of Electrical and Electronic Engineering, Hanoi University of Science and Technology, Hanoi 100000, Vietnam.

weights through the implicit optimality conditions of the underlying nonlinear program, enabling their optimization via a long-horizon performance cost. For online parameter estimation, we formulate a prior-regularized moving horizon estimation (MHE) problem with exponentially decaying weights that prioritizes recent observations to capture surface transitions. AD-MPCC integrates these two components to update the predictive model and tune the controller weights at each time step. Fig. 1 illustrates the AD-MPCC architecture.

Due to the computational burden of Diff-MPCC, we further propose a Pacejka-informed machine learning (PaIML) technique to efficiently approximate the optimal MPCC weight solutions obtained from Diff-MPCC. Specifically, PaIML reduces the input dimension of the online weight adaptation model from 14 to 5 by leveraging the Pacejka tire model and the vehicle's progress along the racetrack. The PaIML model is trained using data generated by Diff-MPCC, collected by running the vehicle on the racetrack under varying conditions. The trained model is then integrated into the AD-MPCC framework to infer near-optimal MPCC weights from the estimated vehicle dynamics in real time.

To highlight the performance of AD-MPCC, we validate the proposed framework in the F1TENTH-Gym environment [15] across both single- and multi-surface scenarios. The results show that AD-MPCC and Diff-MPCC significantly reduce lap times on a single-surface track, while AD-MPCC is the only controller capable of completing laps on an multi-surface road, thereby ensuring safe vehicle operation during racing in an uncertain environment.

The contributions of this work are highlighted as follows.

- We propose a new AD-MPCC framework that performs online estimation model and objective weight adaptation for autonomous racing.
- We provide a prior-regularized MHE scheme for online estimation that rapidly estimate Pacejka parameters in the dynamic model.
- We develop a differentiable programming technique for MPCC that enables gradient-based weight tuning to improve performance of the racing car. Also, a real-time version of Diff-MPCC is designed by proposing a Pacejka-informed ML technique.

Notation: Let $\mathbb{R}$, $\mathbb{N}$, $\mathbb{R}_{>0}$, and $\mathbb{N}_{>0}$ denote the set of real numbers, integer number, positive real numbers, and positive integer number, respectively. For vectors $\mathbf{v}_1, \dots, \mathbf{v}_n$, denote $\text{col}(\mathbf{v}_1, \dots, \mathbf{v}_n) = [\mathbf{v}_1^\top \dots \mathbf{v}_n^\top]^\top$. For vector $\mathbf{u} = [u_1 \dots u_n]^\top \in \mathbb{R}^n$, let $\text{diag}(\mathbf{u})$ be the diagonal matrix whose diagonal is $\mathbf{u}$. For differentiable functions $f(x) : \mathbb{R}^m \rightarrow \mathbb{R}^n$ and $g(z) : \mathbb{R}^n \rightarrow \mathbb{R}$, $\partial_x f$ is the Jacobian matrix of f and $\nabla_{zz}^2 g$ is the Hessian matrix of g .

II. PRELIMINARIES

A. Dynamic vehicle model

This paper considers the impact of track/road surface condition on the dynamics of the vehicle. These effects manifest primarily through lateral and longitudinal forces acting on the vehicle. Thus, we adopt a dynamic bicycle model [7],

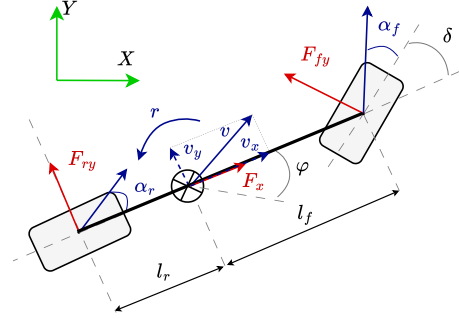


Fig. 2. Single-track dynamic bicycle model of a car.

which provides a compact representation of these effects and is depicted in Fig. 2 with the vehicle states and acting forces shown. Here, $\mathbf{x} = \text{col}(X, Y, \varphi, v_x, v_y, r, \delta)$ denotes the state of the car, where X and Y are the positions in the global Cartesian coordinate frame, φ represents the heading angle, v_x, v_y, r , and δ stand for the longitudinal velocity, the lateral velocity, the yaw rate, and the steering angle, respectively. The control input vector is $\mathbf{u} = [\Delta_\delta \ F_a]^\top$, where F_a is the motor driving force and Δ_δ is the steering angle speed.

The dynamic bicycle model is given by

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} v_x \cos(\varphi) - v_y \sin(\varphi) \\ v_x \sin(\varphi) + v_y \cos(\varphi) \\ r \\ \frac{1}{m}(F_x - F_{f,y} \sin(\delta) + m v_y r) \\ \frac{1}{m}(F_{r,y} + F_{f,y} \cos(\delta) - m v_x r) \\ \frac{1}{I_z}(F_{f,y} l_f \cos(\delta) - F_{r,y} l_r) \\ \Delta_\delta \end{bmatrix}, \quad (1)$$

where l_r and l_f are the distances from the center of gravity (CoG) of the car to the rear and front wheels, I_z is the yaw inertia of the car, and m is the mass of the car. The lateral forces for the front $F_{f,y}$ and rear $F_{r,y}$ wheels are given by the Pacejka model [11]

$$\mu_r = D_r \sin(C_r \arctan(B_r \alpha_r)), \quad (2)$$

$$\mu_f = D_f \sin(C_f \arctan(B_f \alpha_f)), \quad (3)$$

$$\alpha_r = \arctan\left(\frac{v_y - l_r r}{v_x}\right), \quad \alpha_f = \arctan\left(\frac{v_y + l_f r}{v_x}\right) - \delta,$$

$$F_{r,y} = \frac{l_f}{l_r + l_f} m g \mu_r, \quad F_{f,y} = \frac{l_r}{l_r + l_f} m g \mu_f. \quad (4)$$

where D_r, C_r, B_r, D_f, C_f , and B_f are positive scalar parameters, and g is the gravitational constant. Here, μ_r and μ_f represent the friction coefficients between the rear tire and the front tire to the road surface, with $\mu_r, \mu_f \in [-\mu_{\max}, \mu_{\max}]$, where $\mu_{\max}$ is the maximum value of the Pacejka line with respect to the type of road [11]. Additionally, the forces $F_{f,y}$ and $F_{r,y}$ are formulated with the assumption that there are no lateral acceleration effects. α_f and α_r are the side-slip angles of the front and rear wheels, respectively.

