C2SMARTER
CONNECTED COMMUNITIES WITH SMART MOBILITY TO EQUITABLY REDUCE CONGESTION

Physics-informed Machine Learning for Autonomous Vehicle Control

Xianning Li, Kaan Ozbay, and Zhong-Ping Jiang

Physics-informed Machine Learning for Autonomous Vehicle Control*

Xianning Li[1], Kaan Ozbay[2] and Zhong-Ping Jiang[1,2]

***Abstract*— In this paper, a physics-informed machine learning approach is proposed for the lane changing of autonomous vehicles. Combining vehicle longitudinal and lateral control, the lane changing problem is formulated as a nonlinear motion planning and control optimization problem based on model predictive control (MPC). To address the computational challenge in solving the nonlinear MPC problem, a neural network controller is trained based on differential predictive control self-supervised learning framework. Through numerical simulation, the learned controller is compared with a conventional MPC controller. It improves the computational efficiency by 97.36% and provides reduced overshoot of the control inputs.**

***Index Terms*— Lane Changing, Physics-informed machine learning, Differentiable predictive control (DPC)**

I. INTRODUCTION

Autonomous driving technologies, as an important component of intelligent transportation system, hold tremendous potential for improving various types of transportation issues. Lane changing is one of the typical scenarios of autonomous driving, and the design of automatic lane changing involves decision-making, motion planning, and control aspects [1]. Compared to functions such as adaptive cruise control (ACC) and lane keeping, automated lane changing involves both longitudinal and lateral motion planning and control of the vehicle, and therefore the dynamic modeling and control synthesis for the lane changing problem is much more challenging.

Most of the trajectory planning methods for automatic lane changing are based on polynomial curves, geometric curves, or MPC, of which MPC-based methods are widely utilized due to the capability to account for more complex design objectives and constraints, but they are typically used for local planning due to their higher computational cost [2]. Furthermore, in past literature, the longitudinal and lateral planning and control are usually performed separately, or assume the longitudinal velocity as a constant value, in order to reduce the complexity of the MPC problem for real-time control [3], [4]. Nevertheless, the performance of this form of motion planning and control is inferior to that of combining the longitudinal and lateral dimensions [5]. This is the issue that MPC-based optimal control methods frequently face in real-word applications, i.e., the trade-off between performance and computational cost.

*This work has been partially supported by a C2SMARTER grant.

[1]Control and Networks Lab, Department of Electrical and Computer Engineering, Tandon School of Engineering, New York University, Brooklyn, NY 11201 USA `xl5305@nyu.edu; zjiang@nyu.edu`

[2]C2SMARTER Center, Tandon School of Engineering, New York University, Brooklyn, NY 11201 USA `kaan.ozbay@nyu.edu`

With the development of artificial intelligence technology, it brings new opportunities for planning and control. Approximate MPC has been employed in some studies to utilize neural networks to learn explicit control policies, leading to a reduced computational burden in applications [6]. However, Approximate MPC is a supervised learning framework whose performance is limited by the policy dataset, and the learning process can only measure the difference with the existing policy, and cannot directly consider the control objectives and physical constraints, which can lead to learning bias between the learned policy and the theoretical optimal policy [7]. In recent years, physics-informed machine learning has been proposed to tackle physical models and constraints [8], such as the most widely known physics-informed neural network (PINN). As a new approach, it is a deep learning framework primarily used for solving problems described by PDEs [9], and have been proven to be effective in numerous domains, such as fluid mechanics [10], materials [11], thermal systems [12], [13]. Indeed, PINNs integrate PDEs and physical constraints as loss functions into the network training framework, and benefit from this combination of observed data with physical models and constraints with high sample efficiency. Based on the same idea and the development of automatic differentiation techniques, several physics-informed machine learning methods have emerged. DPC is one of the representative methods, which improves on the drawbacks of approximate MPC and proposes an unsupervised learning framework that directly embeds the system dynamics into a neural network, which can be applied to linear systems, nonlinear systems, and systems with unknown parameters [14], [15], [16]. Different from DPC, which uses a neural network to learn control input sequences within the prediction horizon, another method known as AI Pontryagin [17] uses a neural network to learn the current control inputs, with an overall architecture similar to recurrent neural network, sharing the same control policy network at different moments . Currently, there are relatively few real-world applications of physics-informed machine learning, and in autonomous driving, there have been studies applying it to vehicle longitudinal control, e.g., ACC [18] and predictive cruise control [19], but to the best of our knowledge, there has not been any application to combined lateral and longitudinal planning and control of autonomous vehicles.

