Physics Informed Neural Networks: A Quantum Leap in Robust CAVs Control

Ahmed H. Labib[1 2], Ahmed W. elMarsafy[1 2], Ziad M. elGendy[1 2], Zeyad E. Farghaly[1 2], Ziad H. Karawia[1 2], Zeyad M. AbdulFattah[1 2], and Mohammed D. ElHosary[1 2]

[1]Faculty of Engineering, Cairo University
[2]Department of Electronics and Communications Engineering

Abstract

Modeling a physical system governed by a system of differential equations can be both computationally expensive and time-consuming, especially with the lack of an analytical solution.

We will use Physics-Informed Neural Networks by supplying it with the appropriate training data for our connected vehicle's control system, with values for input and output parameters for many instances of its operation, then training the model using an altered loss function which will involve some of the basic governing ODEs or PDEs of the system to allow for improved, more efficient modeling. Next, we will use the mathematical model obtained from our PINNs to design our control system, which will have a controller that is designed with robust control theory in mind, which allows for more resilient and predicted behavior of the control system under unexpected circumstances. Our approach aims to combine an efficient way of modeling our control system, allowing it to work in conditions which are not exactly specified, and applying control theory to advance vehicle-to-vehicle communication.

Keywords: PINNs, Robustness, Loss function, Residual, CAVs

1 Problem Definition

In recent years, the rapid advancement of autonomous vehicle technology has highlighted the potential for Connected and Automated Vehicles (CAVs) to significantly enhance road safety and traffic efficiency. Despite these advancements, traffic accidents remain a major concern, with over 1.19 million deaths [1] due to road crashes, primarily due to human errors and the limitations of current vehicle automation systems. The challenge lies in developing a robust and reliable system that can effectively prevent accidents and ensure smooth traffic flow.

The primary motivation for this project is to address the persistent issue of traffic accidents and to improve overall road safety. By connecting vehicles to form CAVs, we aim to leverage the benefits of vehicle-to-vehicle (V2V) communication and advanced sensing technologies to create a safer driving environment. Furthermore, by incorporating vehicle platoon systems, we can further reduce accidents for vehicle groupings, from petroleum-loaded trucks to colleagues driving to work, and allow for reduced collective air drag, which may result in about 15% lower fuel consumption [2] for a given configuration, therein reducing CO_2 emissions. The potential integration of Light Detection and Ranging (LIDAR) technology will enable precise detection of surrounding vehicles and intervehicle communications, while also bringing a more cost-effective alternative to cameras and GPS sensors, making it more feasible for mass production. To achieve this, we will consider our system as a control system and will begin by modeling our system using its governing Ordinary Differential Equations (ODE) or Partial Differential Equations (PDEs), then design a controller to coordinate between the control inputs, possibly including those given by the LiDAR sensors, such as the position and speed difference of the preceding vehicle, and take appropriate actions such as steering, braking, or even alerting nearby LiDAR-equipped vehicles of an accident. However, such control systems have many control inputs, many of which are ignored to simplify controller design, often resulting in undesirable behavior in real world conditions. This led us to integrate robust control techniques into the controller design, allowing a wider variety of operating conditions and allowing the control system to maintain its desired behavior. To further enhance the robustness of our system, we will deploy Physics-Informed Neural Networks (PINNs), which are given both training data and the governing ODEs or PDEs of our system to develop an accurate and efficient model. This has several benefits, including:

- Creating an efficient, accurate, and robust model of our system.

- The ability to implement more realistic models with more complex ODEs or PDEs, accounting for many of the external factors early on.
- Better prediction of future system states (position of the neighboring vehicle) and more accurate control decisions.

The combination of PINNs and LIDAR data will enable us to create a system that can predict and respond to potential hazards, thus reducing the percentage of accidents occurring.

2 Literature Review

The research and development of fully connected and autonomous vehicles, or CAVs, that are capable of navigating roads without a human driver and connecting and interfacing with other vehicles and their environments, has increased rapidly in recent times. The technology is expected to be available to consumers in the very near future, with some predicting that fully autonomous vehicles could be on the market in 2025 [3].

CAVs are informed on their surroundings to allow them to take quick and intelligent actions and make appropriate maneuvers, whether through adjusting steering angle, accelerating or decelerating, with the adaptive cruise control (ACC) system deciding on such actions based on supplied data like space-gap and relative velocity with other neighboring vehicles.

Some of the more prevalent controller design techniques for ACC systems include: Model Predictive Control (MPC), Sliding Mode Control (SMC), H-infinity, and Minimum-Maximum Robust MPC.

