Driving Closer to the Limit: Improved Virtual Racecar Drivers with Data-Driven Control ⋆

Ruslan Shaiakhmetov[1][0009−0001−0869−6204], Danilo Pianini[1][0000−0002−8392−5409], Valter Venusti[2][0009−0001−0647−3015], Gabriele D'Angelo[1][0000−0002−3690−6651], and Alessandro V. Papadopoulos[3][0000−0002−1364−8127]

[1] Alma Mater Studiorum—Università di Bologna, Cesena, Italy
{ruslan.shaiakhmetov,danilo.pianini,g.dangelo}@unibo.it
[2] Dallara Automobili S.p.A., Parma, Italy V.Venusti@dallara.it
[3] Mälardalen University, Västerås, Sweden alessandro.papadopoulos@mdu.se

Abstract. Accurate simulation of racing cars is crucial in motorsport to quickly identify effective setups before track testing. Typically, professional drivers provide feedback in simulation, but this process is costly and time-consuming. A capable virtual driver, combined with precise car simulations, could significantly speed up setup development. This paper proposes a data-driven predictive control approach, Data-enabled Predictive Control (DeePC), for trajectory tracking in racing simulations. We compare our approach against an industry-standard Proportional-Integral-Derivative controller and a state-of-the-art Model Predictive Control controller, demonstrating that our method is feasible and yields substantial performance improvements, particularly when trajectories approach the car's physical limits.

Keywords: Machine Learning · Model Predictive Control · Autonomous Racing · DeePC · Trajectory Tracking · Racecar Simulation

1 Introduction

Professional racing teams heavily rely on simulators to refine car setups before events, significantly reducing track-time costs and environmental impacts. The relevance of simulation in motorsport has been growing steadily over the years, with teams investing millions of dollars to get ahead of their competitors[4]. A better simulation infrastructure and high correlation between simulator and track

⋆ This work is supported by the project "Ottimizzazione computazione di un modello di simulazione dinamica del veicolo" funded by the European Union - NextGenerationEU and Dallara Automobili S.p.A. through the Italian "National Recovery and Resilience Plan" (PNRR) Mission 4, Component 2, Investment 3.3 (DM 352/2022) - CUP J33C22001400009 and the Swedish Research Council (VR) with the PSI project (No. #2020-05094)

[4] AMG Petronas Formula One Team—How Does F1 Simulation Work? https://archive.is/mrNjx, archived on 2024-04-02

can lead to significant gains: the base car setup can be found early, shrinking the search space and thus more effectively exploiting the free practice sessions to refine details. Increasing cost, environmental concerns, and regulatory constraints further amplify the role of simulation in motorsport [8].

Simulation at a professional level can be distinguished in two main categories:

1. *Driver-in-Loop (DiL)* simulators, where a human drives a model of the car;
2. *Virtual driver* simulators, where the car is driven by a computer program.

The requirement of a professional racing driver to be physically present in the simulator limits the extent to which DiL simulators can be used; on the other hand, accurate simulation of the driver's behavior (a task akin to programming virtually driven cars) is paramount to obtain valuable feedback.

In this work, we focus on the virtual driver simulator case, and, specifically, on the construction of a control system capable of tracking an ideal trajectory (e.g., coming from a real-world lap), for which we propose a data-driven predictive control approach. We compare our proposed approach with an industrial standard Proportional-Integral-Derivative (PID) controller and with academic state-of-the-art Model Predictive Control (MPC).

1.1 Problem statement

Developing a virtual driver involves: (i) provided the track limits and car data, *generate an optimal reference trajectory*; and (ii) *compute the control inputs* to accurately follow it. Here we focus on the second task. In particular, we investigate the usage of data-driven control techniques for autonomous racing.

1.2 Contribution

This paper extends our previous work [17], where we introduced a data-driven predictive control approach for trajectory tracking in racing car simulation. In this extended version, we expand our analysis of the method including:

– the impact of key hyperparameters (prediction horizon and dataset size) on the controller's tracking performance;
– the controller's behavior on synthetic trajectories with progressively shorter lap times, evaluating its response when the reference becomes infeasible.

