2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Learning Model Predictive Control with Error Dynamics Regression for Autonomous Racing

Haoru Xue[1], Edward L. Zhu[2], John M. Dolan[1], and Francesco Borrelli[2]

***Abstract*— This work presents a novel Learning Model Predictive Control (LMPC) strategy for autonomous racing at the handling limit that can iteratively explore and learn unknown dynamics in high-speed operational domains. We start from existing LMPC formulations and modify the system dynamics learning method. In particular, our approach uses a nominal, global, nonlinear, physics-based model with a local, linear, data-driven learning of the error dynamics. We conducted experiments in simulation and on 1/10th scale hardware, and deployed the proposed LMPC on a full-scale autonomous race car used in the Indy Autonomous Challenge (IAC) with closed loop experiments at the Putnam Park Road Course in Indiana, USA. The results show that the proposed control policy exhibits improved robustness to parameter tuning and data scarcity. Incremental and safety-aware exploration toward the limit of handling and iterative learning of the vehicle dynamics in high-speed domains is observed both in simulations and experiments.**

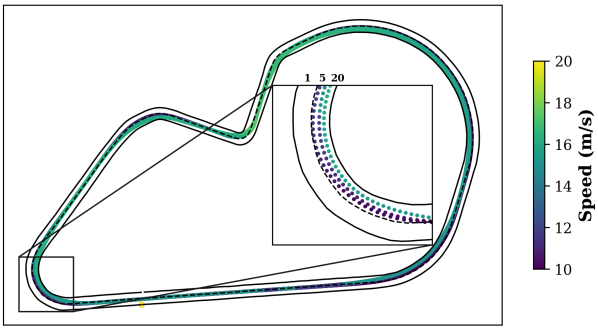

Fig. 1: Overlay of the IAC car's trajectory running LMPC with error dynamics regression. The data are collected at the Putnam Park Road Course in Indiana, USA. The zoomed-in image of the hairpin turn shows iterative improvement towards optimal cornering in the 1st, 5th, and 20th lap.

I. Introduction

Recent progress in autonomous racing research has empowered full-size autonomous race cars at or near the top speed of a professional human racing series. Algorithms that were previously experimented with on small-scale vehicles [1] are deployed and further researched on professional race car platforms. Specifically, the teams in the Indy Autonomous Challenge (IAC) [2], starting in 2021, have been demonstrating multi-agent competition above 240 $\mathrm{km/h}$ [3].

In the field of control, which is tasked with trajectory tracking at the limit of vehicle dynamics handling, various Model Predictive Control (MPC) techniques remain the state-of-the-art on full-size race car systems. These variations of MPC algorithms solve a finite-horizon optimal control problem (FHOCP) by computing optimal actuation commands for a vehicle modeled with kinematic or dynamic models, subject to certain track boundary and state constraints [4], [5], [6], [7]. However, for high-speed racing, the performance of classic, nonadaptive MPC methods is ultimately bounded by the fidelity of the vehicle dynamics model. Inaccurate vehicle models will not only induce suboptimal behavior such as weaving, hunting, and steady-state tracking error, but also become dangerous at the handling limit, especially when the model overpredicts this limit. This introduces a chicken-and-egg problem for autonomous racing research at the handling limit: to safely achieve higher speed levels requires more vehicle dynamics data; but to acquire high-speed data requires the controller to achieve higher speed levels in the first place. Therefore, the motivation of this work is to break the circular problem mentioned earlier by performing incremental and safety-aware exploration toward the limit of handling and iterative learning of the vehicle dynamics in high-speed domains.

In this work, we present a modification to the LMPC approach presented in [8] where we improve significantly upon the model learning strategy. In particular, we propose to learn the error dynamics between a given nominal model and the state evolution of the true system collected during run-time. This is in contrast with the prior approach, which attempts to learn a model of the true system directly. We show through simulation and hardware experiments on a 1/10th scale vehicle that our approach exhibits improved robustness to parameter tuning and reduced modeling errors in scenarios when data are scarce for model learning. We finally demonstrate the effectiveness of our approach on a full-size IAC autonomous race car, the results of which are shown in Figure 1. An open source C++ implementation of our work has been made available at `https://github.com/MPC-Berkeley/Racing-LMPC-ROS2`.

Related Work: Learning unknown or changing system dynamics has been studied within some classic control frameworks such as adaptive control [9], [10]. System identification is usually part of the controller design, which reasons about some modeling error or noise in the system and adapts the system model to account for them. In autonomous racing, previous works have proposed dynamics learning strategies using Gaussian Processes [11], [12], [13] or implicitly with Gaussian noise assumption [14]. The method is further

[1]Haoru Xue and John M. Dolan are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 15213, USA {haorux, jdolan}@andrew.cmu.edu

[2]Edward L. Zhu and Francesco Borrelli are with the Department of Mechanical Engineering, University of California Berkeley, Berkeley, CA 94720, USA {edward.zhu, fborrelli}@berkeley.edu

extended to model multi-vehicle interactions and solve for the optimal overtaking maneuver [15], [16].

Another branch of works exploits the repetitive nature of autonomous racing in a single-vehicle scenario, and improves controller performance without any prior assumption about the distribution of the modeling error or noise. Data-driven methods, such as iterative learning control, can optimize the tracking of a repetitive trajectory given data from previous iterations by solving for control variable corrections [17]. More recently, the learning model predictive control (LMPC) framework has been proposed with a direct regression of local affine dynamics [18], [19], [8]. This combination of LMPC and regressive dynamics learning can iteratively learn the unknown system dynamics and reduce lap-time without an explicit prior reference trajectory or a nominal dynamics model. The approach has also been demonstrated on scaled vehicle hardware platforms in [18].