MPC [4] optimizes system performance by predicting future states and computing optimal control inputs within a given time. MPC explicitly considers system constraints, such as input limits, state boundaries, and rate of change constraints, making it highly suitable for complex and constrained systems. The model can handle multivariable, non-linear, and dynamic systems while ensuring smooth and stable operation.

In order to model CAVs, it should be known that this type of system is non-holonomic meaning that there are constraints on motion that cannot be expressed purely in terms of position coordinates. The kinematic model of the vehicle due to constraints assumes that the vehicle moves without slipping and that is controlled by two inputs: linear driving velocity and angular steering velocity.

Vu Trieu Minh, Reza Moezi[5] proved that the system can be controllable using Jacobian matrices and Lie brackets although being non-holonomic. Furthermore, They attempted to linearize the system using Taylor series expansion around a reference point. This way transforms the system's equations to first-order approximations. Additionally, They attempted to discretize the system using a sampling interval Δt.

For MPC optimization formulation , Objective/Cost function ($\mathbf{J}$) is minimized in order to optimize the control inputs through considering both the tracking error, deviation from reference trajectory, and the smoothness of control inputs, known as the control effort, In addition, MPC incorporates physical constraints on inputs and outputs like maximum speed.

Despite the success of the modified MPC algorithms, challenges remain in dealing with model uncertainty and model-plant mismatches . Further research is needed to assess the effectiveness of softened constraints and output regions.

Second type of controller design techniques is SMC[6]. Sliding mode control (SMC) was originally developed for dynamic systems that can be effectively modeled using ordinary differential equations (ODEs). It employs a discontinuous control action, often referred to as variable structure control (VSC), operating within the continuous-time domain. The resulting feedback system, termed a variable structure system (VSS), is characterized by ODEs with discontinuous right-hand sides. The region in the state-space where sliding mode occurs is known as the sliding manifold.

Chattering, a common phenomenon in SMC, refers to oscillatory motion around the sliding manifold. Two primary mechanisms can induce chattering. The first involves parasitic dynamics, such as fast actuator and sensor dynamics. The second mechanism arises due to non-idealities, and high frequency oscillations. In the absence of switching imperfections, such as delays, these parasitic dynamics can cause small-amplitude, high-frequency oscillations near the sliding manifold. While the motion of a real system typically approximates that of an ideal system (where parasitic dynamics are neglected), the difference between the two, proportional to the neglected time constants, decays rapidly.

Given the uncertainties, disturbances, and model inaccuracies, Robust control is typically implemented when designing controllers that can maintain stability and performance.

Specifically, H∞[7] control is an indispensable type of robust control methodology. it focuses on minimizing the H∞ norm of the transfer function from disturbances to performance outputs. The H∞ norm represents the worst-case am-

plification of disturbances in the system. By minimizing this norm, H∞ control ensures: robust stability and performance optimization. So in CAVs we need to handle uncertainties, path following, speed regulation, Disturbance Rejection, Robust Performance and integration with advanced methods

In order to control vehicle platooning under uncertain external disturbances such as model mismatches, sensor noise, and communication delays, an MPC approach based on Min-Max optimization is proposed[8], Min-Max MPC optimizes control actions by addressing worst-case scenarios through robust constrained optimization. By incorporating causal disturbance feedback, the controller dynamically compensates for past disturbances, thereby reducing conservatism in the control strategy while maintaining compliance with physical constraints such as speed, acceleration, and inter-vehicle distance. The proposed method delivers smooth and stable control performance, effectively minimizing position and velocity tracking errors. Experimental evaluations under three distinct disturbance types demonstrated that the Min-Max MPC approach outperformed other methods in terms of stability and robustness. It achieved a significant reduction in tracking errors for both position and velocity.

Many control system designers often resort to a simpler model of their system, often to simplify controller design and modeling their system. However, Physics-Informed Neural Networks (PINNs) have shown great potential in such tasks, mainly by integrating physical laws and boundary conditions directly into the loss function to be minimized, showcasing exceptional gains over traditional data-driven approaches in tasks as solving highly non-linear problems[9] and in solving inverse optimization problems as inferring ACC design parameters[10] from car following data such as relative velocity and space gap, allowing us to identify the string stability of the ACC platoon, which is a measure of the degree of elimination of disturbances along the platoon. Minimizing such disturbances can also be achieved by robust control techniques.

Adaptive Cruise Control (ACC) has been highly regarded as a method of adjusting the vehicle's speed whether to maintain a safe pre-defined (time/space) distance from a nearby vehicle or to reach some specified speed. Although widely implemented, the design of ACC remains partially undisclosed, specifically regarding its design parameters, but a more easily known spacing policy with the Constant Time Headway Policy (CTHP) being one of the most commonly used, which is able to reproduce ACC vehicle dynamics in platoons.