2 Related Work

The problem of tracking a reference trajectory is a common control challenge. Specifically, the control algorithm relies on feedback mechanisms to adjust the vehicle inputs in real-time.

A standard approach to trajectory tracking in virtual driver simulations and autonomous driving leverages PID controllers [5]. However, PID controllers lack anticipatory control capabilities.

MPC improves by predicting future states based on a model of the system dynamics [4], producing better driving performance [12], and as such becoming increasingly used in autonomous driving [2]. However, the improved performance comes at the cost of increased algorithmic complexity, as MPC requires a detailed and well-calibrated model of the system dynamics to make informed predictions, which can be time-consuming and resource-intensive [10].

Learning-Based Model Predictive Control (LBMPC) methods overcome model-building complexity by deriving dynamics directly from data. They are classified in three families [6], based on what is learned: (i) **system dynamics**: the algorithm improves its model of reality, either during operation or across multiple operational instances; or (ii) **controller design**: control parameters are adjusted to improve performance; or (iii) **constraints**: optimization and constraint satisfaction are decoupled, with MPC used to enforce constraints [9]. The approach closest to that proposed in this work belongs to the first category.

Some approaches treat the underlying system as a black box, using statistical methods to build a system model *implicitly* from data. Examples include methods based on deep neural networks [15] and statistical models (e.g., NARX [11]). DeePC [13] is part of this family. When real-world data is available, this approach can learn the input-output relationships. DeePC has shown promising in handling the nonlinear dynamics inherent in racing environments [3]. With sufficiently large datasets, DeePC can adapt to complex and dynamic scenarios, improving flexibility and robustness compared to traditional MPC approaches [7].

3 Proposed approach

In this work, we investigate the use of DeePC to control a simulated racing vehicle, comparing it with a PID controller and a "classic" MPC controller.

3.1 DeePC

DeePC is a data-driven variation of MPC where the explicit model is substituted by a dataset. The intuition behind DeePC is that a linear combination of past realizations of the system could be used to match the current conditions and forecast future behavior. More formally, DeePC requires a dataset $\mathbf{H} \in \mathbb{R}^{\mathbf{N}\times\mathbf{2T}}$ of N past realizations of the system capturing T samples of the system state, namely T inputs $\mathbf{u}_t^s$ (input at time t of the s-th trajectory), and T outputs $\mathbf{y}_t^s$ (output at time t of the s-th trajectory).

Inputs and outputs are split into two subgroups each, representing the initial and the future state of the system, used respectively to match the current conditions and to forecast the future behavior. More formally: $\mathbf{u}_{\text{init}}$ (inputs selected to model the initial part of the trajectory across all N trajectories), $\mathbf{y}_{\text{init}}$ (outputs selected to model the initial part of the trajectory across all N trajectories), $\mathbf{u}_{\text{fut}}$ (inputs selected to model the future part of the trajectory across all N trajectories), and $\mathbf{y}_{\text{fut}}$.

Based on such dataset, the algorithm has to find a vector of coefficients $\mathbf{g}$, future predicted inputs $\mathbf{u}_{\text{fut}}$, and future predicted outputs $\mathbf{y}_{\text{fut}}$ such that: (i) there is no difference between the vector of previously known inputs $\mathbf{u}^*_{\text{init}}$ and outputs $\mathbf{y}^*_{\text{init}}$; (ii) the difference between the reference trajectory $\mathbf{r}$ and the predicted outputs $\mathbf{y}_{\text{fut}}$ is minimized, thus minimizing the amount of control exerted on the system (the norm of predicted inputs $\mathbf{u}_{\text{fut}}$). Formally, the optimization problem thus can be written as:

$$\begin{aligned} \underset{\mathbf{g},\mathbf{u}_{\text{fut}},\mathbf{y}_{\text{fut}}}{\text{minimize}} \quad & \sum \|\mathbf{y}_{\text{fut}} - \mathbf{r}\|_Q^2 + \|\mathbf{u}_{\text{fut}}\|_R^2 \\ \text{subject to} \quad & \mathbf{Hg} = \mathbf{L} \end{aligned} \tag{1}$$

where Q and R are weighting matrices that balance the trade-off between tracking performance and control effort. Optimization constraints can be expanded as:

$$\underbrace{\begin{pmatrix} \mathbf{u}^1_{\text{init}} & \mathbf{u}^2_{\text{init}} & \cdots & \mathbf{u}^N_{\text{init}} \\ \mathbf{y}^1_{\text{init}} & \mathbf{y}^2_{\text{init}} & \cdots & \mathbf{y}^N_{\text{init}} \\ \mathbf{u}^1_{\text{fut}} & \mathbf{u}^2_{\text{fut}} & \cdots & \mathbf{u}^N_{\text{fut}} \\ \mathbf{y}^1_{\text{fut}} & \mathbf{y}^2_{\text{fut}} & \cdots & \mathbf{y}^N_{\text{fut}} \end{pmatrix}}_{\mathbf{H}\in\mathbb{R}^{\mathbf{2T}\times\mathbf{N}}} \underbrace{\begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_N \end{pmatrix}}_{\mathbf{g}\in\mathbb{R}^{\mathbf{N}}} = \underbrace{\begin{pmatrix} \mathbf{u}^*_{\text{init}} \\ \mathbf{y}^*_{\text{init}} \\ \mathbf{u}_{\text{fut}} \\ \mathbf{y}_{\text{fut}} \end{pmatrix}}_{\mathbf{L}\in\mathbb{R}^{\mathbf{2T}}} \tag{2}$$

Table 1 summarizes all symbols used in this section.

Linear relaxation of non-linear systems DeePC is working with the assumption that the controllable system is linear. If it's not the case, then non-linearities are accounted for by modelling them as noise through a so-called "discrepancy term" σ [3]. The problem can then be rewritten as:

$$\begin{aligned} \underset{\mathbf{g},\mathbf{u}_{\text{fut}},\mathbf{y}_{\text{fut}},\sigma}{\text{minimize}} \quad & \sum \|\mathbf{y}_{\text{fut}} - \mathbf{r}\|_Q^2 + \|\mathbf{u}_{\text{fut}}\|_R^2 + \|\sigma\|^2_{\lambda_y} + \|\mathbf{g}\|^2_{\lambda_g} \\ \text{subject to} \quad & \mathbf{Hg} = \mathbf{L} + \begin{pmatrix} 0 \ \sigma \ 0 \ 0 \end{pmatrix}^\top \end{aligned} \tag{3}$$

Configuration of the algorithm The algorithm can be configured with two key parameters representing the dataset size (N) and the prediction horizon (ph). We expect larger N to provide faster convergence and better overall performance at the expense of increased computational complexity. Instead, ph determines how far into the future the controller should look, and it is equivalent to the length of future input and output traces ($ph = |\mathbf{u}_{\text{fut}}| = |\mathbf{y}_{\text{fut}}|$). Selecting an appropriate prediction horizon impacts performance, as short horizons will may be insufficient to apply corrective actions, and performance will degrade for long horizons, as the learned behavior will slowly drift away from reality.

3.2 DeePC for simulated racing car control

In this section, we apply DeePC to the problem of controlling a simulated racecar. We discuss the control parameters, the simplifications we introduced, and how to extend the model to account for greater complexity. We assume our controller acts on two inputs: acceleration control (throttle and brake) and steering.

Table 1. Summary of the symbols used in Section 3

Parameter	**Description**
$\mathbf{g}$	Vector of coefficients for trajectories in dataset
$\mathbf{H}$	Matrix of dataset for DeePC
N	Number of trajectories in the dataset
T	Length of the trajectory in the dataset
$\mathbf{u}$	Input vector of the model
$\mathbf{y}$	Output vector of the model
Q	Cost weight matrix of output
R	Cost weight matrix of input
$\mathbf{r}$	Reference trajectory
λ_y	Cost weight matrix of discrepancy
λ_g	Cost weight of DeePC coefficients
σ	Discrepancy vector of non-linearities

Throttle and brake The acceleration control varies in a $[-1, 1]$ where negative values represent actions on braking, and positive values throttle. Thus, this model does not account for combined brake/throttle actions. Moreover, we assume that the car can always output its maximum torque, and thus we do not add gearshift control as an input. Both separated brake and throttle control and gear shifting can be included straightforwardly in the model, as the input and output spaces are not constrained by the algorithm.

