MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Physics-informed Machine Learning with Heuristic Feedback Control Layer for Autonomous Vehicle Control

Li, Xianning; Wang, Yebin; Ozbay, Kaan; Jiang, Zhong-Ping

TR2025-087 June 24, 2025

Abstract

This paper proposes a novel physics-informed ma- chine learning framework for motion planning and control of autonomous vehicles. By integrating longitudinal and lateral control, a nonlinear control problem is formulated using Model Predictive Control (MPC). To address computational challenges, a self-supervised framework, Recurrent Predictive Control (RPC), is introduced, leveraging differentiable neural networks and recurrent neural networks to train a neural network controller. Additionally, a heuristic feedback control layer is designed to reduce steady-state errors in the closed-loop tracking. Through numerical simulations and co-simulations using Simulink and CarSim, five neural network controllers are compared with an MPC controller in a lane-changing scenario. The proposed RPC framework improves computational efficiency by 95.91% com- pared to MPC, enhances generalization performance compared to Approximate MPC, and reduces performance loss by 17.01% compared to Differentiable Predictive Control. The heuristic feedback control layer further reduces steady-state errors and improves convergence speed during training.

IEEE Intelligent Vehicles Symposium (IV) 2025

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Physics-informed Machine Learning with Heuristic Feedback Control Layer for Autonomous Vehicle Control

Xianning Li
New York University
Brooklyn, NY 11201, USA

Yebin Wang
Mitsubishi Electric Research Labs.
Cambridge, MA 02139 USA

Kaan Ozbay
New York University
Brooklyn, NY 11201, USA

Zhong-Ping Jiang
New York University
Brooklyn, NY 11201, USA

***Abstract*—This paper proposes a novel physics-informed machine learning framework for motion planning and control of autonomous vehicles. By integrating longitudinal and lateral control, a nonlinear control problem is formulated using Model Predictive Control (MPC). To address computational challenges, a self-supervised framework, Recurrent Predictive Control (RPC), is introduced, leveraging differentiable neural networks and recurrent neural networks to train a neural network controller. Additionally, a heuristic feedback control layer is designed to reduce steady-state errors in the closed-loop tracking. Through numerical simulations and co-simulations using Simulink and CarSim, five neural network controllers are compared with an MPC controller in a lane-changing scenario. The proposed RPC framework improves computational efficiency by 95.91% compared to MPC, enhances generalization performance compared to Approximate MPC, and reduces performance loss by 17.01% compared to Differentiable Predictive Control. The heuristic feedback control layer further reduces steady-state errors and improves convergence speed during training.**

***Index Terms*—physics-informed machine learning, heuristic feedback control layer, recurrent predictive control (RPC), autonomous driving, motion planning and control**

I. INTRODUCTION

Autonomous driving, a core aspect of intelligent transportation systems, offers solutions to various mobility challenges. Key functions include cruise control, lane keeping, and lane changing. Automatic lane changing is more complex as it requires both longitudinal and lateral control, and trajectory planning often uses polynomial or geometric curves, or MPC, which effectively handles constraints but suffers from high computational cost, limiting its use to local planning [1]. To reduce complexity, prior work decouples longitudinal and lateral planning or assumes constant speed [2], though joint trajectory planning improves performance [3], revealing the trade-off between performance and computational cost.

Advances in AI have enabled new planning and control approaches. Approximate MPC (AMPC) leverages neural networks to learn explicit control policies, reducing computation expense but relying on supervised data and lacking direct optimization of control objectives and constraints, leading to learning biases [4]. Physics-informed learning, particularly PINNs, incorporates PDEs and physical constraints into loss functions, combining data with models for high sample efficiency [5], and has shown success in fluid mechanics, materials, and thermal systems [6]. Building on this, Differentiable Predictive Control (DPC) embeds system dynamics into a neural network and trains via unsupervised learning, addressing the limitations of AMPC and generalizing to various system types [7]. However, since DPC predicts the entire control sequence in a single forward pass, it cannot directly leverage the updated states from the physical model, and is also unable to ensure consistency in control inputs across time steps. AI Pontryagin [8] avoids this by using a recurrent architecture to compute control inputs step by step. Although promising, physics-informed control remains underexplored in autonomous driving, with applications mainly in longitudinal control like adaptive cruise control [9]. Recent studies have proposed PINN-based MPC for trajectory tracking [10] and physics-driven neural networks for modeling the nonlinear lateral dynamics of vehicles [11], but no prior studies have tackled joint lateral and longitudinal control using this paradigm.