In this paper, in order to solve the high computational burden suffered by optimization-based combined longitudinal and lateral vehicle motion planning and control under lane changing scenarios, physics-informed machine learning method is adopted to obtain explicit control policies by using neural networks. First, the original problem of lane change motion planning and control is established based on MPC. Then, using DPC, the vehicle model is combined with the control policy network in the form of a computational graph to become a differentiable training network, and the initial conditions and reference values of the MPC problem are adopted as the inputs, and the supervisory signals are generated with the objective function of the MPC problem to form the self-supervised learning framework of the control policy. Next, the trajectories of lane changing are used to generate the training dataset and train the network, and the performance of the learned control policy network is compared with the MPC controller. Summarizing the above, the main contribution of our paper is to initiate a study on physics-informed machine learning for autonomous vehicle control. In particular, it is shown that when formulating the automated lane changing problem as a MPC problem, physics-informed neural networks can be used to generate an explicit lane changing controller with improved computational efficiency.

The rest of the paper is organized as follows. Section II introduces the nonlinear vehicle model and formulates the motion planning and control problem based on MPC. Section III describes a physics-informed machine learning method called DPC that we will use in this paper. Section IV presents a training process for the lane changing control policy and compares it with the conventional MPC controller by means of numerical simulations. Finally, some concluding remarks are given in Section V.

II. PROBLEM STATEMENT

In this section, we first describe a model that captures the vehicle dynamics, and then, formulates the lane changing problem based on the MPC framework.

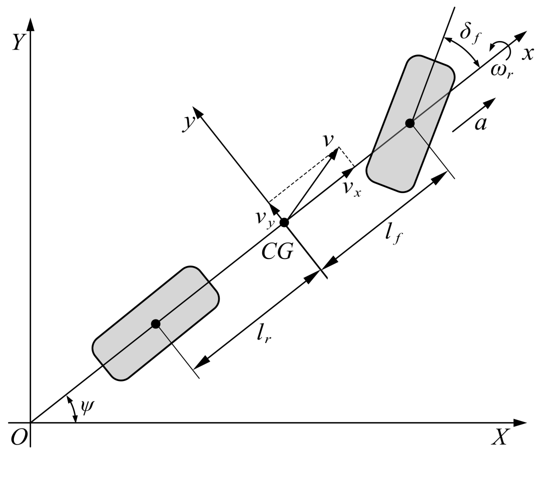

Fig. 1. Nonlinear vehicle model

A. Nonlinear Vehicle Model

Assuming that the steering angle remains small during the lane changing process and that the direction of vehicle acceleration is aligned with its longitudinal velocity, a nonlinear model of the vehicle can be derived based on a two-degree-of-freedom bicycle model. This model is depicted in Figure 1, where the ground coordinate system is denoted by X and Y, while the vehicle coordinate system is denoted by x and y. The vehicle's center of gravity is indicated by CG, with ψ representing the yaw angle. The front and rear axle distances are represented by l_f and l_r, respectively. v_x and v_y indicate the longitudinal and lateral velocities, while a denotes the longitudinal acceleration. δ_f represents the cornering angle, and ω_r indicates the yaw rate. To comprehensively describe the vehicle's longitudinal and lateral dynamics, six states $x = [X, Y, \psi, v_x, v_y, \omega_r]^T$ are selected to describe the system. The longitudinal acceleration and steering angle of the vehicle serve as the control inputs $u = \left[a, \delta_f\right]^T$. Now, the motion of the vehicle can be described by the following ordinary differential equations:

$$\dot{X} = v_x \cos\psi - v_y \sin\psi \tag{1a}$$

$$\dot{Y} = v_x \sin\psi + v_y \cos\psi \tag{1b}$$

$$\dot{\psi} = \omega_r \tag{1c}$$

$$\dot{v}_x = a \tag{1d}$$

$$\begin{aligned}\dot{v}_y = &-\frac{2(C_f + C_r)}{m v_x} v_y \\ &- \left[\frac{2(l_f C_f - l_r C_r)}{m v_x} + v_x\right]\omega_r + \frac{2C_f}{m}\delta_f\end{aligned} \tag{1e}$$

$$\begin{aligned}\dot{\omega}_r = &-\frac{2(l_f C_f - l_r C_r)}{I_z v_x} v_y \\ &- \frac{2(l_f^2 C_f + l_r^2 C_r)}{I_z v_x}\omega_r + \frac{2 l_f C_f}{I_z}\delta_f\end{aligned} \tag{1f}$$

where m is the vehicle mass, I_z is the moment of inertia of the vehicle about the z axis, and C_f and C_r are the cornering stiffness of the front and rear axles, respectively. The above model is a time-invariant nonlinear system denoted as

$$\dot{x}(t) = f\left(x(t), u(t)\right) \tag{2}$$

Using Euler forward discretization, a discrete-time representation of (2) can be obtained as

$$\begin{aligned} x(k+1) &= x(k) + f\left(x(k), u(k)\right)\Delta t \\ &= F\left(x(k), u(k)\right)\end{aligned} \tag{3}$$

Hereafter, $u(k)$ and $x(k)$ are abbreviated as u_k and x_k for convenience.

B. Lane Changing Problem Formulation Based on MPC

Automated lane changing requires considering issues of safety, efficiency and comfort. These issues can be addressed via the cooperation of decision making, motion planning and control modules in autonomous vehicles. This paper assumes that commands received from the decision-making module are safe to execute. Consequently, the motion planning and control of lane changing can be reframed as a control problem for tracking the expected position. If there is a vehicle in front of the target lane, the expected position is set as a target point, ensuring a safe distance from the preceding vehicle [20]. However, this paper does not consider this scenario, and thus the expected position is taken to be the centerline of the target lane, as shown in Fig. 2, where Y_{ref} represents the Y coordinate of the target line, L_w represents the width of a single lane, $v_{x,0}$ represents the initial longitudinal velocity of lane changing, and $v_{x,\text{ref}}$ represents the desired longitudinal velocity after the lane change. In other words, while controlling the longitudinal velocity, this paper's motion control approach does not require the vehicle's longitudinal position, but retains it as a state to introduce more comprehensive safety guarantees in the future.

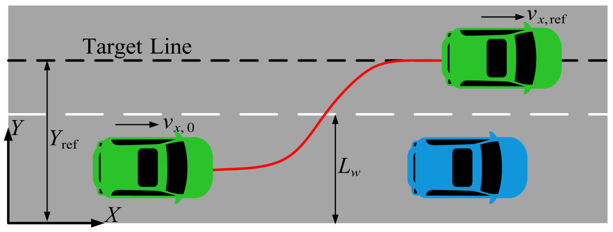

Fig. 2. Lane changing problem description

Based on the above description, the lane changing problem is now formulated as a dynamic optimization problem based on MPC. Notice that, during lane change, the angle of the front wheel changes under piecewise constant controls, and that the number of control steps N_c is the same as the number of prediction steps N_p.

Specifically, we consider the following dynamic optimization problem using a quadratic cost function:

$$\begin{aligned}\min_{U} J_{MPC} = &\sum_{k=0}^{N_p-1} \left(x_k - x_{\text{ref}}\right)^T Q_x \left(x_k - x_{\text{ref}}\right) + u_k^T Q_u u_k \\ &+ \left(x_{N_p} - x_{\text{ref}}\right)^T Q_t \left(x_{N_p} - x_{\text{ref}}\right)\end{aligned} \tag{4}$$