Steering steering is modeled as the angle between centerline of the vehicle and centerline of the front wheels, ranging in $[-15°, +15°]$; we thus model the input provided to wheels rather than the input to be provided to the steering wheel.

Outputs The model's outputs are the following: (i) horizontal and (ii) vertical coordinates of vehicle's center of mass with respect to a global reference point (m) (iii) speed ($^{m}/_{s}$) (iv) heading angle with respect to a global coordinate directed towards east (rad). Thus, in total, the algorithm is subject to six channels: two inputs (every entry in $\mathbf{u}_{\text{fut}}$ and $\mathbf{u}_{\text{init}}$ will have two values) and four outputs (every entry in $\mathbf{y}_{\text{fut}}$ and $\mathbf{y}_{\text{init}}$ entry will have four values).

Dataset size and generation DeePC requires a dataset of past realizations of the system to be able to predict future behavior. The quality of such dataset is paramount for the performance of DeePC. The dataset size N is tightly bound to the complexity of the model at hand. Constrained models generally require smaller datasets than less constrained and more complex models. For the problem at hand, even using our simplified racecar model (cf. Section 4.1), a dataset in the order of hundreds of trajectories is necessary. Thus, we run multiple instances of the racecar model, starting in the origin with a random speed, and record the car's paths in response to random control actions. We control the random generation by setting a different seed for each run to ensure reproducibility.

Dataset roto-translation Trajectories must share the same reference frame, requiring roto-translation of either the dataset or the current trajectory. Here, we roto-translate the current trajectory into the dataset's frame.

4 Experimental evaluation

In this section, we investigate: 1. how robust is DeePC to changes in the prediction horizon *ph* and dataset size N; 2. how DeePC compares to "classic" MPC and traditional PID controller in terms of tracking performance.

4.1 Experimental setup

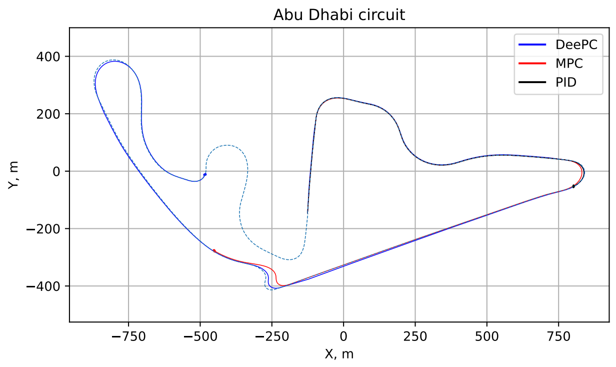

Fig. 1. Trajectory tracking in the Yas Marina circuit in Abu Dhabi. The dashed line defines the target trajectory, solid lines the trajectories as followed by the controllers. The controllers were initialized with time offsets to avoid overlap, which results in portions of the dashed line not being covered by any controller at a given time.

Metrics In this evaluation, we employ the Residual Sum of Squares (RSS) method to gauge how accurately a vehicle follows the reference trajectory. Lower RSS values indicate better tracking performance.

Reference trajectories We test our algorithms on two different trajectory types: (i) a synthetic ∞-shaped Lissajous figure (Figure 5); and (ii) a real-world trajectory of an open-wheel racecar in Yas Marina, Abu Dhabi (Figure 1).

The Lissajous figure has been previously utilized in the literature [1] as a standard reference trajectory for evaluating the performance of trajectory tracking systems. It can be used to generate slower or faster trajectories for a given lap time τ:

$$\begin{cases} x(t) = 100\sin\left(\frac{2\pi t}{\tau}\right) \\ y(t) = 100\cos\left(\frac{4\pi t}{\tau}\right) \end{cases} \qquad \text{for } t \in [0, \tau] \tag{4}$$

The Yas Marina circuit is a track where several major racing championships take place, including Formula 1. We use the real-world trajectory of the track to evaluate the performance of the algorithms in a more realistic scenario, with a variety of high and low-speed corners.