The longitudinal force F_x is generated by the rear-wheel drivetrain and provides the acceleration thrust for the car [7]. It is approximated by a linear combination of the motor driving force and resistances

$$F_x = C_a F_a - C_{r0} - C_{r2} v_x^2, \quad (5)$$

where $C_a F_a$ is a simple drivetrain, C_a is the motor con-

stant, C_{r0} is the rolling resistance, and $C_{r2}v_x^2$ is the drag resistance. The rolling resistance coefficient C_{r0} primarily captures mechanical and drivetrain losses, while the drag coefficient C_{r2} depends on vehicle geometry and aerodynamic properties. Hence, both are treated as constant parameters and adopted from [6]. In contrast, road surface variations are primarily captured by the Pacejka parameters (B, C, D), which vary with the road surface condition [11]. We denote the vector of road surface parameters as $\boldsymbol{\eta} = [D_r \ C_r \ B_r \ D_f \ C_f \ B_f \ C_a]^\top$. The dynamic vehicle model (1) can then be written as depending on $\boldsymbol{\eta}$ as $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}; \boldsymbol{\eta})$, which is discretized with a constant time step $\Delta t > 0$ to obtain the following discrete-time dynamics to be used hereafter

$$\mathbf{x}_{k+1} = \mathbf{x}_k + g(\mathbf{x}_k, \mathbf{u}_k; \boldsymbol{\eta}), \quad (6)$$

where $g(\mathbf{x}_k, \mathbf{u}_k; \boldsymbol{\eta}) = \int_0^{\Delta t} f(\mathbf{x}, \mathbf{u}; \boldsymbol{\eta}) dt$, k is the time-step index, t_k is the corresponding continuous time, $\mathbf{x}_k = \mathbf{x}(t_k)$, and $\mathbf{u}_k = \mathbf{u}(t_k)$. Usually, the parameter $\boldsymbol{\eta}$ of the predictive model is assumed fixed [6], [9]. However, in practice, due to uncertainties and varying road conditions, $\boldsymbol{\eta}$ must be updated to accurately reflect the vehicle's dynamics.

B. Model predictive contouring control

This paper follows the MPCC formulation presented in [6], [7]. The reference path of an MPCC controller is parameterized by a progress variable $\theta \in [0, \theta_{\max}]$, representing a parametric curve in the global X–Y frame, with coordinates given by the functions $X_c(\theta)$ and $Y_c(\theta)$. In this paper, we assume that $X_c(\theta)$ and $Y_c(\theta)$ describe the racetrack centerline, obtained via interpolation between successive track waypoints. The vehicle completes the track as θ increases from 0 to $\theta_{\max}$. At the current position $[X_k, Y_k]$, θ_k is determined as

$$\theta_k = \arg \min_{\theta \in [0, \theta_{\max}]} \|X_k - X_c(\theta)\|_2^2 + \|Y_k - Y_c(\theta)\|_2^2. \quad (7)$$

The lag error e_l and contour error e_c are defined based on the position (X, Y) and progress θ as

$$e_c(\mathbf{x}, \theta) = \sin(\Phi(\theta))(X - X_c(\theta)) - \cos(\Phi(\theta))(Y - Y_c(\theta)),$$

$$e_l(\mathbf{x}, \theta) = -\cos(\Phi(\theta))(X - X_c(\theta)) - \sin(\Phi(\theta))(Y - Y_c(\theta)),$$

where $\Phi(\theta) = \arctan(\nabla Y_c(\theta)/\nabla X_c(\theta))$ is the tangent angle of the reference path, and $X_c(\theta_k)$ and $Y_c(\theta_k)$ denote the position of the center line at θ_k obtained from (7). As in [7], a virtual control input ϑ is introduced as the progress rate of θ along the track. The MPCC optimization problem is then formulated as

$$\begin{aligned} \min J(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}, \boldsymbol{\vartheta}; \mathbf{q}) \\ \text{s.t. } \mathbf{x}_{0|k} = \mathbf{x}_k, \end{aligned} \quad (8a)$$

$$\mathbf{x}_{i+1|k} = \mathbf{x}_{i|k} + g(\mathbf{x}_{i|k}, \mathbf{u}_{i|k}; \boldsymbol{\eta}), \quad (8b)$$

$$\theta_{i+1|k} = \theta_{i|k} + \vartheta_{i|k}, \quad (8c)$$

$$A_x \mathbf{x}_{i|k} \preceq \mathbf{b}_x, \quad (8d)$$

$$A_u \mathbf{u}_{i|k} \preceq \mathbf{b}_u, \quad (8e)$$

$$0 \leq \theta_{i|k} \leq \theta_{\max}, \quad (8f)$$

where T_h is the control horizon, $\mathbf{x} = \text{col}(\mathbf{x}_{1|k}, \dots, \mathbf{x}_{T_h|k})$, $\mathbf{u} = \text{col}(\mathbf{u}_{0|k}, \dots, \mathbf{u}_{T_h-1|k})$, $\boldsymbol{\theta} = [\theta_{1|k} \ \dots \ \theta_{T_h|k}]^\top$, $\boldsymbol{\vartheta} =$

$[\vartheta_{1|k} \ \dots \ \vartheta_{T_h|k}]^\top$ with $\theta_{0|k} = \theta_k$ obtained from (7), (8d), and (8e) are the state and control constraints, respectively. The objective function of MPCC is formulated to increase the virtual control input $\boldsymbol{\vartheta}$, which drives the racing progress forward, while reducing the contouring and lag errors as

$$\begin{aligned} J(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}, \boldsymbol{\vartheta}; \mathbf{q}) = \\ \sum_{i=0}^{T_h-1} (q_c e_c^2(\mathbf{x}_{i|k}, \theta_{i|k}) + q_l e_l^2(\mathbf{x}_{i|k}, \theta_{i|k}) - q_v \vartheta_{i|k} + \mathbf{u}_{i|k}^\top R \mathbf{u}_{i|k}), \end{aligned}$$

in which $\mathbf{q} = [q_c \ q_l \ q_v]^\top$, $q_c, q_l, q_v \in \mathbb{R}_{>0}$ are the MPCC objective weights, and $R = R^\top \in \mathbb{R}^{2 \times 2}$ is a positive definite matrix.

The first element of the optimal input sequence, $\mathbf{u}_{0|k}$, is applied to the vehicle at time step k . For simplicity, we denote $[\mathbf{u}_k \ \vartheta_k]^\top = [\mathbf{u}_{0|k} \ \vartheta_{0|k}]^\top = \pi_q(\mathbf{x}_k; \mathbf{q}, \boldsymbol{\eta})$ where $[\mathbf{u}_k \ \vartheta_k]^\top \in \mathbb{R}^3$.

Our goal is to develop a real-time MPCC-based control framework for autonomous racing cars that can improve racing performance compared to conventional MPCC while maintaining safety under varying road-surface conditions, through online adaptation of the Pacejka parameters and automatic tuning of the MPCC objective weights.

Towards this goal, we present in the next section our framework: Adaptive Differentiable MPCC (AD-MPCC).

III. ADAPTIVE DIFFERENTIABLE MPCC

The AD-MPCC framework comprises two main components, illustrated in Fig. 1 and described in this section. The first component updates the varying Pacejka parameters online, based on the MHE approach. The second component optimally adapts the MPCC objective weights in real time by leveraging advanced differentiable programming.