To address these challenges, this paper formulates a joint longitudinal–lateral control problem that integrates trajectory planning and tracking, and proposes an RPC framework with a heuristic feedback control layer to enhance control accuracy and efficiency. The contributions are summarized as follows:

- A novel Recurrent Predictive Control framework is proposed, where a single-step differentiable unit combining the control policy and vehicle model is recursively applied across the prediction horizon to compute the cost, enabling self-supervised learning and enhancing both computational efficiency and control performance.
- A heuristic feedback control layer is introduced to enhance tracking accuracy near the reference state, effectively reducing steady-state errors.
- The proposed controller is validated through numerical and CarSim–Simulink co-simulations, demonstrating superior efficiency and accuracy compared to MPC, AMPC, and DPC.

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

This work has been supported in part by a gift from the Mitsubishi Electric Research Laboratories and a grant from C2SMARTER Center and in part by the NSF under Grants CNS-2148309 and CPS-2227153.

introduces the physics-informed machine learning framework proposed in this paper. Section IV presents the training process and compares the performance of different controllers through simulations. Finally, some concluding remarks are given in Section V.

II. PROBLEM STATEMENT

In this section, we first present a nonlinear vehicle dynamics model and then construct the automatic lane changing problem based on the MPC framework.

A. Nonlinear Vehicle Model

In this paper, a nonlinear vehicle model is derived based on a two-degree-of-freedom bicycle model, which can be described by the following ordinary differential equations:

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

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

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

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

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

where the ground and vehicle coordinate systems are denoted by (X,Y) and (x,y), respectively, with the center of gravity at CG and yaw angle ψ. Vehicle parameters include front and rear axle distances l_f, l_r, mass m, moment of inertia I_z, and cornering stiffnesses C_f, C_r. The longitudinal and lateral velocities are v_x and v_y, a is the longitudinal acceleration, δ_f the steering angle, and ω_r the yaw rate. The system states are $x = [Y, \psi, v_x, v_y, \omega_r]^T$ and the control inputs are $u = [a, \delta_f]^T$. The above model is a time-invariant nonlinear system denoted as

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

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

$$x(k+1) = x(k) + f(x(k), u(k))\Delta t = F(x(k), u(k)) \tag{3}$$

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

B. Automatic Lane Changing Problem Formulation Based on MPC

In this paper, the trajectory planning and tracking problem for automatic lane changing is reformulated as a centerline tracking problem of the target lane [12], where Y_{ref} is the lateral position of the target, L_w is the lane width, $v_{x,0}$ is the initial speed, and $v_{x,\text{ref}}$ is the desired speed after lane change. Based on the above, the lane-changing task is formulated as an

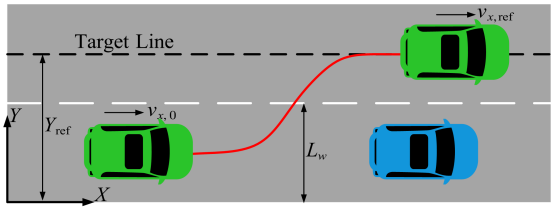


Fig. 1: Motion planning and control problem description

MPC-based dynamic optimization problem. The front wheel angle varies under piecewise constant inputs, with control and prediction horizons set equal ($N_c = N_p$). Specifically, a reference tracking MPC cost function is adopted:

$$\min_U J_{MPC} = \sum_{k=0}^{N_p-1} (x_k - x_{\text{ref}})^T Q_x (x_k - x_{\text{ref}}) + u_k^T Q_u u_k + (x_{N_p} - x_{\text{ref}})^T Q_t (x_{N_p} - x_{\text{ref}}) \tag{4}$$

where $U = [u_0, \ldots, u_{N_p-1}]$ is the control sequence, x_{ref} the reference state, x_{N_p} the terminal state, and Q_x, Q_u, Q_t are the state, control, and terminal penalty matrices, respectively, with the reference state given as follows:

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

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:

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

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

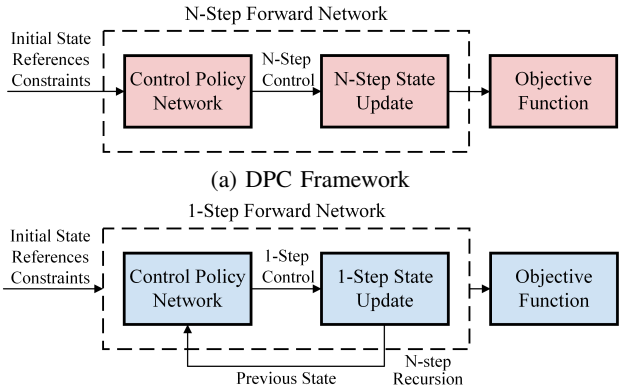
These are hard constraints on the control inputs to prevent them from exceeding reasonable limits, which would result in high energy consumption, low comfort, and potential safety risks.

III. PHYSICS-INFORMED MACHINE LEARNING CONTROLLER DESIGN

In this section, we will first provide a brief review of the DPC method, which is also used to address MPC problems. Then, we will introduce the RPC framework proposed in this paper, followed by the heuristic feedback control layer designed to reduce steady-state tracking errors. Finally, we will demonstrate the design of the lane-changing controller.

A. Recurrent Predictive Control

DPC is a self-supervised learning framework based on direct policy gradients, as shown in Fig. 2a [7]. It connects the control policy network with a differentiable system model, using initial conditions, reference values, and constraints as inputs. Forward propagation computes control sequences and states, while backpropagation updates the network via gradient descent. DPC features: (i) no need for labeled policies, (ii)



(a) DPC Framework

(b) RPC Framework

Fig. 2: DPC and RPC frameworks

direct interaction with system dynamics, and (iii) low computational complexity suitable for real-time control.

However, the control policy network in DPC generates the entire control sequence within the predictive horizon, where only the first control input directly uses state information from the previous time step, causing inconsistency in generating control inputs across time steps. To resolve this, we propose the Recurrent Predictive Control (RPC) framework, shown in Fig. 2b. Based on recurrent neural networks, RPC generates only the current control input and, together with a single-step system update model, forms a forward differentiable unit. This unit is recursively called with updated states, and all policy networks share parameters across time, ensuring consistent and scalable control.

B. Heuristic Feedback Control Layer

In practical applications, it can be observed that controllers designed based on RPC, DPC, and approximate MPC exhibit poor tracking accuracy for certain states near the reference in closed-loop simulations. This issue arises because the tracking cost near the reference is small, making it difficult to accurately learn the corresponding maneuvers using gradient descent alone. To address this problem, we introduce a heuristic feedback control layer that explicitly incorporates current state errors into the control inputs generated by the policy network:

$$u_k = K_k \left(x_{\text{ref}} - x_k\right) \tag{7}$$

where K_k is a gain matrix constructed from the network output. This formulation enhances the interpretability of the neural network controller and facilitates constraint design on the gain matrix to ensure closed-loop stability and reduce steady-state error.

C. RPC-based Lane-Changing Controller Design with Heuristic Feedback Control Layer

The training framework and real-time control framework for the lane-changing controller based on RPC with a heuristic feedback control layer (HFRPC) are shown in Fig. 3. This subsection focuses on introducing the structure design and training process of the neural network controller.

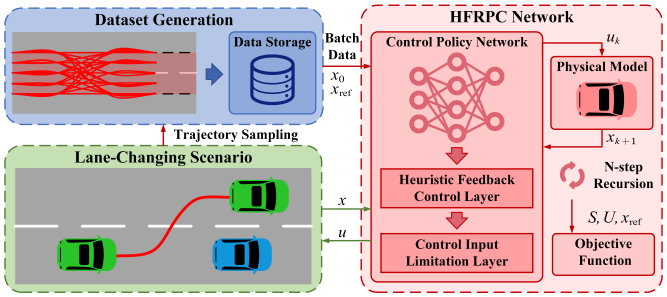


Fig. 3: HFRPC-based Lane-changing Controller Training and Real-time Control Framework: Red Arrows Represent the Data Flow During the Training Process, While Green Arrows Represent the Data Flow During the Real-Time Control Process