Table 2. Parameters of the racecar model

Param.	Description	Value
D_{RF}	Peak value of the friction coefficient (Pacejika D)	1.0
C_{RF}	Shape factor of the friction coefficient (Pacejika C)	1.1
B_{RF}	Stiffness factor of the friction coefficient (Pacejika B)	25.0
m	Racecar mass	896 kg
I_z	Moment of inertia	1500 $kg \cdot m^2$
l_{RF}	Distance from Center of mass (CM) of Rear/Front wheel	1.125 m
ρ	Air density	1.225 kg/m^3
A	Cross sectional area	
C_D	Drag coefficient	$C_D A = 1.35 m^2$
C_L	Downforce lift coefficient	$C_L A = 4.31 m^2$
P	Motor power	620 hp

Racecar model For this initial study, we use a simplified model of a racecar based on the well-known "single-track model", in which the car is approximated as a two-wheeled vehicle with a fixed wheelbase, and the motion is described in terms of a few key parameters summarized in Table 2. In the model we consider nonlinear tire forces based on a simplified Pacejka model [14].

The state of the model is described by: (i) x, y: position of CM, (ii) ϕ: orientation of the model, (iii) V_x, V_y: velocity of CM, (iv) r: angular velocity of the model, (v) δ: steering angle of the front wheel, and (vi) Ξ: throttle; while its motion is governed by the differential equation:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \\ \dot{V_x} \\ \dot{V_y} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} V_x \cos(\phi) - V_y \sin(\phi) \\ V_x \sin(\phi) + V_y \cos(\phi) \\ r \\ \frac{1}{m}(F_x - F_{Fy}\sin(\delta) - F_{Ax}) + V_y r \\ \frac{1}{m}(F_{Ry} + F_{Fy}\cos(\delta) - F_{Ay}) - V_x r \\ \frac{1}{I_z}(F_{Fy} l_F \cos(\delta) - F_{Ry} l_R) \end{bmatrix} \tag{5}$$

where the *tyres*' lateral forces are represented by F_{Fy} and rear F_{Ry}; the *aerodynamic* drag forces are represented by F_{Ax} and F_{Ay} (respectively, front and lateral[5]); and the *motor* force is represented by F_x. Tyre forces are computed as: $F_{RFy} = D_{RF} \sin\left(C_{RF} \operatorname{atan}\left(B_{RF}\alpha_{RF}\right)\right) W$ and $\alpha_{RF} = \operatorname{atan}\left(\frac{V_y - l_R r}{V_x}\right)$, where the weight ($W$) is computed as: $W = mg + \frac{1}{2}\rho A C_L \|V\|^2$. The aerodynamic drag forces are: $F_{Axy} = \frac{1}{2}\rho A C_D V_{xy}^2$.
Finally, the motor force is: $F_x = \min(\max(\Xi P V_x, -x_{WD_R}), x_{WD_R})$.

Car parameters For the experiments, we set the parameters of the car model to mimic a Formula 2 car[6] Their values are summarized in Table 2.

Optimization criteria for DeePC As optimization criterias we used $R = \operatorname{diag}(0.1, 0.1)$, $Q = \operatorname{diag}(1, 1, 1, 100)$, $\lambda_y = \operatorname{diag}(200, 200, 200, 200)$, $\lambda_g = N/20$. Optimization criteria R and Q (for inputs and outputs, respectively) are chosen to balance the trade-off between tracking performance and control effort. Unlike MPC, DeePC has two additional parameters, λ_y and λ_g, that balance the trade-off between the uncertainty of initial conditions and the predicted trajectory based on the dataset. If $Q \ll R$, the controller deviates from the reference; while if $Q \gg R$, the system can become unstable. Similarly, if $\lambda_y \ll \lambda_g$ DeePC tends to ignore initial conditions in favor of making predictions, which can cause tracking deviations; opposedly, if $\lambda_y \gg \lambda_g$ the control becomes more random, decreasing control quality and possibly causing instability.