II. Problem Formulation

In this work, we model the vehicle using the planar dynamic bicycle model [20], [21] with state and input vectors $x = [v_x, v_y, \omega_z, e_\psi, s, e_y]^\top$, $u = [a, \delta]^\top$, where v_x, v_y, and ω_z are the vehicle's longitudinal velocity, lateral velocity, and yaw rate at the center of gravity. s, e_y, and e_ψ are the pose of the vehicle expressed in the Frenet frame w.r.t. a parametric path $\tau : [0, L] \mapsto \mathbb{R}^2$, which is assumed to be continuously twice differentiable and of total length L. In particular, when given an arclength $s \in [0, L]$, τ returns the global x-y position of the path. In the case where τ forms a closed circuit, i.e., $\tau(0) = \tau(L)$, $\tau'(0) = \tau'(L)$, $\tau''(0) = \tau''(L)$, where τ' and τ'' are the element-wise first and second derivatives of τ, we define the circuit as $\bar{\tau}(s) = \tau(\text{mod}(s, L))$ for $s \geq 0$, where mod is the modulo operator. Given a path τ and a vehicle with global position $p = (x, y)$ and heading ψ, the Frenet frame pose is defined as the vehicle's path progress $s(p) = \arg\min_s \|\tau(s) - p\|_2$, its lateral deviation from the path $e_y(p) = \min_s \|\tau(s) - p\|_2$, and its heading deviation from the path tangent $e_\psi(p, \psi) = \psi - \arctan(\tau'(s(p)))$. The inputs to the model are the longitudinal acceleration a and the front steering angle δ. We write the discretized nonlinear dynamics as:

$$x_{k+1} = f(x_k, u_k) \tag{1}$$

The vehicle's state and input are subject to the constraints $\mathcal{X} = \{x \mid -W/2 \leq e_y \leq W/2\}$ and $\mathcal{U} = \{u \mid u_l \leq u \leq u_u\}$, which describe the boundary constraints for a track of constant width W and the limits on achievable acceleration and steering angle.

The objective of this work is to design a control policy which solves the infinite-horizon, minimium lap time, optimal control problem:

$$T = \min_{u_0, u_1, \ldots} \quad \sum_{k=0}^{\infty} \mathbb{1}_{\mathcal{F}}(x_k) \tag{2a}$$

$$\text{subject to} \quad x_0 = \bar{x}, \tag{2b}$$

$$x_{k+1} = f(x_k, u_k), \tag{2c}$$

$$x_k \in \mathcal{X},\ u_k \in \mathcal{U}, \tag{2d}$$

where $\bar{x} \in \mathcal{S} = \{x \mid s = 0\}$ is a state at the start of the track and $\mathcal{F} = \{x \mid s \geq L\}$ is the set of states beyond the finish line of the circuit. If for some time step k, $x_k \in \mathcal{F}$ for the first time, then the vehicle has finished the lap and has transitioned onto the next lap. $\mathbb{1}_{\mathcal{F}}$ is the indicator function for the set $\mathcal{F}$, which is defined as

$$\mathbb{1}_{\mathcal{F}}(x) = \begin{cases} 0 & \text{if } x \in \mathcal{F} \\ 1 & \text{otherwise} \end{cases}$$

Due to the infinite horizon cost in (2a), it should be clear that the optimal control problem posed in (2) is not amenable to online implementation. Moreover, it is likely that there is mismatch between the vehicle model in (2c) and the true vehicle dynamics, the effects of which would be especially apparent at high speeds, where the vehicle is operating at the limit of handling. To address these challenges, previous work on LMPC [19], [22] proposed a finite-horizon convex approximation of (2), which uses state and input trajectories from prior laps to synthesize a convex terminal cost-to-go function and target set. The same data are additionally used to identify affine time-varying (ATV) models which approximate the true dynamics of the vehicle. In this work, we carry over the same approach for terminal cost-to-go function and target set synthesis, but propose a modification to the system identification procedure which results in an LMPC control policy that achieves similar performance to [19], [22] w.r.t. minimum lap time, while exhibiting a significant improvement in robustness to tuning parameters.

III. Local Learning Model Predictive Control

In this section, we briefly describe the procedure for synthesizing LMPC policies over iterations (or laps) of the minimum time control task. For further details, we refer the reader to [19], [22].

The main idea of LMPC is to use the data from iterations 1 to $j - 1$ to synthesize a finite-horizon local convex approximation of (2) for iteration j, which is then solved in a receding horizon manner. Let us first denote the available dataset at iteration j as $\mathcal{D}^j = (\mathbf{x}^j, \mathbf{u}^j) \cup \mathcal{D}^{j-1}$, where $\mathbf{x}^j = \{x_0^j, x_1^j, \ldots, x_{T^j}^j\}$ and $\mathbf{u}^j = \{u_0^j, u_1^j, \ldots, u_{T^j}^j\}$ are the closed-loop state and input sequence for iteration j and T^j denotes the time step at which the lap was completed for iteration j, i.e., the first time step where $\mathbb{1}_{\mathcal{F}}(x_k) = 0$. Note that on the first iteration, the dataset $\mathcal{D}^0$ may be initialized using human-driven laps or closed-loop trajectories from a simple low-speed center-line tracking controller. In addition, we assume that the trajectories stored in $\mathcal{D}^j$ are feasible w.r.t. the constraints (2d) and reach the finish line in a finite amount of time.