First, forward propagation is performed. For each 1-step forward network unit, the inputs consist only of the vehicle's current state and the reference state as follows:

$$\mathbf{x}_k = \left[x_k^T, x_{\text{ref}}^T\right]^T \tag{8}$$

Under the RPC framework, the multilayer perceptron directly generates control inputs without normalization for current time step:

$$\hat{u}_k = \pi_{\text{RPC}}\left(\theta, \mathbf{x}_k\right) \tag{9}$$

where θ represents the network parameters. However, for the HFRPC, the multilayer perceptron is utilized to generate key elements that constitute the feedback gain matrix:

$$\mathbf{g}_k = \pi_{\text{HFRPC}}\left(\theta, \mathbf{x}_k\right) \tag{10}$$

Closed-loop simulation results using alternative neural network controllers indicate that, although the errors in yaw angle, lateral velocity, and yaw rate asymptotically approach zero, noticeable steady-state errors persist in lateral position and longitudinal velocity. In this paper, by means of a Lyapunov function argument [1], we select a feedback control gain matrix of the following form:

$$K_k = \begin{bmatrix} 0 & \mathbf{g}_{k,1} & \mathbf{g}_{k,2}^2 + b_1 & \mathbf{g}_{k,3} & \mathbf{g}_{k,4} \\ \mathbf{g}_{k,5}^2 + b_2 & \mathbf{g}_{k,6} & 0 & \mathbf{g}_{k,7} & \mathbf{g}_{k,8} \end{bmatrix} \tag{11}$$

where the constants b_1 and b_2 and the functions $g_{k,i}$, $i = 1, \ldots, 8$, are selected to ensure the stability of the closed-loop system with respect to the reference trajectories. Subsequently, the control input without normalization at the current time step can be computed as follows:

$$\hat{u}_k = K_k \left(x_{\text{ref}} - x_k\right) \tag{12}$$

The gain matrix in (11) enables corrective actions for lateral deviations near the reference, aligning with human driving intuition. Unlike fixed matrices, it adapts to current and reference states for improved control far from the reference. A

[1] https://zenodo.org/records/15252897

tanh-based output layer then enforces hard bounds on control inputs:

$$u_k = \tanh(\hat{u}_k) \circ \frac{u_{\max} + u_{\min}}{2} + \frac{u_{\max} + u_{\min}}{2} \tag{13}$$

Then, the system state at the next time step can be updated using the physical model:

$$x_{k+1} = F(x_k, u_k) \tag{14}$$

At this point, the forward propagation of a single-step predictive network unit is completed. The unit is then recursively executed until all predictions within the predictive horizon are completed, and the training loss is computed as follows:

$$L = J_{\text{MPC}}(S, U, x_{\text{ref}}) \tag{15}$$

where $S = \left[x_0, x_1, ..., x_{N_p}\right]$, $U = \left[u_0, u_1, ..., u_{N_p-1}\right]$.

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 employed to compute the gradient $\nabla_\theta L$, then the network parameters are then updated as follow:

$$\theta_{k'+1} = \theta_{k'} - \eta \nabla_{\theta_{k'}} L \tag{16}$$

where η represents the learning rate. Forward and backward propagations are 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

Require: Batch of features $\left[x_0^T, x_{\text{ref}}^T\right]^T$, Current network parameters θ

Ensure: Updated network parameters θ_{k+1}

Forward Propagation: Calculating the loss function.
Initialize $k \leftarrow 0$
while $k \leq N_p$ **do**
  Input a batch of features $\left[x_k^T, x_{\text{ref}}^T\right]^T$.
  Generate the key elements of the gain matrix $\mathbf{g}_k$.
  Construct the gain matrix K_k.
  Calculate the control input $\hat{u}_k = K_k(x_{\text{ref}} - x_k)$.
  Calculate the normalized control input u_k.
  Update the state for the next time step x_{k+1}.
  $k \leftarrow k+1$
end while
Calculate the loss $L_k = J_{\text{MPC}}(S, U, x_{\text{ref}})$.
Backward Propagation: Updating the network parameters.
Compute the gradient of the loss function with respect to the network parameters $\nabla_{\theta_{k'}} L$.
Update network parameters $\theta_{k'+1} = \theta_{k'} - \eta \nabla_{\theta_{k'}} L$.