4.2 Baselines

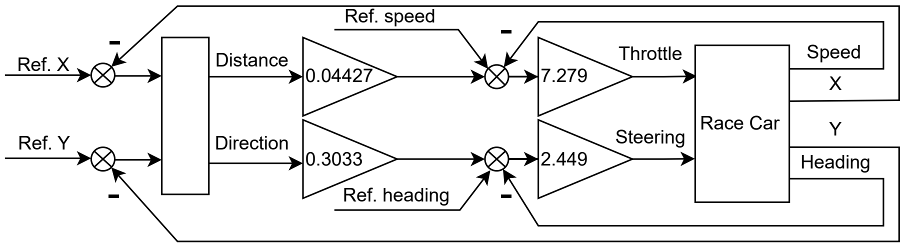

Fig. 2. PID controller scheme.

We compare DeePC with two baselines: (i) a PID controller, and (ii) a "classic" MPC controller. In our paper, we employ the cascade scheme of four propor-

[5] Lateral aerodynamic forces are only relevant when the car slides.
[6] The parameters have been extrapolated with the help of domain experts from the data available at `https://archive.fo/3erTM`

tional controllers depicted in Figure 2, which together compose a PID controller. The inner loops are responsible for controlling the steering and throttle based on the reference speed and heading. The outer loops integrate these inner loops and are responsible for direction and distance control. The parameters have been found using random search optimization.

The MPC controller, instead, is based on a simpler *kinematic* model of the car, which does not consider friction. This way, we replicate the imperfect correlation between the real world and its simulated model.

Experiments and reproducibility We perform three experiments, meant respectively to: 1. evaluate the performance of DeePC with different choices of prediction horizon and dataset size; 2. compare the performance of DeePC with the state of the art on an artificial trajectory with varying lap times; and 3. compare the performance on a real-world trajectory of a race car to gather evidence of whether the approach can be applied to realistic scenarios.

The former two experiments have been opensourced[7] for reuse and reproducibility. An archival copy is also available at Zenodo [16]. The latter experiment could not be included, as the Yas Marina track trajectory is proprietary.

4.3 Impact of dataset size and prediction horizon on DeePC

As a first step, we investigated how the variation of prediction horizon and dataset size influenced the performance of DeePC. To do so, we generated a synthetic ∞-shaped trajectory largely within the capability of the car. We then let the prediction horizon range in $[20ms, 160ms]$ and we inspect the behavior of DeePC with dataset sizes $N \in \{50, 100, 200, 400\}$. For each combination of parameters, we run 30 repetitions differing by the seed used to generate the dataset.

Results are summarized in Figure 3. The controller's ability to foresee into the future correlates positively with the overall quality of control. The best control quality is achieved at about $60ms$, however, this value may be different for different trajectories and car models.

4.4 Synthetic trajectory tracking

Next, we compare PID, MPC and DeePC on a synthetic ∞-shaped trajectory, with the goal of understanding how well they can cope with an increasingly difficult trajectory. In fact, we can apply Equation (4) to generate trajectories increasingly difficult to track, as the lap time τ decreases. In this case, we set the prediction horizon to $60ms$ for both MPC and DeePC.

The experiment results are depicted in Figure 4, and show that the PID controller performs comparably to the predictive controllers for very easy trajectories; however, it quickly becomes the worst performer as the trajectory difficulty

[7] `https://github.com/Crylab/DeePC`

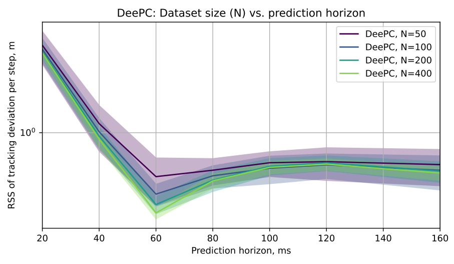

Fig. 3. DeepC performance with dataset size and prediction horizon. Solid lines represent the mean over 30 repetitions, while coloured shaded areas depict ± one standard deviation.

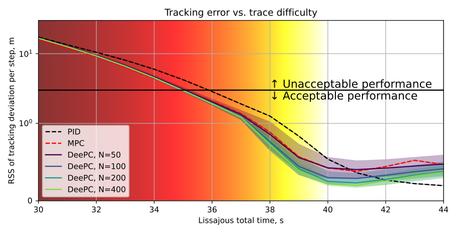

Fig. 4. Error with increasingly easy synthetic trajectories. When on the verge of the car capabilities, DeePC outperforms both baselines. Solid lines are averaged over 30 repetitions, coloured shaded areas depict ± one standard deviation. Acceptable performance is under 7 meters.

increases. MPC has better performance than the PID, and it is comparable to DeePC with $N = 50$. With a larger dataset, DeePC outperforms the baselines, especially for trajectories that sit close to the limit of the car performance.