A. Online Pacejka parameter estimation

This paper employs MHE [16] to estimate $\boldsymbol{\eta}$ from the rolling dataset $\mathcal{D}_k = \{\mathbf{x}_{i+1}, \mathbf{u}_i\}_{i=k-T, \dots, k-1}$, where k is the current time step and $T \in \mathbb{N}_{>0}$ is the window horizon ($k > T$). On multi-surface tracks, the dynamic model must adapt quickly to changing tire-road characteristics by prioritizing recent data over older observations. Hence, we introduce exponentially decaying weights into the MHE objective. Additionally, we regularize toward nominal Pacejka parameters $\boldsymbol{\eta}_{gt}$ obtained from the regular road of the racetrack. Consequently, at time step k , we estimate the parameters $\boldsymbol{\eta}$ from the dataset $\mathcal{D}_k$ via the optimization problem

$$\begin{aligned} \boldsymbol{\eta}_k = \arg \min_{\boldsymbol{\eta} \in \Gamma} \sum_{\tau=1}^{T-1} c^\tau \|\Delta \mathbf{x}_{k-\tau} - g(\mathbf{x}_{k-\tau}, \mathbf{u}_{k-\tau}; \boldsymbol{\eta})\|_2^2 \\ + \frac{\sigma}{2} \|\boldsymbol{\eta} - \boldsymbol{\eta}_{gt}\|_2^2, \end{aligned} \quad (9)$$

where $\hat{\mathbf{x}}_{k-\tau}$ is the predicted value of $\mathbf{x}_{k-\tau}$, Γ is the feasible set for $\boldsymbol{\eta}$, c is the exponential decay factor ($c < 1$) and $\Delta \mathbf{x}_{k-\tau} = \mathbf{x}_{k-\tau+1} - \mathbf{x}_{k-\tau}$. The optimization problem (9) has a differentiable cost and can be efficiently solved via first-order optimization methods [17]. The parameter $\boldsymbol{\eta}$ is updated at each time step. We denote by $\boldsymbol{\eta}_k$ and $\mathbf{q}_k$ the updated value of $\boldsymbol{\eta}$ and the value of $\mathbf{q}$ at time step k .

B. Differentiable MPCC (Diff-MPCC)

This section develops a weight adaptation mechanism using differentiable programming to determine $\mathbf{q}_k$ at each time step. Differentiable MPC [12] has been shown to improve control performance by optimizing the MPC weights. However, extending differentiable programming to MPCC is nontrivial. In particular, MPCC involves three coupled objective weights, q_c , q_l , and q_v , each associated with a distinct control objective. Also, the reference path $(X_c(\theta), Y_c(\theta))$ depends on the optimized progress variable θ .

To evaluate the performance of π_q over a long horizon $H \gg T_h$, let us consider the performance optimization problem in the following form

$$\min_{\mathbf{q}} \ell(\mathbf{q}; \mathbf{x}_k, \boldsymbol{\eta}_k) = \sum_{i=0}^{H-1} l_i(\bar{\mathbf{x}}_{i+1|k}, \bar{\mathbf{u}}_{i|k}, \bar{\vartheta}_{i|k}), \quad (10a)$$

$$\text{s.t. } \bar{\mathbf{x}}_{i+1|k} = \bar{\mathbf{x}}_{i|k} + g(\bar{\mathbf{x}}_{i|k}, \bar{\mathbf{u}}_{i|k}; \boldsymbol{\eta}_k), \quad \bar{\mathbf{x}}_{0|k} = \mathbf{x}_k, \quad (10b)$$

$$\begin{bmatrix} \bar{\mathbf{u}}_{i|k} & \bar{\vartheta}_{i|k} \end{bmatrix}^\top = \pi_q(\bar{\mathbf{x}}_{i|k}; \mathbf{q}, \boldsymbol{\eta}_k), \quad (10c)$$

where ℓ is the performance cost function, l_i is the twice differentiable convex stage cost function, and $\mathbf{x}_k$ and $\boldsymbol{\eta}_k$ denote the current state and estimated parameters at time step k . In (10), we optimize the MPCC weights $\mathbf{q}$ at time step k by minimizing the performance cost ℓ , subject to the system dynamics and the MPCC feedback policy. Here, $\bar{\mathbf{x}}_{i|k}$, $\bar{\mathbf{u}}_{i|k}$, and $\bar{\vartheta}_{i|k}$ denote the predicted state, input, and virtual input associated with (8), respectively. Since the prediction horizon T_h of MPCC is limited by computational resources, the cost ℓ represents the closed-loop performance of the MPCC policy over a longer horizon H .

We assume that $\mathbf{q}$ belongs to the set $\mathcal{Q}$ of admissible parameters, for example, $q_c, q_l, q_v > 0$. We also assume that the MPCC policy (10c) is feasible for all $\mathbf{q} \in \mathcal{Q}$. Note that the constrained optimization problem (10) can be rewritten as an unconstrained optimization problem by substituting (10b) and (10c) into (10a). In addition, if the gradient $\frac{\partial \ell}{\partial \mathbf{q}}$ can be computed, $\mathbf{q}_k$ can be optimized using the following projected gradient method

$$\mathbf{q}_k^{(j+1)} = \text{Proj}_{\mathcal{Q}} \left(\mathbf{q}_k^{(j)} - \gamma^{(j)} \frac{\partial \ell}{\partial \mathbf{q}} \right), \quad (11)$$

where $\text{Proj}_{\mathcal{Q}}(\mathbf{q})$ is the projection of $\mathbf{q}$ onto $\mathcal{Q}$, j is the iteration index of the gradient-based update, and $\gamma^{(j)} > 0$ is the learning rate. By the chain rule, the gradient of ℓ with respect to $\mathbf{q}$ can be computed as

$$\frac{\partial \ell}{\partial \mathbf{q}} = \sum_{i=0}^{H-1} \frac{\partial \ell}{\partial \bar{\mathbf{x}}_{i+1|k}} \frac{\partial \bar{\mathbf{x}}_{i+1|k}}{\partial \mathbf{q}} + \frac{\partial \ell}{\partial \bar{\mathbf{u}}_{i|k}} \frac{\partial \bar{\mathbf{u}}_{i|k}}{\partial \mathbf{q}} + \frac{\partial \ell}{\partial \bar{\vartheta}_{i|k}} \frac{\partial \bar{\vartheta}_{i|k}}{\partial \mathbf{q}}, \quad (12a)$$

$$\frac{\partial \bar{\mathbf{x}}_{i+1|k}}{\partial \mathbf{q}} = \frac{\partial \bar{\mathbf{x}}_{i|k}}{\partial \mathbf{q}} + \frac{\partial g}{\partial \bar{\mathbf{x}}_{i|k}} \frac{\partial \bar{\mathbf{x}}_{i|k}}{\partial \mathbf{q}} + \frac{\partial g}{\partial \bar{\mathbf{u}}_{i|k}} \frac{\partial \bar{\mathbf{u}}_{i|k}}{\partial \mathbf{q}}, \quad (12b)$$

$$\begin{bmatrix} \frac{\partial \bar{\mathbf{u}}_{i|k}}{\partial \mathbf{q}} \\ \frac{\partial \bar{\vartheta}_{i|k}}{\partial \mathbf{q}} \end{bmatrix} = \frac{\partial \pi_q}{\partial \mathbf{q}}(\bar{\mathbf{x}}_{i|k}; \mathbf{q}_k^{(j)}, \boldsymbol{\eta}_k). \quad (12c)$$

Computing $\frac{\partial \ell}{\partial \mathbf{q}}$ via (12) requires H iterations from 0 to $H-1$, starting from $\frac{\partial \bar{\mathbf{u}}_{0|k}}{\partial \mathbf{q}}$ and $\frac{\partial \bar{\vartheta}_{0|k}}{\partial \mathbf{q}}$ to $\frac{\partial \bar{\mathbf{u}}_{H-1|k}}{\partial \mathbf{q}}$ and $\frac{\partial \bar{\vartheta}_{H-1|k}}{\partial \mathbf{q}}$.