where $U = \left[u_0, u_1, \ldots, u_{N_p-1}\right]$ is the sequence of control inputs in the predictive horizon, x_{ref} is the reference state of the vehicle, x_{N_p} is the terminal state of the vehicle, Q_x is the state tracking penalty matrix, Q_u is the control penalty matrix, and Q_t is the terminal state tracking penalty matrix. The first part of the objective function represents the accumulation of penalties over the predicted horizon. The term $\left(x_k - x_{\text{ref}}\right)^T Q_x \left(x_k - x_{\text{ref}}\right)$ penalizes deviations from the reference state, guiding the vehicle's current state toward it. The term $u_k^T Q_u u_k$ penalizes the control input, acting as a soft constraint to reduce energy consumption and enhance comfort during lane changes. The second part of the objective function, $\left(x_{N_p} - x_{\text{ref}}\right)^T Q_t \left(x_{N_p} - x_{\text{ref}}\right)$, penalizes the terminal state, aiming to bring it closer to the reference state. For the lane-changing scenario shown in Fig. 2, the reference state is

$$x_{\text{ref}} = \left[X_{\text{ref}}, \frac{3L_w}{2}, 0, v_{x,\text{ref}}, 0, 0\right]^T \tag{5}$$

where the reference values for yaw angle, lateral vehicle speed, and yaw rate are all 0, meaning that the vehicle should keep a straight line when completing the lane changing.

For the system (3), the following constraints on the control inputs will be considered:

$$a_{\min} \leqslant a(k) \leqslant a_{\max} \tag{6a}$$

$$\delta_{f,\min} \leqslant \delta_f(k) \leqslant \delta_{f,\max} \tag{6b}$$

These are hard constraints on the control inputs to prevent them from exceeding reasonable limits, which would result in high energy consumption, low comfort, and potential safety risks.

III. PHYSICS-INFORMED MACHINE LEARNING CONTROLLER DESIGN

In this section, we first introduce DPC as a physics-informed machine learning approach for autonomous vehicle control and then present a learning-based algorithm for the design of lane changing controllers.

A. Differential Predictive Control

DPC is a self-supervised learning framework based on direct policy gradient as shown in Fig. 3 [14], [15]. DPC directly connects the control policy network to the system model, forming a differentiable network that takes initial conditions and reference and constraint parameters as the inputs, with outputs used to compute the objective function. The system model can be any differentiable model, such as linear model, nonlinear model, or neural network model. Compared to the original MPC problem, DPC embeds system equation constraints directly into the training network, strictly guaranteeing the relationship between state updates and control inputs, while incorporating other inequality constraints into the training objective function as soft constraints. During forward propagation, the output of the control policy network acts directly on the system model, providing an evaluation of the current policy network. In back propagation, the parameters of the control policy network can be updated, allowing it to learn policies that reduce the value of the objective function.

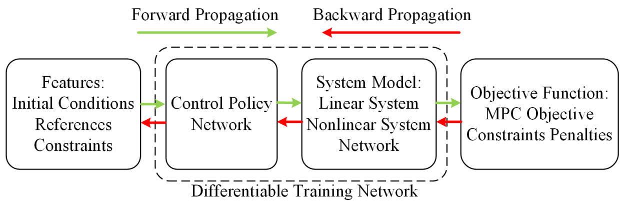
Forward Propagation
Backward Propagation
Features: Initial Conditions References Constraints
Control Policy Network
System Model: Linear System Nonlinear System Network
Objective Function: MPC Objective Constraints Penalties
Differentiable Training Network

Fig. 3. DPC framework

To summarize existing applications of DPC, the DPC framework offers the following key advantages: i) its self-supervised learning framework requires no control policy reference, i.e., labels, ii) it directly considers the effect of the control policy on system dynamics, avoiding deviations between the learned policy and the nominal optimal policy, and iii) its low computational complexity for optimizing the control inputs makes the trained control policy network conducive to real-time control.