IV. SIMULATION RESULTS

This section presents the controller implementation, including training data generation, training process, and comparison with baseline methods via simulations and co-simulations. All code is available on GitHub[2]. Table I summarizes road, and controller parameters, with vehicle data from [2].

TABLE I: Road and Controller Parameters

Parameter	Value(unit)
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
b_1	0.6
b_2	0
Q_x	diag $[60, 500, 50, 10, 100]$
Q_u	diag $[5, 500]$
Q_t	diag $[60, 1000, 70, 20, 200]$

A. Dataset Generation and Training Process

The training set was built from $6,300$ lane-changing trajectories with varied initial and reference states, yielding $321,300$ samples, ensuring that all samples correspond to operating points under good road adhesion conditions. The dataset is split into training and validation sets with a ratio of 80% to 20%. The vehicle speed ranged from 80 km/h to 100 km/h, with a maximum initial Y deviation of 4 m. The control policy network is a fully connected MLP with three hidden layers of 256 nodes each, using GELU as the activation function. It is trained for 1,000 epochs using a learning rate of 0.0001 and a batch size of 10,000.

In addition to HFRPC, controllers based on DPC and RPC were trained using the same dataset. For AMPC, MPC-generated control inputs were used as labels, with and without the heuristic feedback layer. All data-driven methods shared the same MLP architecture (except output dimensions) and included control input limitation layers. Training was performed on a laptop with an NVIDIA 2070 GPU. The evolution of training and validation losses for all methods is presented in Fig. 4; final loss and training time are listed in Table III. The RPC framework achieves a 17.54% lower final training loss than DPC, with only limited further reduction from the heuristic layer. All three methods show comparable final validation losses. AMPC and HFAMPC adopt least-squares loss, with HFAMPC achieving lower final training and validation losses. Methods with heuristic feedback layers converge faster and exhibit smoother loss curves. AMPC trains the fastest, HFRPC the slowest (within 50 minutes), but HFRPC shows the fastest convergence, suggesting that fewer epochs may suffice in practice to reduce training time.

TABLE II: Training Loss and Time Summary

Method	AMPC	HFAMPC	DPC	RPC	HFRPC
Training Loss	0.0182	0.0133	2821	2326	2321
Validation Loss	0.0191	0.0168	2783	2783	2774
Training Time	200s	366s	1670s	1930s	2900s

[2]https://github.com/XianningLi/IV-2025.git

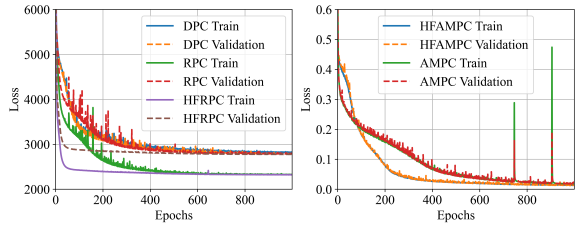


Fig. 4: Evolution of Training Losses for All Methods

B. Closed-loop Numerical Simulation Verification

Closed-loop numerical simulations were conducted to compare MPC, AMPC, and the three physics-informed controllers under no model mismatch. MPC employed CasADi, which uses an interior-point method for efficient nonlinear optimization [13]. Simulations ran with a 0.01 s step size and 0.05 s control period on an i7 2.30GHz CPU. Fig. 5 illustrates the lane change from 80 to 100 km/h, where the neural network controller yields smoother control inputs, and the physics-informed learning controller exhibits behavior more consistent with MPC.