Although traditional control schemes can yield satisfactory results in managing simple and linear tasks, they struggle to address dynamic systems that change over

time and are affected by noise disturbances. (MPC) approaches typically do not account for these noise disturbances. To enhance trajectory tracking under such conditions, we will employ MPC combined with Physics-Informed Neural Networks (PINNs) within the framework of Artificial Systems, Computational Experiments, and Parallel Execution (APC).

Soft constraints will be implemented to manage uncertainties and noise[11], which will allow for minor violations and errors while still maintaining a good trajectory. The developed PINN model differs from Recurrent Neural Networks (RNNs) in that it incorporates physical information in its parameters, which helps to reduce pre-training time for the scheme.

In conclusion, Robust MPC effectively manages complex dynamical systems by handling uncertainties and constraints. Integrating Physics-Informed Neural Networks (PINNs) enhances its performance by embedding physical laws into neural networks, ensuring adherence to system dynamics and improving accuracy. This combined approach shows promise for advancing control methodologies, with future work focused on testing and refining it for superior performance in complex scenarios.

3 Mathematical Modeling and Methodology

> *"Much of the activity of science is an application of that paradigm : given the description of some natural phenomena, find the* ***differential equations*** *for the processes that produce the phenomena."*
>
> **Herbert Simon**
>
> Nobel prize in Economics,1978

A fundamental component of contemporary mathematical modeling, partial differential equations (PDEs) are necessary to comprehend the behavior of complex systems in a wide range of scientific fields., where dynamics vary across both space and time. From the microscopic interactions of particles to the vast dynamics of celestial bodies, PDEs are employed to study heat conduction in materials, wave propagation across diverse media, fluid dynamics in gases and liquids, and electromagnetic fields in space.

The partial differential equation in its general linear form is expressed as:

$$F(x, t, u, \nabla u, u_t, \nabla^2 u, u_{tt}, ...) = 0 \tag{1}$$

where $u : D \times (0, T) \to \mathbb{R}^m$, where $D \subset \mathbb{R}^d$

x is displacement.

t is time.

u is the system's state.

∇u is the gradient of u.

u_{tt} is the second-order derivative of the temporal coordinates.

$(0, T)$ is the time domain over which the solution is sought, with T being the final time.

Extending this notion to non-linear PDE is essential to encapsulate a wide range of problems in mathematical physics such as studying hydrodynamic traffic flow models for traffic simulation.

It is generally expressed as

$$u_t + \mathcal{N}[\mu; \lambda] = 0 \tag{2}$$

where $u(t, x)$ denotes the hidden solution (state of the system) and $\mathcal{N}[.; \lambda]$ is a nonlinear operator parameterized by λ.

3.1 Longitudinal Control Dynamics

The longitudinal control equation focuses on managing a vehicle's acceleration and deceleration to maintain speed and safe distances between vehicles. The high-order Ordinary Differential Equation (ODE) for longitudinal dynamics is given by:

$$m\dot{v} = u - \frac{1}{2}\rho C_d A v^2 - F_{rr} \tag{3}$$

- F_{rr}: Rolling resistance force (e.g., 300 N, typical for a 1500 kg vehicle).
- The $\frac{d^2v}{dt^2}$ term is no longer needed since this is a first-order equation.
- $m\dot{v}$: Newton's second law of motion where m is the vehicle's mass and $\dot{v}$ is the vehicle's acceleration.

- u: Control force of throttle/brake input.
- $\frac{1}{2}\rho C_d A v^2$: Aerodynamic drag force where
  - ρ is air density.
  - C_d is drag coefficient.
  - A is frontal area.
  - v is vehicle's velocity.

This equation describes how the vehicle's acceleration (second derivative of velocity v) is influenced by the control input $u(t)$ while considering resistive forces (damping and stiffness). It ensures smooth speed transitions and prevents abrupt changes that could destabilize the vehicle.

This longitudinal dynamics equation is directly linked to:

3.1.1 Adaptive Cruise Control (ACC)

The control input $u(t)$ comes from the ACC system, which adjusts acceleration and braking to maintain a safe distance from the lead vehicle. The equation helps model how the Connected and Autonomous Vehicle (CAV) responds to control commands while considering real-world resistances.

3.1.2 Platooning

In a platoon of CAVs, each vehicle needs to adjust its speed smoothly based on information from:

- Its own sensors.
- Wireless Vehicle-to-Vehicle (V2V) communication.

The equation helps design control laws that prevent string instability, where disturbances amplify along the vehicle string.