A. Local Convex Target Set

As the closed-loop trajectories stored in $\mathcal{D}^j$ are from the actual vehicle, it is straightforward to see that if we can control the vehicle to any state in $\mathcal{D}^j$, then there exists a known feasible control sequence which can be used to complete the lap from that state. LMPC uses this fact to construct a safe set as a discrete collection of the states in $\mathcal{D}^j$, which is then used as the terminal set for a FHOCP [22].

In this work, we construct convex terminal sets which are local about a given state x. These terminal sets sacrifice the theoretical guarantee of persistent feasibility for the FHOCP in favor of reduced computational complexity for fast online control. In particular, for a given x, we take the K-nearest neighbors from each of the previous P laps by evaluating the weighted Euclidean distance over the relevant states:

$$(x_k^i - x)^\top D(x_k^i - x),\ \forall i \in \{j, j-1, \dots, j-(P-1)\},\ k \in \{0, 1, \dots, T^i\} \tag{3}$$

for some given P and $D \succeq 0$. Letting $\mathbf{X}^j(x; \mathcal{D}^j) \in \mathbb{R}^{n \times KP}$ denote the matrix formed by the KP states closest to x, we then define the target set as the convex hull of these states: $\mathcal{X}_N^j(x; \mathcal{D}^j) = \{\bar{x} \in \mathbb{R}^n \mid \exists \lambda \in \mathbb{R}^{KP},\ 0 \le \lambda \le 1,\ \mathbf{1}^\top \lambda = 1, \mathbf{X}^j(x; \mathcal{D}^j)\lambda = \bar{x}\}$.

B. Local Terminal Cost-to-go Function

By assumption, the closed-loop state trajectories stored in $\mathcal{D}^j$ reach the finish line in a finite amount of time. We can therefore compute the cost-to-go for any state x_k^i in $\mathcal{D}^j$ as $T^i - k$, which represents the time remaining to finish the i-th lap for $i \in \{0, \dots, j\}$. Let $J_N^j(x; \mathcal{D}^j) \in \mathbb{R}^{KP}$ denote the vector of cost-to-go values calculated for the states in $\mathbf{X}^j(x; \mathcal{D}^j)$. We can therefore define the minimum cost-to-go function for any $\bar{x} \in \mathcal{X}_N^j(x; \mathcal{D}^j)$ as:

$$\begin{aligned} &Q_N^j(\bar{x}, x; \mathcal{D}^j) \\ &\quad = \min_{\lambda \in \mathbb{R}^{KP}} \ J_N^j(x; \mathcal{D}^j)^\top \lambda \\ &\quad \text{subject to } \ 0 \le \lambda \le 1,\ \mathbf{1}^\top \lambda = 1,\ \mathbf{X}^j(x; \mathcal{D}^j)\lambda = \bar{x} \end{aligned}$$

This function can be thought of as a local convex approximation to the optimal cost-to-go (from solving (2)) at iteration j, which when incorporated as the terminal cost-to-go function, allows an FHOCP to reason about infinite-horizon behavior of the vehicle, to the extent that it is captured in the dataset.

C. Local LMPC

Using the terminal target set and cost-to-go function synthesized from the dataset $\mathcal{D}^{j-1}$, we may now construct the FHOCP at time step k of iteration j for our local LMPC policy as follows:

$$\begin{aligned} &J_k^j(x_k, u_{k-1}, \bar{\mathbf{z}}_k; \mathcal{D}^{j-1}) = \\ &\min_{\mathbf{x}, \mathbf{u}, \lambda} \ \sum_{t=0}^{N-1} \mathbb{1}_{\mathcal{F}}(x_t) + c_u \|u_t\|_2^2 + c_{\Delta u}\|u_t - u_{t-1}\|_2^2 \\ &\qquad + J_N^{j-1}(\bar{x}_{k+N}; \mathcal{D}^{j-1})^\top \lambda && \text{(4a)} \\ &\text{subject to } \ x_0 = x_k,\ u_{-1} = u_{k-1} && \text{(4b)} \\ &\qquad x_{t+1} = A(\bar{z}_{k+t}; \mathcal{D}^{j-1})x_t + B(\bar{z}_{k+t}; \mathcal{D}^{j-1})u_t \\ &\qquad\quad + C(\bar{z}_{k+t}; \mathcal{D}^{j-1}),\ t \in \{0, \dots, N-1\}, && \text{(4c)} \\ &\qquad x_t \in \mathcal{X},\ u_t \in \mathcal{U},\ t \in \{0, \dots, N\} && \text{(4d)} \\ &\qquad \mathbf{X}^{j-1}(\bar{x}_{k+N}; \mathcal{D}^{j-1})\lambda = x_N, && \text{(4e)} \\ &\qquad 0 \le \lambda \le 1,\ \mathbf{1}^\top \lambda = 1 && \text{(4f)} \end{aligned}$$

where $\bar{z}_k = (\bar{x}_k, \bar{u}_k)$ and $\bar{\mathbf{z}}_k = \{\bar{z}_k, \dots, \bar{z}_{k+N}\}$ is a state and input sequence which is used to form the local convex approximation. In practice, this is chosen as the solution to (4) from the previous time-step. Construction of the A, B, and C matrices in the ATV dynamics (4c) is the main contribution of this work and will be discussed in the next section. Note that (4) is a convex program which can be solved efficiently using existing solvers.