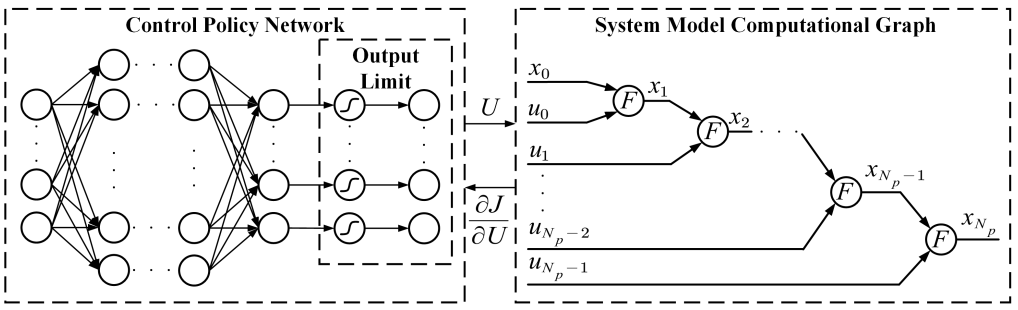

Fig. 4. Lane changing training network construction

B. DPC-based Controller Design

It should be mentioned that the lane changing optimal control problem considered in this paper does not explicitly list any hard constraints on the vehicle state. Instead, a state tracking penalty term is used to monitor the desirable transient behavior of the vehicle state. For constraints on the control inputs, as shown in Fig. 4, the training network in this paper differs from the DPC method by incorporating an output limitation layer [19], thereby achieving hard constraints on the control inputs.

Subsequently, the process of updating the parameters of the control policy network under the self-supervised learning framework in this paper is described. First, forward propagation is performed. Given that the reference state is constant over the predictive horizon, the inputs to the training network, i.e., the features, consist only of the vehicle's initial state and the reference state as follows:

$$\mathbf{x} = \left[x_0^T, x_{\text{ref}}^T\right]^T \tag{7}$$

This prevents the need to repeat inputs from the reference state sequences, as done in existing DPC methods. The control policy network then calculates the control input sequence over the predictive horizon:

$$U = \pi(\theta, \mathbf{x}) \tag{8}$$

where $U = \left[u_0, u_1, ..., u_{N_p-1}\right]$, and θ represents the network parameters. And the tanh function is applied at the output limitation layer, restricting each output to between -1 and 1 before mapping it to the control input constraint space. If an output of the neural network is y_{ij}, which corresponds to u_{ij} in the control input sequence, the output limitation layer performs the following transformation:

$$u_{ij} = \tanh(y_{ij}) \cdot \left(u_{ij,\max} - u_{ij,\min}\right) + \frac{u_{ij,\max} + u_{ij,\min}}{2} \tag{9}$$

Next, the future states within the predictive horizon are computed using the system model's computational graphs:

$$S = G(x_0, U) \tag{10}$$

where $S = \left[x_0, x_1, ..., x_{N_p}\right]$. Then, the training loss is computed as follows:

$$L = J_{MPC}(S, U, x_{\text{ref}}) + \lambda \parallel \theta \parallel_2^2 \tag{11}$$

Considering the variety of vehicle states during lane changes, an additional L2 regularization term $\lambda \parallel \theta \parallel_2^2$ is introduced to improve the network's generalization performance, where λ is a regularization factor.

Following this, backward propagation is performed to compute the gradient of the loss w.r.t. the network parameters and to update them. The chain rule is used for gradient computation:

$$\begin{aligned} \nabla_\theta L &= \frac{\partial J_{MPC}}{\partial \theta} + \frac{\partial \lambda \parallel \theta \parallel_2^2}{\partial \theta} \\ &= \frac{\partial J_{MPC}}{\partial U}\frac{\partial U}{\partial \theta} + \frac{\partial J_{MPC}}{\partial S}\frac{\partial S}{\partial U}\frac{\partial U}{\partial \theta} + \frac{\partial \lambda \parallel \theta \parallel_2^2}{\partial \theta} \end{aligned} \tag{12}$$

The network parameters are then updated as follow:

$$\theta_{k+1} = \theta_k - \eta \nabla_{\theta_k} L \tag{13}$$

where η represents the learning rate. Forward and backward propagations are continuously performed until the set number of training epochs is reached. The process for updating the control policy network parameters is summarized in Algorithm 1.