3.1.3 Cooperative Adaptive Cruise Control (CACC)

Here, $u(t)$ is influenced by data from multiple vehicles (V2V), not just onboard sensors. The dynamics help synchronize acceleration and deceleration between CAVs to maintain tight but safe spacing.

3.1.4 Control Design

Many papers use this exact model to design:

- Feedback controllers.
- Model Predictive Control (MPC) for CAVs.
- Physics-Informed Neural Networks (PINNs) for learning controllers.

State Variables

- v: Longitudinal velocity
- ψ: Yaw angle
- $r = \dot{\psi}$: Yaw rate
- δ: Steering angle (input)
- u: Longitudinal force (input)

Vehicle Parameters

- m: Mass of the vehicle
- I_z: Yaw moment of inertia
- l_f, l_r: Distance from center of gravity to front and rear axles
- C_{af}, C_{ar}: Cornering stiffness of front and rear tires
- ρ: Air density
- C_d: Drag coefficient
- A: Frontal area
- F_{rr}: Rolling resistance force

Nonlinear State Equations

1. Longitudinal Dynamics

$$\dot{v} = \frac{u - \frac{1}{2}\rho C_d A v^2 - F_{rr}}{m}$$

2. Yaw Rate Dynamics

$$\dot{r} = \frac{l_f F_{yf} - l_r F_{yr}}{I_z}$$

3. Yaw Angle Dynamics

$$\dot{\psi} = r$$

Tire Slip Angles

$$\alpha_f = \delta - \frac{l_f r}{v}, \quad \alpha_r = -\frac{l_r r}{v}$$

Lateral Tire Forces (Nonlinear Model)

$$F_{yf} = C_{af} \tan^{-1}(3\alpha_f), \quad F_{yr} = C_{ar} \tan^{-1}(3\alpha_r)$$

Final Nonlinear State-Space Representation

$$\begin{cases} \dot{v} = \frac{u - \frac{1}{2}\rho C_d A v^2 - F_{rr}}{m} \\ \dot{r} = \frac{l_f C_{af} \tan^{-1}(3\alpha_f) - l_r C_{ar} \tan^{-1}(3\alpha_r)}{I_z} \\ \dot{\psi} = r \end{cases}$$

where:

$$\alpha_f = \delta - \frac{l_f r}{v}, \quad \alpha_r = -\frac{l_r r}{v}$$

3.2 Role in CAV Control

This model is essential for:

- Speed regulation under dynamic scenarios like stop-and-go traffic.
- Following distance control using Adaptive Cruise Control (ACC).
- Integration with trajectory planning in full autonomous driving.

3.3 Lateral Control Dynamics

The lateral control equation governs the vehicle's steering behavior, ensuring accurate lane-keeping and trajectory tracking. The ODE for lateral dynamics is:

$$\alpha_f = \delta - \frac{r l_f}{v}, \quad \alpha_r = -\frac{r l_r}{v} \tag{4}$$

Assuming small slip angles, the lateral tire forces are:

$$F_{y,f} = C_{\alpha f}\alpha_f, \quad F_{y,r} = C_{\alpha r}\alpha_r$$

The yaw acceleration $\dot{r} = \frac{d^2\psi}{dt^2}$ is:

$$I_z\dot{r} = l_f F_{y,f} - l_r F_{y,r}$$

Substituting the tire forces:

$$I_z\dot{r} = l_f C_{\alpha f}\left(\delta - \frac{rl_f}{v}\right) - l_r C_{\alpha r}\left(-\frac{rl_r}{v}\right)$$

$$I_z\dot{r} = l_f C_{\alpha f}\delta - \frac{l_f^2 C_{\alpha f} + l_r^2 C_{\alpha r}}{v} r$$

- δ: Steering angle (driver input) [rad]
- r: Yaw rate (rotation speed around vertical axis) [rad/s]
- $\dot{r}$: Yaw acceleration [rad/s^2]
- v: Vehicle longitudinal velocity [m/s]
- α_f, α_r: Front/Rear tire slip angles [rad]
- $F_{y,f}, F_{y,r}$: Front/Rear lateral tire forces [N]
- $C_{\alpha f}, C_{\alpha r}$: Front/Rear tire cornering stiffness coefficients [N/rad]
- l_f, l_r: Distance from center of mass to front/rear axles [m]
- I_z: Vehicle yaw moment of inertia [kg·m^2]

3.4 Significance in Research

These equations are pivotal for designing advanced control systems in CAVs. By understanding and solving these ODEs, researchers can develop controllers that enhance:

- **Safety**: Preventing collisions through precise speed and steering control.
- **Efficiency**: Optimizing fuel consumption and traffic flow in platooning scenarios.
- **Stability**: Ensuring smooth and predictable vehicle behavior under varying road conditions.