Finally, let $\mathbf{u}^\star = \{u_0^\star, \dots, u_{N-1}^\star\}$ be the optimal control sequence which solves (4) at time step k. The input $u_k = u_0^\star$ is applied to the vehicle and the FHOCP (4) is repeated at time step $k+1$ for the measured state x_{k+1} until the vehicle reaches the finish line, at which point $\mathcal{D}^j$ will be constructed using the closed-loop trajectory for lap j.

IV. Learned Vehicle Dynamics Model

We now introduce the main contribution of our work, which is a learning strategy on the dynamics states v_x, v_y, ω_z based on regressive error. The learning process takes as inputs the nominal dynamics model and a set of neighboring states, and outputs a learned local affine approximation of the true vehicle dynamics. We assume that the true dynamics of the system can be written as follows

$$x^+ = f(x, u) + e(x, u), \tag{5}$$

where f are the nominal dynamics from (1) and e denotes some unknown modeling error which we would like to identify using data. For a given state and input x and u, let us denote the prediction of the nominal model as $\hat{x}^+ = f(x, u)$. We may now linearize the error dynamics about a reference state and input $\bar{z}$ as follows:

$$x^+ - \hat{x}^+ = A^e x + B^e u + C^e, \tag{6}$$

where A^e and B^e are the Jacobians of e w.r.t. x and u evaluated at $\bar{z}$ and $C^e = e(\bar{x}, \bar{u}) - A^e\bar{x} - B^e\bar{u}$. It can be clearly seen from (6) that the linearization of the unknown error term is related to the system state and input x and u, the actual state evolution x^+, and the prediction of the nominal model $\hat{x}^+$, which are all quantities that can be easily obtained from the dataset $\mathcal{D}^{j-1}$ at iteration j. As such, we query $\mathcal{D}^{j-1}$ to find the M nearest neighbors $\mathbf{z} = \{z_1, \dots, z_M\}$ of $\bar{z}$ and their corresponding state evolutions $\mathbf{x}^+ = \{x_1^+, \dots, x_M^+\}$ (using a technique similar to (3)). We then compute the prediction of the nominal model at each of the state and input pairs in $\mathbf{z}$ to obtain $\hat{\mathbf{x}}^+ = \{\hat{x}_1^+, \dots, \hat{x}_M^+\}$. As mentioned before, we are interested in learning only the error dynamics associated with the velocity components of the state x, as the kinematic components of the state are well-understood. Therefore, using the datasets $\mathbf{z}$, $\mathbf{x}^+$, and $\hat{\mathbf{x}}^+$, we define the residuals for each velocity state as follows:

$$\begin{aligned} A^e[11:13]x_m[1:3] + B^e[11]a_m + C^e[1] &= v_{x,m}^+ - \hat{v}_{x,m}^+, \\ A^e[21:23]x_m[1:3] + B^e[22]\delta_m + C^e[2] &= v_{y,m}^+ - \hat{v}_{y,m}^+, \\ A^e[31:33]x_m[1:3] + B^e[32]\delta_m + C^e[3] &= \omega_{z,m}^+ - \hat{\omega}_{z,m}^+, \end{aligned} \tag{7}$$

where $m \in \{1, \dots, M\}$ and we use the notation $A[11:13]$ to index the first three elements of the first row of matrix A. Note that we have injected additional structure into the individual residuals by making explicit the dependence of each velocity state on a certain subset of the input compo-

nents. From these three expressions, we define the following regressor vectors:

$$\Gamma_{v_x} = \begin{bmatrix} A^e[11{:}13] \\ B^e[11] \\ C^e[1] \end{bmatrix}, \Gamma_{v_y} = \begin{bmatrix} A^e[21{:}23] \\ B^e[22] \\ C^e[2] \end{bmatrix}, \Gamma_{\omega_z} = \begin{bmatrix} A^e[31{:}33] \\ B^e[32] \\ C^e[3] \end{bmatrix}.$$

We solve the regression problem in a manner similar to [8], where we weigh the importance of data point m using the Epanechnikov kernel function [23] with bandwidth h:

$$K(u) = \begin{cases} \frac{3}{4}(1 - u^2/h^2), & |u| < h \\ 0, & \text{otherwise} \end{cases}.$$

such that larger penalties are assigned to the residuals for data points that are close to the linearization point $\bar{x}$. For $l = \{v_x, v_y, \omega_z\}$, the weighted least squares problem is defined as follows:

$$\Gamma_l^\star = \underset{\Gamma_l}{\operatorname{argmin}} \sum_{m=1}^{M} K\left(||\bar{z} - z_m||_Q^2\right) y_m^l(\Gamma_l) + \epsilon \|\Gamma_l\|_2^2, \quad (8)$$

where $Q \succeq 0$ denotes the relative scaling between the variables, $y_m^l(\Gamma_l)$ are the l^2-norms of the residuals in (7), and $\epsilon > 0$ is a regularization parameter.