Algorithm 1 Control Policy Network Parameters Updating

```
1: Forward Propagation: Calculating the loss function.
2: Input a batch of features $\left[x_0^T, x_{\text{ref}}^T\right]^T$.
3: Calculate the control input sequence $U = \pi(\theta, \mathbf{x})$.
4: Calculate the future states $S = G(x_0, U)$.
5: Calculate the loss $L = J_{MPC}(S, U, x_{\text{ref}}) + \lambda \parallel \theta \parallel_2^2$.
6: Backward Propagation: Updating the network parameters.
7: Compute the gradient of the loss function w.r.t. the network parameters by leveraging chain rule as (12).
8: Update network parameters $\theta_{k+1} = \theta_k - \eta \nabla_{\theta_k} L$.
```

IV. SIMULATION RESULTS

In this section, the implementation of the controller from the previous section is described. This includes details on the training dataset generation and training process, as well as a comparison of numerical simulation results between the proposed controller and the MPC controller. The vehicle, road, and controller parameters used in these processes are summarized in Table I, , with the vehicle parameters sourced from [3].

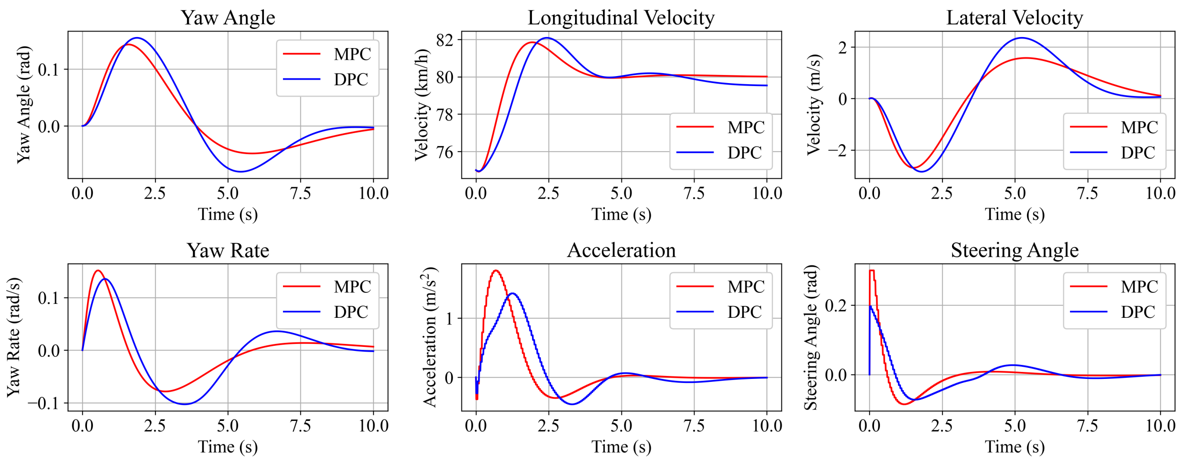

Fig. 5. Comparison of MPC and DPC lane changing process from 75km/h to 80km/h

TABLE I
VEHICLE, ROAD AND CONTROLLER PARAMETERS

Parameter	Value(unit)
m	1270 kg
I_z	1536.7 kg $\cdot$ m^2
l_f	1.015 m
l_r	1.895 m
C_f	1250 N/rad
C_r	755 N/rad
L_w	4 m
$a_{\max}$	3 m/s^2
$a_{\min}$	-3 m/s^2
$\delta_{f,\max}$	0.3 rad
$\delta_{f,\min}$	-0.3 rad
N_p	10
Δt	0.5 s
Q_x	diag $[0,20,500,10,1,100]$
Q_u	diag $[5,500]$
Q_t	diag $[0,20,1000,20,2,200]$
λ	0.2

A. Dataset Generation and Training Process

Considering the similarity in vehicle operation patterns across different lane-changing processes, multiple states from various lane-changing trajectories are utilized as the training set. This helps avoid an oversized dataset and prevents network performance degradation by excluding states that should not appear in the lane-changing process, which would deviate from the actual scenario. During implementation, it was found that using only lane-changing trajectories where the initial vehicle is parallel to the lane results in insufficient data richness for lateral control. Therefore, a total of 6300 lane-changing trajectories with varying initial Y coordinates, initial vehicle speeds, initial yaw angles, reference Y coordinates, and reference vehicle speeds, comprising 315,000 samples, are used as the training set for the control policy network. The control policy network is then trained. This paper uses a fully connected neural network with three hidden layers, each with 256 nodes. The training process consists of 1,000 epochs with a learning rate of 0.0001, and data is loaded 10,000 samples per batch.