The main difficulty lies in computing $\frac{\partial \pi_q}{\partial \mathbf{q}}$, since there is no explicit mapping from $\mathbf{q}_k$ to the solution of MPCC. In each iteration of the gradient-based update (11), the MPCC optimization problem (8) must be solved H times to compute $\frac{\partial \ell}{\partial \mathbf{q}}$ when $\mathbf{q}_k^{(j)}$ is updated. Indeed, to compute $\frac{\partial \pi_q}{\partial \mathbf{q}}(\bar{\mathbf{x}}_{0|k}; \mathbf{q}_k^{(j)}, \boldsymbol{\eta}_k)$, (8) must be solved at $(\bar{\mathbf{x}}_{0|k}, \mathbf{q}_k^{(j)}, \boldsymbol{\eta}_k)$ to obtain $\bar{\mathbf{u}}_{0|k}$ and $\bar{\vartheta}_{0|k}$. Then, by (10b), we can compute $\bar{\mathbf{x}}_{1|k}$. Repeatedly applying this procedure, we obtain $\{\bar{\mathbf{x}}_{i|k}\}_{i=1,\dots,H}$, $\{\bar{\mathbf{u}}_{i|k}\}_{i=0,\dots,H-1}$, and $\{\bar{\vartheta}_{i|k}\}_{i=0,\dots,H-1}$.

Let $h_{\text{eq}}(\mathbf{z}; \boldsymbol{\eta}_k) = 0$ denote the equality constraints (8a)–(8c), where $\mathbf{z} = \text{col}(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}, \boldsymbol{\vartheta})$. The inequality constraints (8d)–(8f) are denoted by $h_{\text{in}}(\mathbf{z}) \preceq 0$. A common approach to compute $\frac{\partial \pi_q}{\partial \mathbf{q}}$ is sequential quadratic programming (SQP) [14], which approximates the nonconvex optimization problem (8) by a quadratic program through linearization of the dynamics (8b) and quadratic approximation of the cost function $J(\mathbf{z}; \mathbf{q}) = J(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}, \boldsymbol{\vartheta}; \mathbf{q})$. Although SQP can accelerate the computation of $\frac{\partial \pi_q}{\partial \mathbf{q}}$, the approximation error can accumulate over the long horizon H , potentially violating the dynamics constraint (10b) and can lead to unsafe control actions. Moreover, the existence of $\frac{\partial \pi_q}{\partial \mathbf{q}}$ cannot be guaranteed when the parameter $\mathbf{q}_k$ is updated according to (11), particularly when the cost function $J(\mathbf{z}; \mathbf{q})$ is non-convex and the optimization problem may admit multiple local minima.

Instead, we approximately compute $\frac{\partial \pi_q}{\partial \mathbf{q}}$ using nonlinear programming methods by applying the implicit function theorem (IFT) [18] to the optimality conditions. Specifically, consider the following optimization problem at the j -th iteration of the policy update

$$(\mathcal{P}_j) : \min_{\mathbf{z}} \bar{J}^{(j)}(\mathbf{z}; \mathbf{q}_k^{(j)}, \boldsymbol{\eta}_k) \text{ s.t. } h_{\text{eq}}(\mathbf{z}; \boldsymbol{\eta}_k) = 0, \quad (13)$$

where

$$\bar{J}^{(j)}(\mathbf{z}; \mathbf{q}_k^{(j)}, \boldsymbol{\eta}_k) = J(\mathbf{z}; \mathbf{q}_k^{(j)}) - \epsilon_1 \sum_i^{n_h} \log(-h_{\text{in},i}(\mathbf{z})) + \frac{\epsilon_2}{2} \|\mathbf{z} - \mathbf{z}^{(j-1)}\|_2^2,$$

in which $h_{\text{in},i}$ are log barrier functions, n_h is the number of inequalities, $\epsilon_1 > 0$ is the barrier parameter, $\epsilon_2 > 0$, and $\mathbf{z}^{(j-1)} = \text{col}(\mathbf{x}^{(j-1)}, \mathbf{u}^{(j-1)}, \boldsymbol{\theta}^{(j-1)}, \boldsymbol{\vartheta}^{(j-1)})$ is a (local) minimum of $\mathcal{P}_{j-1}$. Let us define $h_{\text{in}}(\mathbf{z}) = [h_{\text{in},1}(\mathbf{z}) \ h_{\text{in},2}(\mathbf{z}) \ \dots \ h_{\text{in},n_h}(\mathbf{z})]^\top$. Note that the solutions of $\mathcal{P}_j$ approach those of (8) as $\epsilon_1, \epsilon_2 \rightarrow 0$ when $\mathbf{q} = \mathbf{q}_k^{(j)}$, $\boldsymbol{\eta} = \boldsymbol{\eta}_k$. Thus, by choosing small enough ϵ_1 and ϵ_2 , instead of computing $\frac{\partial \pi_q}{\partial \mathbf{q}}(\bar{\mathbf{x}}_{i|k}; \mathbf{q}_k^{(j)}, \boldsymbol{\eta}_k)$, we can approximate it by $\frac{\partial \pi_q}{\partial \mathbf{q}}(\bar{\mathbf{x}}_{i|k}; \mathbf{q}_k^{(j)}, \boldsymbol{\eta}_k)$ where $\bar{\pi}_q(\bar{\mathbf{x}}_{i|k}; \mathbf{q}_k^{(j)}, \boldsymbol{\eta}_k)$ is the approximated MPCC controller by $\mathcal{P}_j$.

Solving problem $\mathcal{P}_j$ is with using $\mathbf{z}^{(j-1)}$, obtained from $\mathcal{P}_{j-1}$, as the warm start. The Lagrangian of (13) is $L(\mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\beta}; \mathbf{q}_k^{(j)}, \boldsymbol{\eta}_k) = \bar{J}^{(j)}(\mathbf{z}; \mathbf{q}_k^{(j)}, \boldsymbol{\eta}_k) + \boldsymbol{\lambda}^\top h_{\text{eq}}(\mathbf{z}; \boldsymbol{\eta}_k) + \boldsymbol{\beta}^\top h_{\text{in}}(\mathbf{z})$, and the first order optimality conditions of (13)

Algorithm 1 Gradient-based MPCC weights update

Require: Set of MPCC weights $\mathcal{Q}$, state $\mathbf{x}_k$ and parameter $\boldsymbol{\eta}_k$ at step k , loss performance ℓ , initial weights $\mathbf{q}_k^{(0)}$, and number of iteration n_{iter} .