Finally, we can construct the matrices A^e, B^e, and C^e from $\Gamma_l^\star$ as follows:

$$A^e = \begin{bmatrix} \Gamma_{v_x}^\star[1:3] \\ \Gamma_{v_y}^\star[1:3] \\ \Gamma_{\omega_z}^\star[1:3] \\ \mathbf{0}_{3\times 3} \end{bmatrix} \mathbf{0}_{6\times 3}, \; B^e = \begin{bmatrix} \Gamma_{v_x}^\star[4] & 0 \\ 0 & \Gamma_{v_y}^\star[4] \\ 0 & \Gamma_{\omega_z}^\star[4] \\ \mathbf{0}_{3\times 2} \end{bmatrix},$$

$$C^e = \begin{bmatrix} \Gamma_{v_x}^\star[5] & \Gamma_{v_y}^\star[5] & \Gamma_{\omega_z}^\star[5] & \mathbf{0}_{1\times 3} \end{bmatrix}^\top.$$

These are then added to the nominal ATV dynamics to obtain the complete ATV model for (4c): $A(\bar{z}; \mathcal{D}^{j-1}) = A^f + A^e$, $B(\bar{z}; \mathcal{D}^{j-1}) = B^f + B^e$, and $C(\bar{z}; \mathcal{D}^{j-1}) = C^f + C^e$, where A^f and B^f are the Jacobians of f w.r.t. x and u evaluated at $\bar{z}$ and $C^f = f(\bar{x}, \bar{u}) - A^f \bar{x} - B^f \bar{u}$. We note that the key difference between this procedure and that in [8] is that the regressive learning is performed on the *error* between the prediction of the nominal model and the actual state evolution data instead of on the entire model in (5).

To understand why our approach to model learning is more robust to scarce data, consider the case where all $\|\bar{z} - z_m\|_Q$ in (8) are close to or greater than the kernel bandwidth h (i.e., when the data z_m are far from the linearization point $\bar{z}$ w.r.t. h). When this occurs, $K(\|\bar{z} - z_m\|_Q^2) \approx 0$, which means that the regularization term in (8) dominates the least squares objective and we can therefore conclude that the optimal solution $\Gamma_l^\star \approx 0$. When regressing the nominal dynamics matrices directly, as in [8], this would result in setting elements of those matrices to approximately zero, which would be disastrous in terms of model accuracy. On the other hand, when regressing the error matrices, setting elements to zero would simply correspond to *not* applying any data-based correction to the nominal linearized dynamics. This allows for a natural trade-off between the nominal and data-corrected models based on the quality of the dataset. Of course, this could be mitigated by the selection of a large enough bandwidth h. However, as this is a tuning parameter which is meant to act as a threshold on a weighted distance metric in high dimensions, it may not be immediately apparent what constitutes a good setting. We therefore want the LMPC policy to be robust to values of h to allow for safe tuning of the controller.

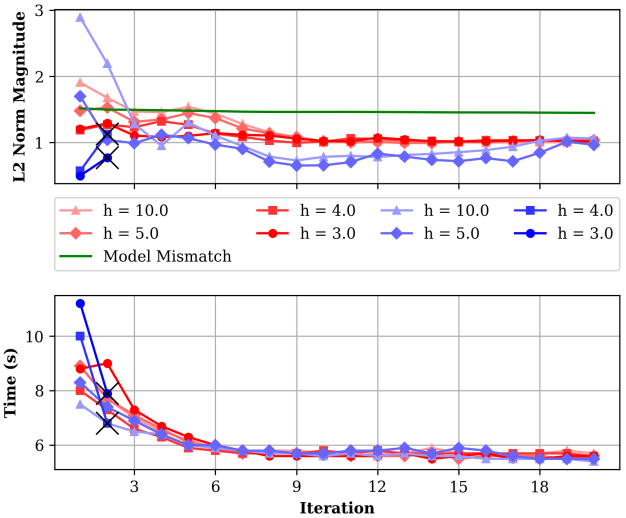

Fig. 2: Average error of the learned model (Top) and lap times (bottom) over iterations for various settings of the bandwidth h. Red and blue lines correspond to the error regression and full regression cases, respectively. A cross is placed where failure occurs due to the vehicle leaving the track.

V. Experiments & Results

We now present a set of experiments in simulation and hardware to demonstrate the effectiveness of combining LMPC and error dynamics regression in autonomous racing. The key metrics throughout these experiments are

- 20th-iteration lap time (ILT-20): the average lap time after running the LMPC for 20 iterations.
- Iteration to fail (ITF): the average number of iterations before an experiment fails due to loss of control or track boundary violation.

A. Robustness Study on Learning Parameters

This experiment aims to study how LMPC parameter tunings affect the safety of the learning process. The motivation is that there are not many trial-and-error opportunities when tuning parameters on safety-critical hardware such as a race car, and a robust control setup should be able to handle a broad range of parameter settings without destabilizing the system. We set up simulation experiments to compare our approach against the LMPC implementation in [8] in terms of their robustness to the change in the bandwidth parameter h and the control-rate cost (CRC) ($c_{\Delta u}$ in (4a)). Intuitively, the bandwidth h governs a trade off between the quality and quantity of the data used for model regression. Whereas a high bandwidth allows for potentially more data points to be selected, they can be far (in the sense of the norm $\|\cdot\|_Q$) from the linearization point. On the other hand, a low bandwidth requires that the data points be close to the linearization point, but could result in few data points being selected when data is sparse. As for the CRC, it is analogous to the "learning rate" for LMPC. A higher CRC will penalize drastic changes in the control input and keep

the vehicle closer to the safe set over the MPC horizon. Both h and the CRC are therefore key parameters when tuning the behavior of the LMPC policy. For all simulation studies, we use the L-shaped track shown in Figure 5 and model the rigid body with the dynamic bicycle model. The tire forces are modeled with the Pacejka tire model [24] with tire-road friction coefficient $\mu = 0.9$.