B. Numerical Simulation of the Closed-loop System

After training, the learned DPC controller is compared with a conventional MPC controller via Python-based numerical simulation. The MPC controller uses the same parameters as the DPC controller, and the nonlinear MPC problem is solved by the efficient solver CasADi [21]. The simulation uses a step size of 0.01 seconds and a control period of 0.05 seconds, running on a computer with an i7 CPU @ 2.30GHz.

A comparison of changes in vehicle state and control inputs between the MPC and DPC controllers for a vehicle accelerating from 75km/h to 80km/h during a lane change is shown in Fig. 5, and a comparison of vehicle position changes per second is shown in Fig. 6. Key performance indicators are recorded in Table II.

TABLE II
SIMULATION RESULTS

Controller	MPC	DPC
v_x Average	79.98 km/h	79.77 km/h
$\lvert a\rvert$ Maximum	1.81 m/s^2	1.42 m/s^2
δ_f Maximum	0.30 rad	0.20 rad
Calculation Time	17.78 ms	0.47 ms

In terms of vehicle trajectory, the DPC controller's control effect is roughly equivalent to that of the MPC controller. For changes in states and control inputs over time, the vehicle with the DPC controller converges faster in the lateral direction, but in the longitudinal direction, there is a small steady-state error between the actual speed and the reference speed. The average longitudinal speeds under the two controllers are 79.98km/h and 79.77km/h, respectively, with a difference of only 0.26%, which has negligible effect on the overall lane-changing process. Moreover, the overshoots as seen from the maximum absolute acceleration and steering angle values have been reduced by 21.55% and 33.33%, respectively, thanks to the output limitation layer in the control strategy network. In terms of computational speed, the DPC

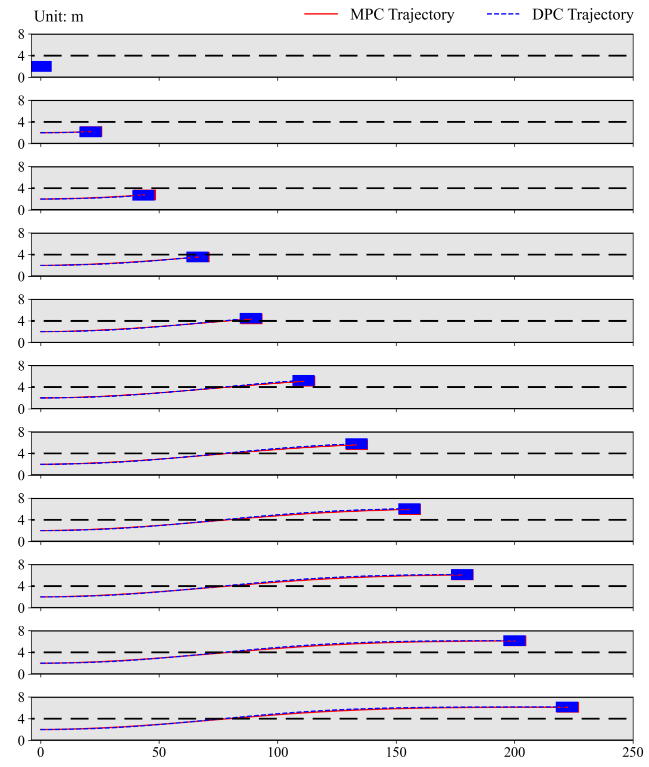

Fig. 6. Comparison of MPC and DPC lane changing process per second

controller's average calculation speed is 97.36% faster than the MPC controller. Overall, the DPC controller reduces the overshoot of the control inputs and offers extremely high computational efficiency while maintaining comparable control performance.

V. Summary and Future Work

In this paper, the autonomous lane changing problem is studied from the perspective of physics-informed machine learning. In particular, DPC is extended to address the combined longitudinal and lateral control of autonomous vehicles with nonlinear models. It is shown through numerical simulation that the learned controller based on DPC compares favorably as the previously developed lane changing MPC controller. The DPC-based vehicle lane changing controller leads to improved computational efficiency by 97.36% and provides reduced overshoot of the control inputs through the introduction of an output limitation layer in the control policy network.