Moreover, we found that the classic MPC approach is more sensitive to the precision of the numerical solver than DeePC: we encountered cases in which the resulting behavior is unstable for the same selection of optimization criteria for which DeePC is capable to follow the trajectory instead. An example is shown in Figure 5, in which MPC accelerates too early and loses control of the car after a classic "pendulum" effect.

4.5 Real-world trajectory tracking

The last experiment is akin to the previous one, but the trajectory is a real-world lap of a racecar around the track of Yas Marina in Abu Dhabi. In this experiment we make the lap increasingly more difficult to follow by varying the peak friction coefficient of the tires, emulating the effect of an increasingly slippery track[8]. We analyze conditions that range from a slippery track in which the lap is plainly not achievable ($D_{RF} = 0.8$), to conditions of extremely high grip ($D_{RF} = 1.6$).

The resulting data is summarized in Figure 6. In this test, with a much more complex trajectory featuring slow and fast turns, DeePC outperforms both baselines consistently. In this case, in fact, the simplified kinematic model of MPC is not capable to accurately predict the car's behavior in many conditions, while the data-based model internal to DeePC can produce more accurate predictions.

[8] Track conditions impact significantly the grip available to the car in real-world racing. Even if the track remains dry, the rubber laid down by the cars can make it more or less slippery, and the track temperature can also affect the grip. Moreover, the tire compound can be chosen to be more or less grippy, typically presenting a trade-off between peak performance and durability.

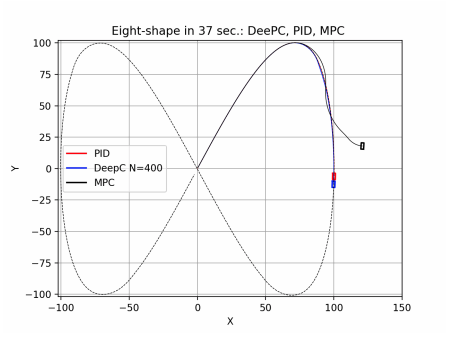

Fig. 5. ∞-shaped trajectory tracking with unstable MPC

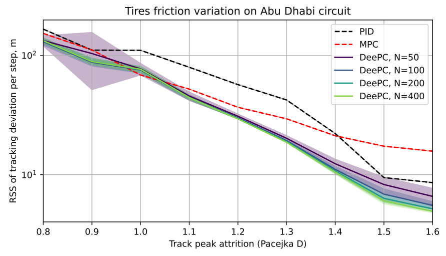

Fig. 6. Error with increasingly higher peak tyre friction in the Yas Marina circuit. DeePC outperforms both baselines. Solid lines are averaged over 30 repetitions, coloured shaded areas depict ± one standard deviation.

5 Conclusion and future work

In this paper, we extended our previous investigation of DeePC for trajectory tracking in race car simulation, investigating the impact of prediction horizon and dataset size on the controller's tracking performance and the controller's behavior on synthetic trajectories with progressively shorter lap times. We compared it against traditional PID and classical MPC controllers. Our results show DeePC is viable, outperforming traditional methods especially on complex tracks near the vehicle's limits. Its data-driven nature makes DeePC suitable even when building accurate real-world models is challenging, provided the dataset is sufficiently rich and the prediction horizon appropriately balanced.

Future work will extend the approach to advanced car models (including multibody dynamics, finite element tire models, and aerodynamics), improve dataset generation for greater efficiency, and tackle the separate problem of trajectory generation based on track and car characteristics, as discussed in Section 1.1. Moreover, we will more precisely characterize the efficiency profile of DeePC compared to MPC. In our experiments, execution times were comparable between the two methods; however, we expect DeePC to scale better than MPC with model complexity and prediction horizon. This expectation will be verified in future work.