In our first simulation experiment, we set CRC $= 0.1$ and compare the effect of the bandwidth h on the accuracy of the learned model for our error regression LMPC and the full regression LMPC from [8]. In the error regression case, we use the kinematic bicycle model as the nominal model in (1) which induces mismatch with the simulation model. To measure accuracy of the learned model, we compute the Frobenius norm of the difference between the learned dynamics matrix $A(\bar{z}, \mathcal{D}^{j-1})$ and the linearized dynamics of the simulation model at each time step and record the average over each iteration. The results for $h = \{3, 4, 5, 10\}$ are shown in Figure 2, where we may immediately observe that for high settings of h, both the error and full regression cases perform similarly in both the learned model accuracy and lap times. Importantly, we see that the error regression case can correct for the mismatch between the nominal and simulation models, which is shown by the green line. For low settings of h, it can be seen that while [8] fails after the second iteration, the error regression LMPC is able to remain stable and converge to similar performance as the high bandwidth cases. This is due to the reasons discussed in Section IV where sparsity w.r.t. h can result in significant model inaccuracies in [8], but would naturally induce a fall back to the nominal model in the error regression case. Sparsity in the regression data is especially acute in the initial laps when the LMPC policy is rapidly improving vehicle performance as there can be significant differences in the data between successive laps. We note that the behavior of the error regression case is largely dependent on the accuracy of the nominal model when data is sparse and stability cannot be guaranteed for arbitrary realizations of the nominal model. However, especially in car racing, it is not unreasonable that to assume access to a fairly accurate nominal model which can be obtained through standard system identification approaches. Future work will examine the effect of distance between the nominal and true system on the stability of the error regression LMPC.

In our second simulation experiment, we fix the bandwidth to $h = 5$ and ran both LMPC implementations with CRC $= \{1.0, 0.5, 0.1, 0.05, 0.01\}$ and recorded the lap time at every iteration. Like the previous study, we purposely introduce mismatch in the nominal model. Though for this study, we do so through the tire-road friction coefficient. Whereas a value of 0.9 is used in the simulation model, a value of 1.2 is chosen for the LMPC nominal dynamics model in (1), which is typically only seen on full-scale race cars with racing slick tires. This would have induced dangerous over-confidence in the model about the surface friction, as shown in the following hardware experiments, if no dynamics learning is used. Table I presents the key results of this experiment,

Control Rate Cost	ILT-20		ITF	
	[8]	ours	[8]	ours
1.0	**6.4**	6.5	20+	20+
0.5	**6.0**	6.2	20+	20+
0.1	**5.5**	5.6	20+	20+
0.05	-	**5.2**	6	**20+**
0.01	-	**5.0**	4	**20+**

TABLE I: Results of the simulation robustness study. A hyphen in the 20th-Iteration Lap Time (ILT-20) means that the algorithm fails before the 20th iteration. An Iteration to Fail (ITF) of 20+ means that the algorithm did not fail in the 20-iteration experiment.

LMPC Configuration	Avg. ILT-20	Avg. ITF
Full Regression ([8], $h = 10.0$)	6.69	20+
Error Regression (ours, $h = 10.0$)	6.82	20+
Full Regression ([8], $h = 3.0$)	6.53	10.5
Error Regression (ours, $h = 3.0$)	6.66	19.1

TABLE II: Results of the BARC hardware experiment.

where we observe that with CRC penalties of 1.0, 0.5, and 0.1, both [8] and our approach show similar performance at the end of 20 laps, with [8] achieving slightly faster lap time reduction as evidenced in Figure 3. However, once we decrease the CRC below 0.1, we see that [8] fails to complete the 20 laps, whereas our method exhibits greater robustness to different settings of the CRC and can still converge to high performing lap times within 20 iterations.

Overall, our simulation experiments suggest that LMPC with error dynamics regression exhibits greater stability and robustness to key parameter settings and works well even when scarce dynamics data are available. These properties could potentially make the system more suitable for hardware deployment in safety-critical applications.

B. Hardware Experiment Comparing Learning Performance

In this section, we present results from physical experiments with the Berkeley Autonomous Race Car (BARC) platform. These vehicles are based on 1/10th scale, off-the-shelf RC cars that have been modified for autonomous racing. LMPC computation is done on a laptop computer with a 2.6 GHz 9th-Gen Intel Core i7 CPU, localization is provided by a motion capture system, and communication with the vehicle is handled with ROS2. We conducted two sets of experiments to demonstrate the effect of the bandwidth h on the stability and learning performance of [8] and our error regression LMPC. Each set of experiments consists of 10 runs of at most 20 laps with CRC $= 0.1$. We record the lap times and record any failure cases before the 20-lap threshold.