Ensure: MPCC weights $\mathbf{q}_k^{(j)} \in \mathcal{Q}$

```

for  $j = 1, 2, \dots, n_{\text{iter}}$  do
  Set  $\bar{\mathbf{x}}_{0|k} = \mathbf{x}_k$ 
  for  $i = 0, 1, \dots, H - 1$  do
    Compute  $\mathbf{x}^{(j)}, \mathbf{u}^{(j)}, \boldsymbol{\theta}^{(j)}, \boldsymbol{\vartheta}^{(j)}$  in (13)
    Set  $\bar{\mathbf{u}}_{i|k} = \mathbf{u}_{0|k}^{(j)}$  and  $\bar{\boldsymbol{\vartheta}}_{i|k} = \boldsymbol{\vartheta}_{0|k}^{(j)}$ 
    Compute  $\bar{\mathbf{x}}_{i+1|k}$  from  $\bar{\mathbf{x}}_{i|k}$  and  $\bar{\mathbf{u}}_{i|k}$  in (10b)
    Compute  $\frac{\partial \boldsymbol{\kappa}}{\partial \mathbf{w}}$  at  $\mathbf{x}^{(j)}, \mathbf{u}^{(j)}, \boldsymbol{\theta}^{(j)}, \boldsymbol{\vartheta}^{(j)}$ 
    Compute  $\frac{\partial \bar{\mathbf{u}}_{i|k}}{\partial \mathbf{q}}$  and  $\frac{\partial \bar{\boldsymbol{\vartheta}}_{i|k}}{\partial \mathbf{q}}$  in (12c) by (15)
    Compute  $\frac{\partial \bar{\mathbf{x}}_{i+1|k}}{\partial \mathbf{q}}$  in (12b)
  end for
  Compute  $\frac{\partial \ell}{\partial \mathbf{q}}$  in (12a)
  Update MPCC weights  $\mathbf{q}_k^{(j+1)}$  in (11)
end for
  
```

are given by

$$0 = \nabla_{\mathbf{z}} L(\mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\beta}; \mathbf{q}_k^{(j)}, \boldsymbol{\eta}_k), \quad (14a)$$

$$0 = h_{\text{eq}}(\mathbf{z}; \boldsymbol{\eta}_k), \quad (14b)$$

$$0 = \text{diag}(\boldsymbol{\beta}) h_{\text{in}}(\mathbf{z}; \boldsymbol{\eta}_k) + \epsilon_1 [1 \ \dots \ 1]^\top, \quad (14c)$$

$$0 \prec \boldsymbol{\beta}. \quad (14d)$$

By the IFM, letting $\mathbf{w} = \text{col}(\mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\beta})$, we can compute

$$\frac{\partial \bar{\pi}_{\mathbf{q}}}{\partial \mathbf{q}}^\top = \frac{\partial \mathbf{w}}{\partial \mathbf{q}}^\top \frac{\partial \bar{\pi}_{\mathbf{q}}}{\partial \mathbf{w}}^\top \stackrel{\text{IFM}}{=} - \frac{\partial \boldsymbol{\kappa}}{\partial \mathbf{q}}^\top \underbrace{\left(\frac{\partial \boldsymbol{\kappa}}{\partial \mathbf{w}}^\top \right)^{-1}}_{\mathbf{v}} C, \quad (15)$$

where $\frac{\partial \bar{\pi}_{\mathbf{q}}}{\partial \mathbf{w}} = C^\top = \begin{bmatrix} 1 & 0 & \dots & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & \dots & 0 \\ 0 & 0 & \dots & 1 & \dots & 0 \end{bmatrix}$, $\boldsymbol{\kappa}$ is the vector function whose elements are gathered from the right-hand side of (14a)-(14c), and

$$\frac{\partial \boldsymbol{\kappa}}{\partial \mathbf{w}} = \begin{bmatrix} \nabla_{\mathbf{z}\mathbf{z}}^2 L & \partial_{\mathbf{z}}^\top h_{\text{eq}}(\mathbf{z}^{(j)}; \boldsymbol{\eta}_k) & \partial_{\mathbf{z}}^\top h_{\text{in}}(\mathbf{z}^{(j)}) \\ \partial_{\mathbf{z}} h_{\text{eq}}(\mathbf{z}^{(j)}; \boldsymbol{\eta}_k) & 0 & 0 \\ \text{diag}(\boldsymbol{\beta}) \partial_{\mathbf{z}} h_{\text{in}}(\mathbf{z}^{(j)}) & 0 & \text{diag}(h_{\text{in}}(\mathbf{z}^{(j)})) \end{bmatrix}.$$

In (15), $\mathbf{v}$ is computed effectively by exploiting the sparsity of $\frac{\partial \boldsymbol{\kappa}}{\partial \mathbf{w}}$ to solve $\frac{\partial \boldsymbol{\kappa}}{\partial \mathbf{w}}^\top \mathbf{v} = C$. Algorithm 1 summarizes the gradient-based weights update for given $\boldsymbol{\eta}_k$ and $\mathbf{x}_k$.

Remark 1: Several methods exist to compute $\frac{\partial \pi_{\mathbf{q}}}{\partial \mathbf{q}}$, including differentiable nonlinear MPC [19], [20], linearized MPC [14], and JAXOPT [21]. In the proposed differentiable MPCC (Diff-MPCC) framework, the nonsingularity of the Hessian matrix $\nabla_{\mathbf{z}\mathbf{z}}^2 L$ is enforced by incorporating convex terms into (13), namely barrier and regularization functions. Furthermore, since $h_{\text{in}}(\mathbf{z}) < 0$, the KKT matrix $\frac{\partial \boldsymbol{\kappa}}{\partial \mathbf{w}}$ is invertible if the Jacobian $\partial_{\mathbf{z}} h_{\text{eq}}(\mathbf{z}; \boldsymbol{\eta}_k)$ has full row rank and the Hessian matrix $\nabla_{\mathbf{z}\mathbf{z}}^2 L$ is nonsingular.

Remark 2: Unlike differentiable MPC, differentiable MPCC requires higher-order derivatives of the reference path, namely $\partial_{\theta} X_c(\theta)$, $\partial_{\theta} Y_c(\theta)$, $\nabla_{\theta\theta}^2 X_c(\theta)$, and $\nabla_{\theta\theta}^2 Y_c(\theta)$, as they explicitly appear in $\nabla_{\mathbf{z}\mathbf{z}}^2 L$. Hence, the reference

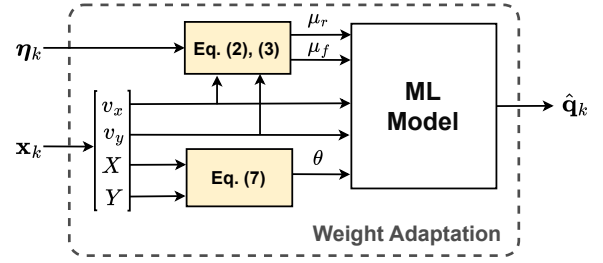


Fig. 3. Diagram of Pacejka-informed machine learning (PaIML) model for MPCC weights online adaptation.

path $(X_c(\theta), Y_c(\theta))$ must be at least twice continuously differentiable to ensure well-defined and continuous second-order derivatives.

C. Online adaptation of MPCC weights

At each time step k , changes in the Pacejka parameters require re-running Algorithm 1 to compute new optimal MPCC weights $\mathbf{q}$, which may exceed real-time computation budgets. To address this issue, we approximate the mapping from $(\mathbf{x}_k, \boldsymbol{\eta}_k)$ to the optimal weights via a supervised learning model trained offline.