3.5 Model Predictive Control for Connected and Autonomous Vehicles (CAVs)

3.5.1 Problem Formulation

The state vector includes X, Y, v, ψ, and δ representing the position, speed, heading, and steering angle of the vehicle. The control inputs are the acceleration a and steering rate $\dot{\delta}$. The kinematic bicycle model is used to describe vehicle dynamics:

$$\dot{X} = v\cos(\psi) \tag{5}$$

$$\dot{Y} = v\sin(\psi) \tag{6}$$

$$\dot{v} = a \tag{7}$$

$$\dot{\psi} = \frac{v}{L_f + L_r}\tan(\delta) \tag{8}$$

$$\dot{\delta} = \dot{\delta} \tag{9}$$

3.6 Cost Function

$$J = \sum_{k=0}^{N-1} \left[(x_k - x_{ref})^T Q (x_k - x_{ref}) + u_k^T R u_k\right] + (x_N - x_{ref})^T P (x_N - x_{ref}) \tag{10}$$

3.7 Parameters Explained in Detail

- $x_k = [X_k, Y_k, v_k, \psi_k, \delta_k]$: State vector at step k.
- $x_{ref} = [X_{ref}, Y_{ref}, v_{ref}, \psi_{ref}, \delta_{ref}]$: Reference state vector.
- $u_k = [a_k, \dot{\delta}_k]$: Control input vector.
- $Q = \text{diag}(q_X, q_Y, q_v, q_\psi, q_\delta)$: Weighting matrix on state tracking error.
- $R = \text{diag}(r_a, r_{\dot{\delta}})$: Weighting matrix on control input.
- P: Terminal weighting matrix.
- N: Prediction horizon (e.g., $N = 10$ to 50 for CAVs).

3.8 MPC Simulation

We designed a Simulink block diagram for an MPC-based control system in order to simulate longitudinal (speed) dynamics and yaw dynamics for an autonomous vehicle.

3.9 Trial Simulation : a sinosoidal reference path (with noise and uncertainty)

1. **System Representation**

A vehicle moving on a 2D plane can be described using the following three states:

- x (Position): The longitudinal displacement along the path.
- v(Velocity): The speed of the vehicle along the trajectory.
- ψ (Yaw Angle): The rotation angle of the vehicle relative to the x-axis.