Our future work will be directed at analyzing the stability of the closed-loop vehicle system with the proposed learning control policy. In addition, more advanced neural networks will be considered to learn control policies rather than only relying on feedforward neural networks, in order to enable more complex autonomous driving maneuvers and improve the learning efficiency of the sample datasets.

References

[1] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixao, F. Mutz *et al.*, "Self-driving cars: A survey," *Expert systems with applications*, vol. 165, p. 113816, 2021.

[2] C. Ma and D. Li, "A review of vehicle lane change research," *Physica A: Statistical Mechanics and its Applications*, vol. 626, p. 129060, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378437123006155

[3] Y. Zhong, L. Guo, Y. Zhang, Q. Liu, and H. Chen, "Optimal lane change control of intelligent vehicle based on mpc," in *2019 Chinese Control And Decision Conference (CCDC)*. IEEE, 2019, pp. 1468–1473.

[4] W. Tang, M. Yang, C. Wang, B. Wang, L. Zhang, and F. Le, "Mpc-based path planning for lane changing in highway environment," in *2018 Chinese Automation Congress (CAC)*. IEEE, 2018, pp. 1003–1008.

[5] L. Cui, K. Ozbay, and Z.-P. Jiang, "Combined longitudinal and lateral control of autonomous vehicles based on reinforcement learning," in *2021 American control conference (ACC)*. IEEE, 2021, pp. 1929–1934.

[6] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer, "Learning an approximate model predictive controller with guarantees," *IEEE Control Systems Letters*, vol. 2, no. 3, pp. 543–548, 2018.

[7] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 712–733, 2020.

[8] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021.

[9] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.

[10] A. M. Tartakovsky, C. O. Marrero, P. Perdikaris, G. D. Tartakovsky, and D. Barajas-Solano, "Physics-informed deep neural networks for learning parameters and constitutive relationships in subsurface flow problems," *Water Resources Research*, vol. 56, no. 5, p. e2019WR026731, 2020.

[11] E. Zhang, M. Dao, G. E. Karniadakis, and S. Suresh, "Analyses of internal structures and defects in materials using physics-informed neural networks," *Science advances*, vol. 8, no. 7, p. eabk0644, 2022.

[12] S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks for heat transfer problems," *Journal of Heat Transfer*, vol. 143, no. 6, p. 060801, 2021.

[13] G. Gokhale, B. Claessens, and C. Develder, "Physics informed neural networks for control oriented thermal modeling of buildings," *Applied Energy*, vol. 314, p. 118852, 2022.

[14] J. Drgona, A. Tuor, and D. Vrabie, "Learning constrained adaptive differentiable predictive control policies with guarantees," *arXiv preprint arXiv:2004.11184*, 2020.

[15] J. Drgoňa, K. Kiš, A. Tuor, D. Vrabie, and M. Klaučo, "Differentiable predictive control: Deep learning alternative to explicit model predictive control for unknown nonlinear systems," *Journal of Process Control*, vol. 116, pp. 80–92, 2022.

[16] W. S. Cortez, J. Drgona, D. Vrabie, and M. Halappanavar, "Robust differentiable predictive control with safety guarantees: A predictive safety filter approach," *arXiv preprint arXiv:2311.08496*, 2023.

[17] L. Böttcher, N. Antulov-Fantulin, and T. Asikis, "AI Pontryagin or how artificial neural networks learn to control dynamical systems," *Nature communications*, vol. 13, no. 1, p. 333, 2022.

[18] S. Yang, S. Chen, V. M. Preciado, and R. Mangharam, "Differentiable safe controller design through control barrier functions," *IEEE Control Systems Letters*, vol. 7, pp. 1207–1212, 2022.

[19] W. Shi, X. Luo, J. Hong, C. Zhao, B. Gao, and H. Chen, "Accelerating model predictive control with neural network optimizer," in *2023 7th CAA International Conference on Vehicular Control and Intelligence (CVCI)*. IEEE, 2023, pp. 1–7.

[20] S. Chakraborty, L. Cui, K. Ozbay, and Z.-P. Jiang, "Automated lane changing control in mixed traffic: An adaptive dynamic programming approach," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2022, pp. 1823–1828.

[21] J. A. E. 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.