In the first experiment, we run [8] and our error regression LMPC with $h = 10$. This corresponds to the high bandwidth case in the simulation study where both policies achieved similar performance and were able to successfully complete 20 laps. The results of our experiment are shown in the top row of plots in Figure 4, where we again see that the two approaches show similar performance in lap time reduction and are able to remain stable over 20 laps. This corroborates our simulation study and shows that with proper tuning, our error regression LMPC is essentially equivalent to [8]. In the second experiment, we run [8] and our error regression LMPC with $h = 3$. This corresponds to the low bandwidth

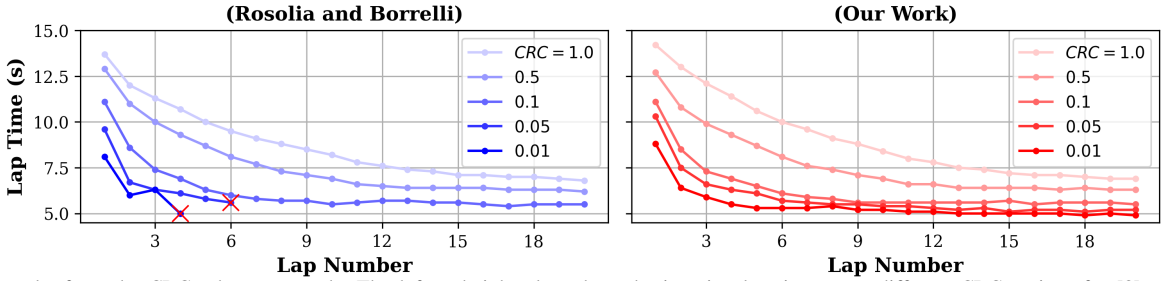

Fig. 3: Results from the CRC robustness study. The left and right plots show the iteration lap times over different CRC tunings for [8] and our error regression LMPC respectively. A red cross is placed where failure occurs due to the vehicle leaving the track.

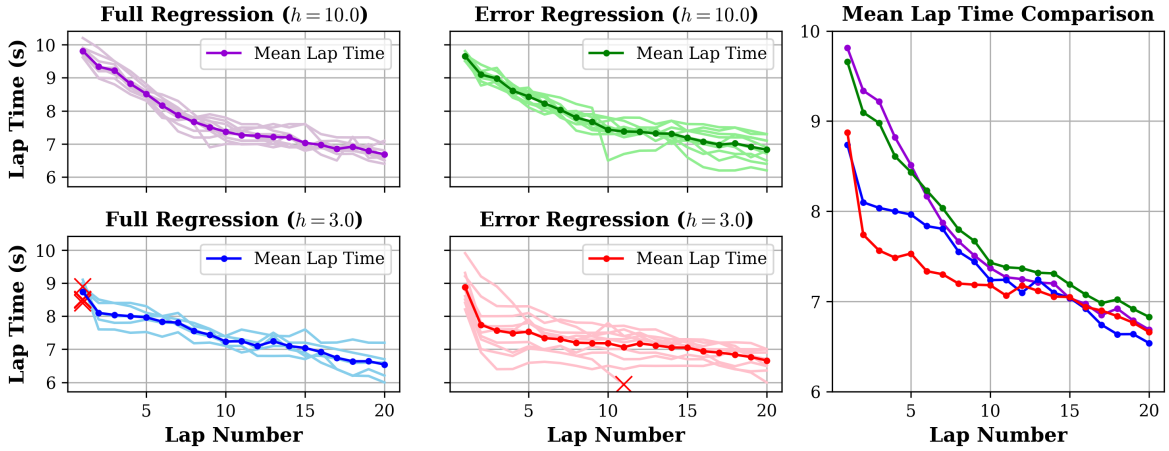

Fig. 4: Per-iteration results of the BARC hardware experiment. The four smaller plots on the left show the lap-time reduction of the 10 trials. The combined plot on the right compares the average lap times achieved. A red cross is placed where failure occurs due to the vehicle leaving the track.

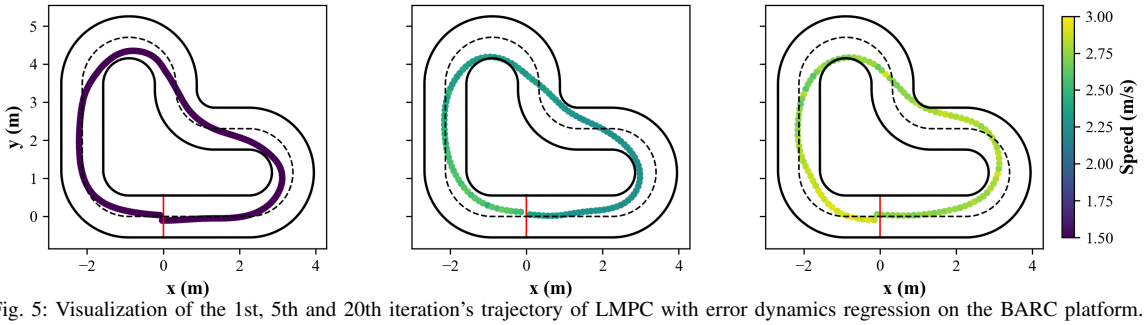

Fig. 5: Visualization of the 1st, 5th and 20th iteration's trajectory of LMPC with error dynamics regression on the BARC platform.

case in the simulation study where we data sparsity has a destabilizing effect on [8]. The results of our experiment are shown in the bottom row of plots in Figure 4, which also supports our observations from the simulation study. In particular, we observe that when a low bandwidth setting is used, [8] failed in 5 of the 10 trials, whereas our error learning LMPC only failed in a single trial.