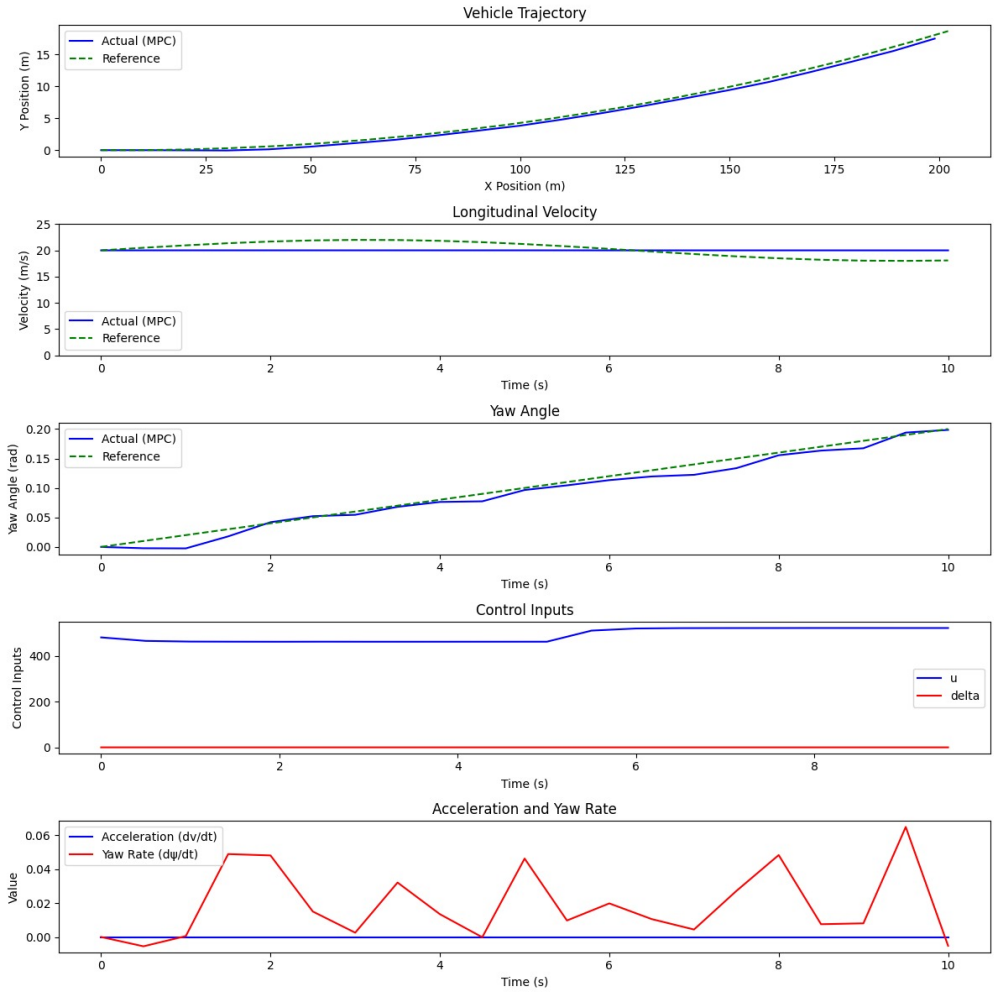

Figure 2: Trial 2 – Tracking performance under varying control inputs.

3.10 Trajectory Tracking Performance (First Plot)

- **Red dashed line:** Reference trajectory
- **Blue line:** Actual trajectory

- The vehicle follows the reference, with noticeable lag, suggesting that MPC effectively controls the system but could be improved for real-time adjustments.

3.11 Velocity Tracking Performance (Second Plot)

- The reference speed (red dashed line) is smoother than the actual speed (blue solid line).
- The actual speed deviates significantly.

3.12 Yaw Angle Tracking (Third Plot)

- The yaw angle tracking is smooth but has slight tracking errors during rapid transitions (at peaks).
- This suggests that the controller effectively prevents sudden oscillations but could benefit from faster responsiveness.

3.13 Control Inputs (Last Plot)

- The applied force varies smoothly, indicating optimized tuning of control weights in MPC to prevent excessive fluctuations.

3.14 Part 2: Error Analysis

3.14.1 Position Tracking Error

- The error follows a sinusoidal-like pattern, showing that the controller continuously corrects deviations but with lag.
- The controller is not perfectly rejecting disturbances, leading to sustained tracking deviations.

3.14.2 Speed Tracking Error

- The speed error follows a sinusoidal pattern, indicating slow response time.
- The maximum error is large, suggesting that speed tracking needs further improvement.

3.14.3 Yaw Angle Tracking Error

- The yaw error follows the same pattern as the speed error, indicating a delay in response.
- There is significant overshoot, suggesting that the yaw dynamics are not fully captured in the model.

The MPC controller did not perform perfectly. There are noticeable tracking errors in position, speed, and yaw angle, indicating delays and oscillations in the system's response.

3.15 Physics Informed Neural Networks (PINNs)

We are going to introduce a radical but advantageous solver for partial differential equations known as Physics-Informed Neural Networks (PINNs). We will discuss how these networks can effectively solve PDEs with minimal data, making them a more efficient alternative to traditional solvers, which can be time-consuming and less effective. PINNs approximate solutions by enforcing the governing PDEs and initial/boundary conditions within the loss function. This approach is advantageous compared to traditional solvers, which have several disadvantages, including stability issues. Traditional methods often require small time steps to maintain stability, resulting in slow simulations. In addition, they frequently necessitate special handling to address the complexities of PDEs, especially in high-dimensional problems. Neural networks can be trained to solve supervised linear tasks based on any given physical law described by general nonlinear partial differential equations. Physics-Informed Neural Networks are a deep learning framework specifically designed to solve these equations by integrating both data-driven and physical constraints, rather than relying solely on traditional solvers. In our work, we will employ a specific type of PINN that accepts control input u and an initial state $y(0)$. This allows the model to be integrated into a control framework known as Model Predictive Control (MPC), which we are interested in, as the network predicts the system's evolution. Firstly, We govern the differential equation for the PINN, which is the same as (2).

Definition of the PDE Residual

$$\mathcal{F}(y) := \partial_t y + \mathcal{N}[y] = 0 \tag{11}$$

where $\mathcal{F}(y)$ is the residual of the differential equation.

It measures how well a candidate solution y satisfies the governing equation. When $\mathcal{F}(y)$=0 exactly, y is an exact solution of the differential equation. The network is trained so that $\mathcal{F}(y)$ is driven close to zero. We will define the loss function we want to ensure both known data and thee physics of the problem are respected.