The training data is generated using a racing vehicle simulated under various road-surface conditions. The dataset consists of state vectors $\mathbf{x}_k^l$ and corresponding tire parameter vectors $\boldsymbol{\eta}_k^l$, where k denotes the time index and l denotes the road-surface condition. For each road type l , the constant Pacejka parameters $\boldsymbol{\eta}^l$ are computed using the semi-empirical formulation described in [11, Chapter 4]. Given $\boldsymbol{\eta}^l$, a standard MPCC controller with the dynamic model parameterized by $\boldsymbol{\eta}^l$ is employed to generate the training data. We define the dataset $\mathcal{S} = \{ \{ (\mathbf{x}_k^l, \boldsymbol{\eta}_k^l), \mathbf{q}_k^l \} \}_{k=0, \dots, N_l} \}_{l=1, \dots, L}$, where N_l is the number of data points collected on road l , and L is the total number of considered roads. The label $\mathbf{q}_k^l$ is obtained by Algorithm 1 with given $(\mathbf{x}_k^l, \boldsymbol{\eta}_k^l)$.

In practice, obtaining exact tire parameters in real-world environments is generally difficult due to modeling uncertainties, and constructing test conditions with different road surfaces for data collection can be costly. Hence, data can be collected using high-fidelity simulators, such as F1TENTH-Gym [15] for autonomous racing or CARLA [22] for general autonomous driving, to generate representative datasets prior to real-world deployment.

At each time step k , using the full input $[\mathbf{x}_k \ \boldsymbol{\eta}_k]^\top$ results in a 14-dimensional feature vector, which increases inference time and makes the ML model more complex. To deal with this issue, we propose Pacejka-informed machine learning (PaIML) in Fig. 3, which leverages the physical structure of the Pacejka's magic formula together with an ML model to infer the optimal weights from lower-dimensional inputs. PaIML is also considered as a physics-informed ML technique [23], [24]. In particular, $\boldsymbol{\eta}$ mainly influences the lateral tire forces through the friction coefficients μ_r and μ_f , which are computed from $\boldsymbol{\eta}$ via (2) and (3). Thus, we replace the input $\boldsymbol{\eta}$ with the reduced representation consisting of μ_r and μ_f for the inference model. Moreover, the vehicle position (X, Y) encodes track curvature through

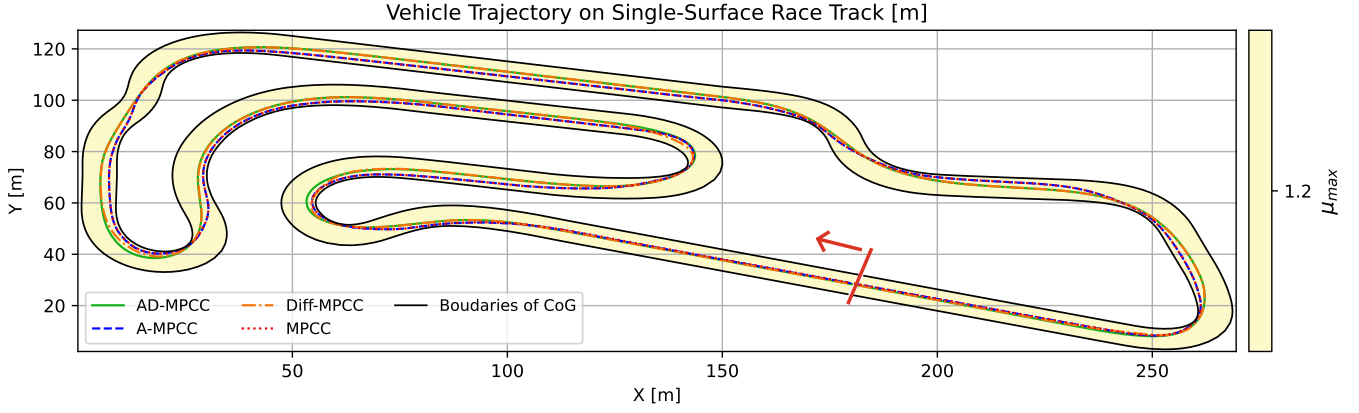


Fig. 4. The trajectories of the four MPCC controllers during the ninth lap. The road surface is uniform with $\mu_{\max} = 1.2$. All methods complete 10 laps.

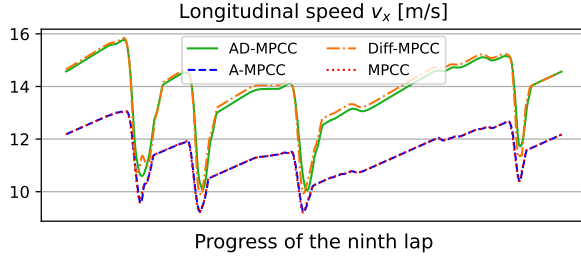


Fig. 5. Longitudinal velocity during the ninth lap in the single-surface scenario as the vehicle completes the racetrack from left to right.

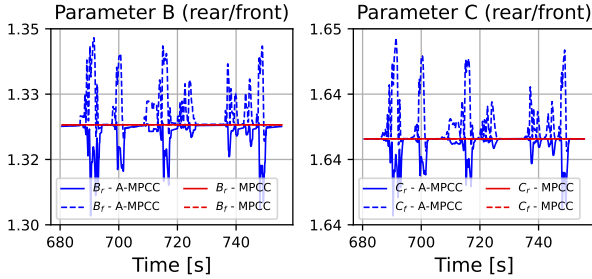


Fig. 6. Online-estimated and fixed Pacejka parameters (results for sample B and C only) of the ninth lap in single-surface road scenario.

the progress variable θ obtained from (7). Together with the longitudinal and lateral velocities v_x and v_y , these variables capture the dominant information describing the vehicle state. This results in the 5-dimensional input vector $[\mu_r \ \mu_f \ v_x \ v_y \ \theta]^\top$.

Consequently, PaIML learns the mapping from the five inputs to the optimal MPCC weights $\hat{\mathbf{q}}$, as illustrated in Fig. 3. Since the regression task is low-dimensional (five inputs and three outputs) and a large dataset can be generated offline, we employ XGBoost [25] for the ML component of PaIML. Tree-based ensemble methods are well suited for structured low-dimensional data and provide built-in regularization, resulting in robust performance without extensive hyperparameter tuning.

IV. SIMULATION RESULTS

A. Scenario setup

To evaluate the proposed AD-MPCC, we compare it against three different MPCC schemes:

- **MPCC**: A baseline controller that employs a time-invariant predictive model with fixed Pacejka param-

TABLE I
RESULTS OF SINGLE-SURFACE CASE FOR 10 LAPS.

	MPCC [6]	A-MPCC	Diff-MPCC	AD-MPCC
Avg. lap Time [s]	75.57	75.62	64.08	64.89
Avg. v_x [m s^{-1}]	11.383	11.41	13.591	13.501
Avg. lateral offset [m]	1.953	1.815	1.040	0.926
Avg. comp. time [ms]	20.45	21.21	23.56	24.18

eters $\eta = \eta_{gt}$, where η_{gt} is obtained from the semi-empirical formulation [11] with $\mu_{\max} = 1.2$.

- **A-MPCC**: Ablation which only adopts the online parameter estimation from (9) for the predictive model in the MPCC, instead of a time-invariant model.
- **Diff-MPCC**: Ablation which only updates the MPCC weights via the trained PaIML model given the states $\mathbf{x}$ and fixed parameters η_{gt} .

To train the PaIML model for the Diff-MPCC and AD-MPCC methods, data are collected by running a vehicle with the standard MPCC controller using η across different single-surface road conditions, as presented in Subsection III-C. Therein, $\mu_{\max}$ varies from 0.6 to 1.2 in increments of 0.1. The vehicle is initialized with $v_x = 10 \text{ m s}^{-1}$ and completes one lap for each condition, resulting in a dataset of 15,992 samples used to train the PaIML model.