C. Hardware Demonstration on a Full-Size Race Car

In this demonstration, we deployed our LMPC and error dynamics regression strategy on a full-size *IndyLights* race car which has been used in the Indy Autonomous Challenge [3]. The race track used for this demonstration is the Putnam Park Road Course in Indiana, USA. We construct the initial dataset $\mathcal{D}^0$ for the vehicle by running a tracking MPC on the center line of the track at 10 $\mathrm{m/s}$ for 3 laps. We note that this step could also be done with a more rudimentary controller. We then ran LMPC with error dynamics regression for 10 iterations. Figure 1 visualizes the trajectory driven by the vehicle, with a zoomed-in view to highlight how LMPC optimizes the shape of the trajectory for a hairpin turn. It can be clearly seen in Figure 1 that the vehicle starts at lower speeds closer to the center line and incrementally increases its speed through the corners and approaches the track boundary constraint.

VI. Conclusion

This work proposes a new error dynamics learning approach under the LMPC framework to iteratively and safely explore the boundary of vehicle handling limit in autonomous racing. We based the learning on a nominal dynamics model, which provides explanability and safety fallback for the learning process. We then derived the regression method on top to learn a local approximation of the nonlinear model mismatch. Through simulation and hardware experiments, we showed that our method exhibits greater robustness against parameter tuning and data scarcity compared with previous LMPC works, and offers new potential solutions to address the challenge of vehicle dynamics data acquisition in high-speed autonomous racing.

REFERENCES

[1] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "F1TENTH: An Open-source Evaluation Environment for Continuous Control and Reinforcement Learning," in *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*. PMLR, Aug. 2020, pp. 77–89.

[2] "Indy Autonomous Challenge," Feb. 2023. [Online]. Available: https://www.indyautonomouschallenge.com

[3] A. Wischnewski, M. Geisslinger, J. Betz, T. Betz, F. Fent, A. Heilmeier, L. Hermansdorfer, T. Herrmann, S. Huch, P. Karle, F. Nobis, L. Ögretmen, M. Rowold, F. Sauerbeck, T. Stahl, R. Trauth, M. Lienkamp, and B. Lohmann, "Indy autonomous challenge – autonomous race cars at the handling limits," 2022.

[4] A. Wischnewski, T. Herrmann, F. Werner, and B. Lohmann, "A Tube-MPC Approach to Autonomous Multi-Vehicle Racing on High-Speed Ovals," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 368–378, Jan. 2023.

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

[6] 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," *Journal of Field Robotics*, vol. 40, no. 4, pp. 783–809, 2023.

[7] J. Funke, P. Theodosis, R. Hindiyeh, G. Stanek, K. Kritatakirana, C. Gerdes, D. Langer, M. Hernandez, B. Müller-Bessler, and B. Huhnke, "Up to the limits: Autonomous Audi TTS," in *2012 IEEE Intelligent Vehicles Symposium*, Jun. 2012, pp. 541–547.

[8] U. Rosolia and F. Borrelli, "Learning How to Autonomously Race a Car: A Predictive Control Approach," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2713–2719, Nov. 2020.

[9] M. Bujarbaruah, X. Zhang, U. Rosolia, and F. Borrelli, "Adaptive mpc for iterative tasks," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 6322–6327.

[10] A. Sasfi, M. N. Zeilinger, and J. Köhler, "Robust adaptive mpc using control contraction metrics," *Automatica*, vol. 155, p. 111169, 2023.

[11] 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.

[12] L. Hewing, A. Liniger, and M. N. Zeilinger, "Cautious NMPC with Gaussian Process Dynamics for Autonomous Miniature Race Cars," in *2018 European Control Conference (ECC)*, Jun. 2018, pp. 1341–1348.

[13] J. Ning and M. Behl, "Vehicle dynamics modeling for autonomous racing using gaussian processes," 2023.

[14] D. Kalaria, Q. Lin, and J. M. Dolan, "Delay-aware robust control for safe autonomous driving," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022, pp. 1565–1571.

[15] E. L. Zhu, F. L. Busch, J. Johnson, and F. Borrelli, "A gaussian process model for opponent prediction in autonomous racing," 2023.

[16] T. Brüdigam, A. Capone, S. Hirche, D. Wollherr, and M. Leibold, "Gaussian process-based stochastic model predictive control for overtaking in autonomous racing," 2021.

[17] D. Bristow, M. Tharayil, and A. Alleyne, "A survey of iterative learning control," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, 2006.

[18] U. Rosolia and F. Borrelli, "Learning Model Predictive Control for Iterative Tasks: A Computationally Efficient Approach for Linear System," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3142–3147, Jul. 2017.

[19] M. Brunner, U. Rosolia, J. Gonzales, and F. Borrelli, "Repetitive learning model predictive control: An autonomous racing example," in *2017 IEEE 56th annual conference on decision and control (CDC)*. IEEE, 2017, pp. 2545–2550.

[20] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2015-August, pp. 1094–1099, 2015.

[21] F. Christ, A. Wischnewski, A. Heilmeier, and B. Lohmann, "Time-optimal trajectory planning for a race car considering variable tyre-road friction coefficients," *Vehicle System Dynamics*, vol. 59, no. 4, pp. 588–612, Apr. 2021.

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

[23] V. A. Epanechnikov, "Non-Parametric Estimation of a Multivariate Probability Density," *Theory of Probability & Its Applications*, vol. 14, no. 1, pp. 153–158, Jan. 1969.

[24] H. B. Pacejka and E. Bakker, "The magic formula tyre model," *Vehicle system dynamics*, vol. 21, no. S1, pp. 1–18, 1992.