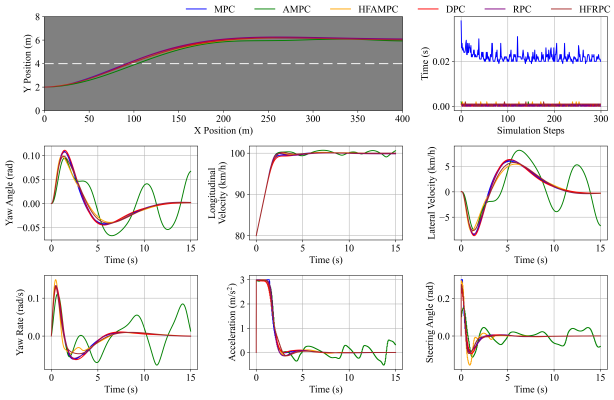


Fig. 5: Lane Change from 80 to 100 km/h

To evaluate generalization, 128 test scenarios were constructed from combinations of four initial and four reference speeds, each associated with eight left/right lane-change targets. The test range, with up to 8 meters of lateral offset and a 40 km/h speed difference, doubled that of the training set. Lane-change trajectories and key metrics are summarized in Fig. 6 and Table 2. Neural network controllers improved computational efficiency by over 95%, and only methods incorporating the heuristic feedback control layer achieved a 100% lane-change success rate. Among all methods, HFRPC demonstrated the best overall performance across all evaluated metrics.

C. Co-Simulation of CarSim and Simulink

To assess real-world applicability, a co-simulation using CarSim and Simulink was conducted. The scenario involves a leading vehicle moving at 60 km/h, initially 100 m ahead.

TABLE III: Closed-loop Numerical Simulation Performance

Method	MPC	AMPC	HFAMPC	DPC	RPC	HFRPC
v_x RMSE [m/s]	1.23	1.59	1.22	1.41	1.29	1.19
Y RMSE [m]	2.22	2.63	2.22	4.40	2.38	2.28
Avg. Calc. Time [s]	0.0249	0.0006	0.0011	0.0006	0.0006	0.0011
a Var. [$(m/s^2)^2$]	0.7877	1.1211	0.8299	0.6402	0.6874	0.6655
δ_f Var. [rad^2]	0.0034	0.0072	0.0045	0.0025	0.0028	0.0027

Lane-change decisions are based on a critical safety distance model defined as:

$$D_{\text{safe}} = 0.1v + \frac{v^2}{130} \tag{17}$$

where D_{safe} is the safety distance, and v is the speed of the following (ego) vehicle. A lane change is triggered when the gap to the preceding vehicle is less than D_{safe}.

Three scenarios were used to compare steady-state tracking errors between MPC and physics-informed controllers: acceleration (80 $\rightarrow$ 100 km/h), constant speed (90 km/h), and deceleration (100 $\rightarrow$ 80 km/h). High-fidelity simulation was used, while control relied on a simplified model. Simulations ran for 40 s, results are shown in Fig. 7, and errors after stabilization are listed in Table IV. The simulation video is available on the website[3]. The heuristic feedback layer significantly reduces errors, yielding MPC-level performance.

TABLE IV: CarSim and Simulink Co-Simulation Results

Method	MPC	DPC	RPC	HFRPC
Lateral Position [cm]	0.03	8.29	8.28	0.01
Longitudinal Speed [km/h]	0.94	1.32	1.44	0.72

V. SUMMARY

In this paper, the motion planning and control problem of autonomous driving is studied from the perspective of physics-informed machine learning. A novel RPC framework is proposed based on the recurrent neural network architecture to address the combined longitudinal and lateral control of autonomous vehicles with nonlinear models. Additionally, a heuristic feedback control layer is proposed, which can be integrated into different neural network controllers to reduce steady-state tracking errors. The results from closed-loop numerical simulations and co-simulations demonstrate that the proposed HFRPC framework is computationally more efficient than traditional MPC and shows performance improvement over previous physics-informed DPC method. In addition, the proposed approach has led to significant reduction of the steady state tracking error. Since the proposed method is developed based on a standard MPC framework, it holds the potential to be extended to more complex tasks such as urban driving, highway merging, or obstacle avoidance in future work.