To prioritize vehicle progress speed while maintaining safety, we select the stage cost $\ell = 0.01e_c^2 + e_l^2 - 400\vartheta$, thereby assigning a significantly larger weight to ϑ relative to e_c and e_l . Additionally, the prediction horizon is set to $H = 100$, which corresponds to a 5-second rollout in the outer loop of the MPCC. Algorithm 1 is then used with the specified ℓ and H to generate the output labels for the PaIML model. The PaIML model is implemented using XGBoost with 100 gradient-boosted trees and a maximum depth of 6. For a fair comparison, the objective weights of MPCC and A-MPCC are fixed at $q_c = 1$, $q_l = 1000$, and $q_v = 250$. The MHE parameters are $(T, c, \sigma) = (10, 0.8, 1.0)$ for the single-surface case and $(5, 0.6, 0.2)$ for the multi-surface case.

The simulation environment is configured with a constant sampling time of $\Delta t = 50 \text{ ms}$ and executed on a desktop PC equipped with an Intel® Core™ i7-14700KF processor (20 cores, 28 logical processors, 3.4 GHz), and 32 GB of DDR5 memory. The MPCC prediction horizon is set to $T_h = 20$, corresponding to a 1 s look-ahead. The resulting nonlinear

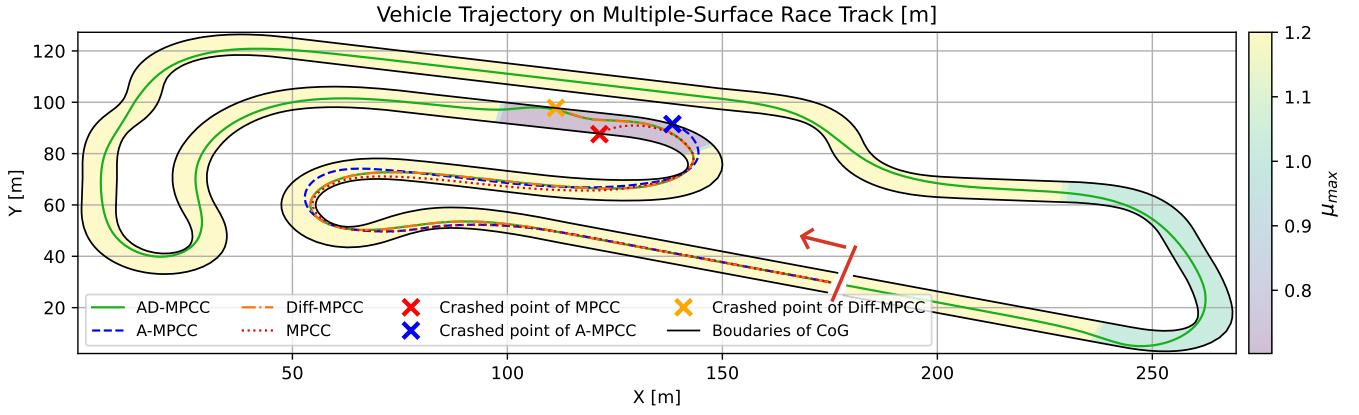


Fig. 7. The trajectories of the racing with the four MPCC controllers. The road consists of multiple surfaces, with the $\mu_{\max}$ varying from 0.7 to 1.2.

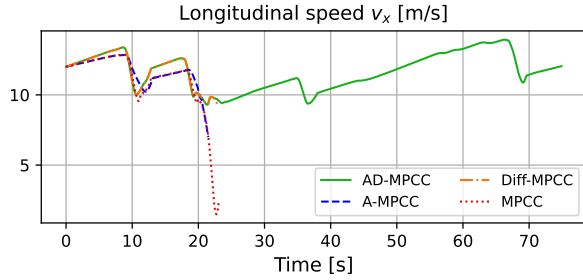


Fig. 8. Longitudinal velocity of the multiple-surface case. Only AD-MPCC can complete the racetrack.

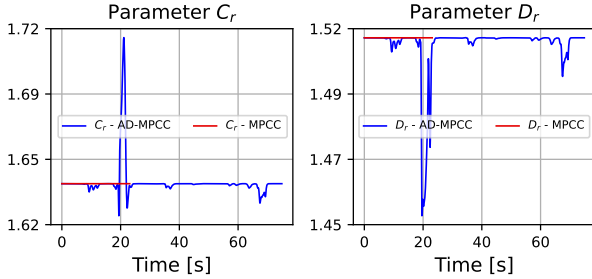


Fig. 9. Online-estimated and fixed Pacejka parameters (results for sample B and C only) in multiple-surface road scenario.

optimization problem is modeled in CASADi [26] and solved by IPOPT [27] on the CPU. Besides, to improve solving time, we may use commercial optimization solvers, such as FORCES PRO [28] and KNITRO [29].

For evaluation, the Oschersleben racetrack from the F1TENTH-Gym environment [15] is used. Its compact layout and frequent curves make it challenging for autonomous racing due to limited recovery space and continuous high-curvature maneuvering. Experiments are conducted using a full-scale AMZ autonomous racing vehicle model under two scenarios: a known single-surface racetrack and an unknown multi-surface racetrack. The half-track width is 4 m, and a collision is declared if the lateral offset between the vehicle's CoG and the centerline exceeds this limit.

B. Single-surface road

In this scenario, all MPCC-based controllers are evaluated on a known road surface with $\mu_{\max} = 1.2$. The vehicle is initialized with $v_x = 10 \text{ m/s}$ and runs for 10 laps.

From Table I, Diff-MPCC and AD-MPCC complete 10 laps faster than the standard MPCC controller. Specifically,

TABLE II
RESULTS OF MULTI-SURFACE CASE FOR 1 LAP.

	MPCC [6]	A-MPCC	Diff-MPCC	AD-MPCC
Avg. lap time [s]	crashed	crashed	crashed	74.9
Avg. v_x [m s^{-1}]	crashed	crashed	crashed	11.65
Avg. lateral offset [m]	crashed	crashed	crashed	1.069
Avg. comp. time [ms]	crashed	crashed	crashed	14.91

Diff-MPCC achieves the lowest lap time of 64.08 s, compared to 75.57 s for MPCC. Moreover, AD-MPCC achieves a similar lap time of 64.89 s while maintaining a smaller mean lateral offset of 0.926 m compared to 1.040 m for Diff-MPCC, indicating that AD-MPCC provides a better trade-off between safety and speed. A-MPCC is about 0.05 s slower per lap than MPCC but reduces the lateral offset by 0.14 m.

Consistent with these faster lap times, Diff-MPCC and AD-MPCC exhibit higher mean longitudinal velocities v_x , although they require slightly longer average computation time per step than MPCC. The MPCC baseline averages 20.45 ms, while our methods range from 21.21 ms to 24.18 ms (smaller than $\Delta t = 50 \text{ ms}$). Additionally, Diff-MPCC and AD-MPCC have smaller mean lateral offsets (below 1.05 m), which is approximately half of those observed for MPCC and A-MPCC. This indicates that the controllers still have margin to further increase speed by prioritizing progress maximization via q_v in (8).