References

1. Bini, E., Papadopoulos, A.V., Higgins, J., Bezzo, N.: Optimal reference tracking for sampled-data control systems. In: 61st IEEE Conference on Decision and Control. pp. 7141–7148 (2022). `https://doi.org/10.1109/cdc51059.2022.9992462`
2. Choi, W.Y., Lee, S.H., Chung, C.C.: Horizonwise model-predictive control with application to autonomous driving vehicle. IEEE Trans. Industrial Informatics **18**(10), 6940–6949 (2022). `https://doi.org/10.1109/tii.2021.3137169`

3. Coulson, J., Lygeros, J., Dorfler, F.: Data-enabled predictive control: In the shallows of the deepc. In: 18th ECC. pp. 307–312 (2019). https://doi.org/10.23919/ECC.2019.8795639
4. Darby, M.L., Nikolaou, M.: MPC: Current practice and challenges. Control Engineering Practice **20**(4), 328–342 (2012). https://doi.org/10.1016/j.conengprac.2011.12.004
5. Farag, W.: Complex trajectory tracking using PID control for autonomous driving. International Journal of Intelligent Transportation Systems Research **18**(2), 356–366 (2019). https://doi.org/10.1007/s13177-019-00204-2
6. Hewing, L., Wabersich, K.P., Menner, M., Zeilinger, M.N.: Learning-based model predictive control: Toward safe learning in control. Annual Review of Control, Robotics, and Autonomous Systems **3**(1), 269–296 (2020). https://doi.org/10.1146/ANNUREV-CONTROL-090419-075625
7. Huang, L., Lygeros, J., Dörfler, F.: Robust and kernelized data-enabled predictive control for nonlinear systems. IEEE Trans. Control Systems Technology **32**(2), 611–624 (2024). https://doi.org/10.1109/tcst.2023.3329334
8. Judde, C., Booth, R., Brooks, R.: Second place is first of the losers: An analysis of competitive balance in formula one. Journal of Sports Economics **14**(4), 411–439 (2013). https://doi.org/10.1177/1527002513496009
9. Koller, T., Berkenkamp, F., Turchetta, M., Krause, A.: Learning-based model predictive control for safe exploration. In: IEEE CDC. pp. 6059–6066 (2018). https://doi.org/10.1109/cdc.2018.8619572
10. Kozak, S.: From PID to MPC: Control engineering methods development and applications. In: K&I 2016 (2016). https://doi.org/10.1109/cyberi.2016.7438634
11. Limon, D., Calliess, J., Maciejowski, J.: Learning-based nonlinear model predictive control. IFAC-PapersOnLine **50**(1), 7769–7776 (2017). https://doi.org/10.1016/j.ifacol.2017.08.1050
12. Liniger, A., Domahidi, A., Morari, M.: Optimization-based autonomous racing of 1:43 scale RC cars. Optimal Control Applications and Methods **36**(5), 628–647 (2014). https://doi.org/10.1002/oca.2123
13. Markovsky, I., Huang, L., Dörfler, F.: Data-driven control based on the behavioral approach: From theory to applications in power systems. IEEE Control Systems **43**(5), 28–68 (2023). https://doi.org/10.1109/mcs.2023.3291638
14. Pacejka, H.B., Bakker, E.: The magic formula tyre model. Vehicle System Dynamics **21**(sup001), 1–18 (1992). https://doi.org/10.1080/00423119208969994
15. Rokonuzzaman, M., Mohajer, N., Nahavandi, S., Mohamed, S.: Model predictive control with learned vehicle dynamics for autonomous vehicle path tracking. IEEE Access **9** (2021). https://doi.org/10.1109/access.2021.3112560
16. Shaiakhmetov, R.: Crylab/deepc: Initial release (2024). https://doi.org/10.5281/ZENODO.11515536, https://zenodo.org/doi/10.5281/zenodo.11515536
17. Shaiakhmetov, R., Pianini, D., Venusti, V., Papadopoulos, A.V.: A data-driven predictive control driver for racing car simulation. In: DS-RT. pp. 140–143. IEEE (2024). https://doi.org/10.1109/DS-RT62209.2024.00033, https://doi.org/10.1109/DS-RT62209.2024.00033