The total loss function is governed by

$$MSE = MSE_y + MSE_f \tag{12}$$

where MSE is The overall mean squared error (loss) that the network minimizes during training, MSE_y is the data loss term, which enforces that the network's outputs match any available (boundary or initial) data, and MSE_f is the physics loss term, which enforces that the network's outputs satisfy the governing differential equation (via the residual $\mathcal{F}(y)$). Each part on the right-hand side of the loss function is governed by

$$MSE_y = \frac{1}{N_y}\sum_{i=1}^{N_y}\frac{1}{N_t}\sum_{j=1}^{N_t}|y_i(t^j) - \hat{y}_i^j|^2 \tag{13}$$

$$MSE_f = \frac{1}{N_y}\sum_{i=1}^{N_y}\frac{1}{N_{\mathcal{F}}}\sum_{k=1}^{N_{\mathcal{F}}}|\mathcal{F}(y_i(t^k))|^2 \tag{14}$$

$y_i(t_j)$: The i^{th} output of the neural network at time t_j. For a system with multiple state variables, i indexes those variables.

$\hat{y}_i^j$: The target (or measured) value for the i^{th} state variable at time t_j (e.g., from initial or boundary conditions).

N_y : The number of outputs (i.e., the dimensionality of the state).

N_t : The number of data points (time samples) for which we have target values.

$\mathcal{F}(y_i(t_k))$: The residual computed for the i-th output at a collocation point t_k. It quantifies the violation of the governing equation.

$N_{\mathcal{F}}$: The number of collocation points where the physics loss is evaluated. These points are chosen throughout the domain to enforce the differential equation globally.

We will use special type of pinns which has control input and initial state y(0) that will help us much more for the control framework we want to reach so we can involve control system,

3.16 Augmented Network for Control

The PINN (special model for control) framework defines the system's predicted state as:

$$y(t) = f_w(t, y(0), u), t \in [0, T] \tag{15}$$

where $y(t)$ is The predicted state at time t. $y(0)$ is The initial state (which can vary over different control intervals). u is the control input applied over the time interval. t is the continuous time variable (within an inner interval of length T). w are The parameters (weights) of the PINN network.

After that we would have the training process that is important for minimizing the loss, in this step we will use gradient-based methods (such as ADAM which is an optimization algorithm), also we will use automatic differentiation to compute both the time derivative $\partial_t y$ (needed for $\mathcal{F}(y)$) and the loss gradients with respect to the network parameters.

After the PINN is trained it will be used for self-loop mode for control the network will predict the state at end of a time interval T , we will define that in Discrete time chaining for control For Model Predictive Control (MPC), the continuous prediction is "chained" into discrete steps. The prediction at each control time step is given by:

$$y[k] = f_w(T, y[k-1], u[k]) \tag{16}$$

where $y[k]$ is the predicted state at discrete time step k. $y[k-1]$ is the state at the previous timestep (or the measured state). $u[k]$ is the control input applied during the k^{th} interval and T is the duration of the inner continuous time interval (often equal to the sampling time T_s).

This **chaining** allows the PINN network to simulate long-term behavior by using its own prediction as the next initial condition. This new special PINN will have some different loss function than we specified before.

3.17 The New Loss Function

The training loss in the special PINN is analogous to traditional PINNs but now includes the augmented inputs. It has two parts:

Data loss

$$MSE_y = \frac{1}{N_y} \sum_{i=1}^{N_y} \frac{1}{N_t} \sum_{j=1}^{N_t} |y_i(v^j) - \hat{y}_i^j|^2 \tag{17}$$

where v^j=(t,y(0),u)^j:The full input tuple for the j^{th} training sample (time, initial state, control). $\hat{y}^j$ is the known or desired output for that training sample and N_t is the number of training data points (typically corresponding to initial conditions).

Physics Loss

$$MSE_f = \frac{1}{N_y}\sum_{i=1}^{N_y}\frac{1}{N_{\mathcal{F}}}\sum_{k=1}^{N_{\mathcal{F}}}|\mathcal{F}(y_i(v^k))|^2 \tag{18}$$

where v^k are Collocation points (now each v^k includes a time value, an initial state, and a control input).