Figs. 4, 5, and 6 show the vehicle trajectory, longitudinal velocity, and Pacejka parameters for the ninth lap. We report the ninth lap as the vehicle has reached steady-state behavior by this point. Fig. 5 shows that Diff-MPCC and AD-MPCC maintain higher velocities than A-MPCC and MPCC for most of the lap. Fig. 6 shows that, even on a known surface, the estimated η slightly fluctuates due to the simplified Pacejka model and simulation uncertainties, but remains close to the nominal constant values.

C. Multiple-surface road

This scenario considers a multiple-surface road where $\mu_{\max}$ varies from 0.7 to 1.2. The vehicle is initialized with a longitudinal velocity of $v_x = 12 \text{ m/s}$ and runs for 1 lap.

Fig. 7 shows that only AD-MPCC successfully completes the lap. In contrast, MPCC, Diff-MPCC, and A-MPCC all crash at the first slippery segment where $\mu_{\max} = 0.75$, indicating that fixed or partially adapted models are insufficient

to handle sudden friction changes. Fig. 9 illustrates that the Pacejka parameters vary in this region when estimated online. For example, the estimated parameter D_r decreases to approximately 1.45, compared to 1.55 for the nominal model, reflecting the reduced tire-road friction. Table II and Fig. 8 show that AD-MPCC not only maintains stability but also achieves competitive performance, with a mean velocity of 11.65 m s^{-1} and an average lateral offset of 1.069 m. The relatively small lateral offset indicates that AD-MPCC still has margin to further increase speed.

In the single-surface scenario, AD-MPCC achieves the smallest lateral error compared to the other methods. This smaller lateral offset contributes to improved safety, which becomes particularly important when racing in unknown road conditions. As shown in Fig. 7, AD-MPCC is the only controller safely navigating the slippery road segment. Our code is published on <https://github.com/nxt-lab/AD-MPCC.git>.

V. CONCLUSION

This paper presented AD-MPCC, a framework that jointly adapts the predictive vehicle model and tunes MPCC objective weights for autonomous racing under multiple road-surface conditions. The approach combines a prior-regularized moving-horizon estimator for online Pacejka parameter estimation with a differentiable MPCC formulation for weight adaptation, which is approximated offline using a Pacejka-informed machine learning model for real-time deployment. Simulation results in the FITENTH-Gym environment demonstrate improved performance on known tracks and robustness in unknown multiple-surface scenarios. Specifically, AD-MPCC achieves a lap time approximately 11 s faster than the standard MPCC on the known single-surface racetrack and is the only method capable of completing the unknown multiple-surface race. Despite these promising results, the current framework relies on simulator-generated data. Future work will focus on real-world deployment on an autonomous racing platform and on providing deeper theoretical analysis to establish the stability and robustness of AD-MPCC.

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 Journal of Intelligent Transportation Systems*, vol. 3, pp. 458–488, 2022.
- [2] X. Sun, M. Zhou, Z. Zhuang, S. Yang, J. Betz, and R. Mangharam, "A benchmark comparison of imitation learning-based control policies for autonomous racing," in *2023 IEEE Intelligent Vehicles Symposium (IV)*, 2023, pp. 1–5.
- [3] K. Yu, M. Fu, T. Zhang, and Y. Yang, "Enhancing safety in autonomous racing with constrained reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 10, no. 6, pp. 6448–6455, 2025.
- [4] Z. Zang, A. Amine, N.-M. T. Kokolakis, T. X. Nghiem, U. Rosolia, and R. Mangharam, "SIT-LMPC: safe information-theoretic learning model predictive control for iterative tasks," *IEEE Robotics and Automation Letters*, vol. 11, no. 1, pp. 986–993, 2026.
- [5] D. Lam, C. Manzie, and M. Good, "Model predictive contouring control," in *49th IEEE Conference on Decision and Control (CDC)*, Dec. 2010, pp. 6137–6142, ISSN: 0191-2216.
- [6] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1: 43 scale RC cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.
- [7] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, 2019.
- [8] J. Pinho, G. Costa, P. U. Lima, and M. Ayala Botto, "Learning-based model predictive control for autonomous racing," *World Electric Vehicle Journal*, vol. 14, no. 7, p. 163, 2023.
- [9] D. R. Gomes, M. A. Botto, and P. U. Lima, "Learning-based model predictive control for an autonomous formula student racing car," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2024, pp. 12 556–12 562.
- [10] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 269–296, 2020.
- [11] H. Pacejka, *Tire and vehicle dynamics*. Elsevier, 2005.
- [12] F. Jahncke, B. Zarrouki, M. Piccinini, J. D'sa, D. Isele, S. Bae, and J. Betz, "Differentiable weights-varying nonlinear MPC via gradient-based policy learning: An autonomous vehicle guidance example," *IEEE Robotics and Automation Letters*, vol. 11, no. 3, pp. 3724–3731, 2026.
- [13] B. Amos, I. D. J. Rodriguez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable MPC for end-to-end planning and control," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 8299–8310.
- [14] E. Adabag, M. Greiff, J. Subosits, and T. Lew, "Differentiable model predictive control on the GPU," *preprint arXiv:2510.06179*, 2025.
- [15] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "FitentH: An open-source evaluation environment for continuous control and reinforcement learning," *Proceedings of Machine Learning Research*, vol. 123, 2020.
- [16] Y. Kebbat, A. Rauh, N. Ait-Oufroukh, D. Ichalal, and V. Vigneron, "Learning-based model predictive control with moving horizon state estimation for autonomous racing," *International Journal of Control*, vol. 98, no. 7, pp. 1542–1552, 2025.
- [17] A. Beck, *First-order methods in optimization*. SIAM, 2017.
- [18] M. Blondel and V. Roulet, "The elements of differentiable programming," *arXiv preprint arXiv:2403.14606*, 2024.
- [19] J. Frey, K. Baumgärtner, G. Frison, D. Reinhardt, J. Hoffmann, L. Fichtner, S. Gros, and M. Diehl, "Differentiable nonlinear model predictive control," *arXiv preprint arXiv:2505.01353*, 2025.
- [20] R. Zuliani, E. C. Balta, and J. Lygeros, "Differentiable-by-design nonlinear optimization for model predictive control," *arXiv preprint arXiv:2509.12692*, 2025.
- [21] M. Blondel, Q. Berthet, M. Cuturi, R. Frostig, S. Hoyer, F. Llinares-López, F. Pedregosa, and J.-P. Vert, "Efficient and modular implicit differentiation," *Advances in neural information processing systems*, vol. 35, pp. 5230–5242, 2022.
- [22] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.
- [23] N. T. Nguyen, B. Nguyen, and T. X. Nghiem, "Physics-informed data-driven modeling of HVAC systems: A systematic analysis," *IEEE Access*, vol. 14, pp. 6481–6500, 2026.
- [24] B. Nguyen, N. T. Nguyen, and T. X. Nghiem, "Structure-and stability-preserving learning of port-hamiltonian systems," *arXiv preprint arXiv:2604.13297*, 2026.
- [25] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 785–794.
- [26] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADI: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.
- [27] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.
- [28] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *International Journal of Control*, vol. 93, no. 1, pp. 13–29, 2020.
- [29] R. H. Byrd, J. Nocedal, and R. A. Waltz, "Knitro: An integrated package for nonlinear optimization," in *Large-scale nonlinear optimization*. Springer, 2006, pp. 35–59.