[3]https://youtu.be/FjwLgkqRQnQ

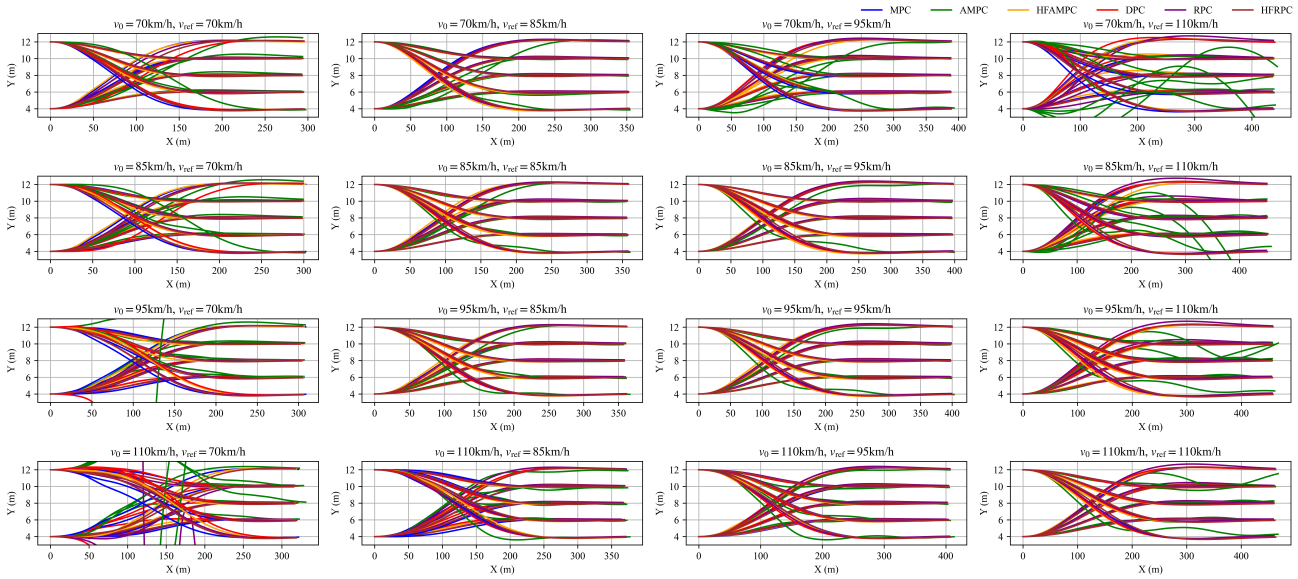


Fig. 6: Closed-loop Numerical Simulation Results. Failure cases: AMPC exhibited 12 significant deviations, DPC and RPC failed in 2 and 1 cases, respectively. HFAMPC and HFRPC achieved a 100% success rate.

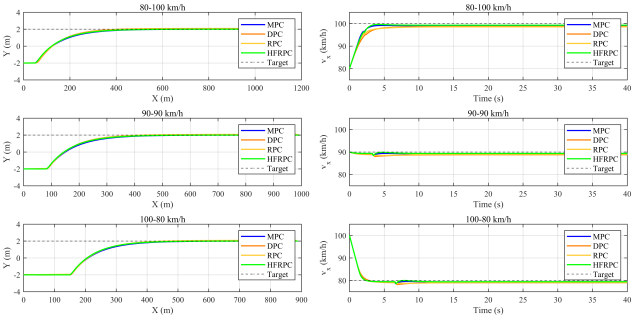


Fig. 7: CarSim and Simulink Co-Simulation Results

REFERENCES

[1] C. Ma and D. Li, "A review of vehicle lane change research," *Physica A: Statistical Mechanics and its Applications*, vol. 626, p. 129060, 2023.

[2] 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 Conferenc Conference (CCDC)*, 2019, pp. 1468–1473.

[3] 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)*, 2021, pp. 1929–1934.

[4] 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, 2021.

[5] 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.

[6] 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. 0644, 2022.

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

[8] 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.

[9] 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, 2023.

[10] L. Jin, L. Liu, X. Wang, M. Shang, and F.-Y. Wang, "Physical-informed neural network for mpc-based trajectory tracking of vehicles with noise considered," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 3, pp. 4493–4503, 2024.

[11] M. Piccinini, S. Taddei, M. Larcher, M. Piazza, and F. Biral, "A physics-driven artificial agent for online time-optimal vehicle motion planning and control," *IEEE Access*, vol. 11, pp. 46 344–46 372, 2023.

[12] S. Chakraborty, L. Cui, K. Ozbay, and Z.-P. Jiang, "Automated lane changing control in mixed traffic: An adaptive dynamic programming approach," *Transportation Research Part B: Methodological*, vol. 187, p. 103026, 2024.

[13] 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.