Total Loss

Therefore, the total loss in PINN is given by

$$MSE = MSE_y + \lambda + MSE_f \tag{19}$$

The last step is defining the MPC Formulation using PINN. The control problem in MPC is formulated as an optimization. A typical quadratic cost function is:

$$J = \sum_{j=N_1}^{N_2}||y[k+j] - y^{ref}[k+j]||^2_{\mathbf{Q}} + \sum_{i=0}^{N_\omega=1}||\Delta\mathbf{u}[k+i]||^2_{\mathbf{R}} \tag{20}$$

subject to the dynamics constraint (here replaced by the PINC network):

$$y[k+j+1] = \hat{f}_w(y[k+j], u[k+j]), \quad \forall j = 0, \ldots, N_2 - 1 \tag{21}$$

- $y[k+j]$: Predicted state at future step $k+j$.
- $y^{\text{ref}}[k+j]$: Reference state to be tracked.
- Q, R: Weight matrices that penalize state tracking error and control effort, respectively.
- $\Delta u[k+i]$: Incremental control changes.
- $\hat{f}_w$: The PINN network acting as the system's predictive model.

This formulation enables real-time optimization where the PINC network replaces traditional numerical solvers to predict future behavior under control inputs.

4 Experimental Setup

We implemented a **Physics-Informed Neural Network (PINN)** for autonomous vehicle dynamics modeling and control, combining data-driven learning with first-principles physics. The framework solves two coupled problems:

- Learning a neural surrogate model of vehicle dynamics governed by:
  $m\dot{v} = u - \frac{1}{2}\rho C_d A v^2 - F_{rr}$
  $I_z\dot{r} = l_f C_{\alpha f}\left(\delta - \frac{rl_f}{v}\right) - l_r C_{\alpha r}\left(\frac{rl_r}{v}\right)$
- Enabling real-time MPC through differentiable physics constraints

The neural network we implemented extends PINNs with control inputs which then be inserted into the MPC. Key innovations include hybrid ADAM/L-BFGS training for physics-compliant convergence, residual connections for gradient stability, and embedded actuator dynamics ($\tau_u\dot{u} + u = u_{cmd}$) for realistic control synthesis. The architecture demonstrates how PINNs can bridge simulation-to-reality gaps in autonomous systems while maintaining interpretability through physical parameterization ($C_{\alpha f}$, C_d, etc.).

Adam optimizes NN weights using adaptive momentum ($m_t = \beta_1 m_{t-1} + (1 - \beta_1)g_t$) and scaling ($v_t = \beta_2 v_{t-1} + (1 - \beta_2)g_t^2$). It corrects bias ($\hat{m}_t = m_t/(1 - \beta_1^t)$) and updates parameters ($\theta_{t+1} = \theta_t - \eta\hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon)$).

Key advantages:

- Automatic learning rate adaptation
- Momentum acceleration
- Robustness to noisy gradients

The training process utilizes numerical integration via `solve_ivp` to simulate state transitions:

$$(v_0, \dot{v}_0, \psi_0, r_0, u, \delta) \rightarrow (v_1, \dot{v}_1, \psi_1, \dot{\psi}_1) \tag{22}$$

Normalization scales are applied as:

4.1 Network Architecture

The CAV_PINC_Network comprises:

- **Input Layer**: 7D (time + states + controls)

Feature	Normalization
Velocity (v)	$v/30$
Acceleration ($\dot{v}$)	$\dot{v}/5$
Yaw angle (ψ)	$\psi/0.5$
Yaw rate (r)	$r/1$
Control input (u)	$u/3000$
Steering angle (δ)	$\delta/0.5$

- **Initial Layer**: 256 units, tanh activation, He-normal initialization
- **Residual Blocks** (5x):
  - Main path: Dense + tanh
  - Residual path: Linear dense
  - Merge: Add + LayerNorm + tanh
- **Output Layer**: 4D ($\hat{v}$, $\widehat{\dot{v}}$, $\hat{\psi}$, $\widehat{\dot{\psi}}$)

For the complete source code of the setup, refer to Appendix A.

4.2 Training Protocol

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{data}} + \lambda \mathcal{L}_{\text{phys}} \quad \text{where} \quad \lambda = 0.1 \frac{\mathcal{L}_{\text{data}}}{\mathcal{L}_{\text{phys}}} \tag{23}$$

Training parameters:

- Optimizer: Adam ($\eta = 10^{-3}$) where η is the learning rate.
- Batch size: 256

4.3 Implementation Details

4.3.1 Simulation Setup

- Time step: $\Delta t = 0.5$s
- Horizon: $T = 10$s (20 steps)

- Physical parameters:

$$\begin{aligned} m &= 1500\,\text{kg}, \quad I_z = 2500\,\text{kg}\cdot\text{m}^2 \\ C_{\alpha f} &= C_{\alpha r} = 50000\,\text{N/rad} \\ C_d &= 0.3, \quad A = 2.2\,\text{m}^2 \\ \rho &= 1.225\,\text{kg/m}^3, \quad F_{rr} = 300\,\text{N} \end{aligned} \tag{24}$$

4.3.2 MPC Formulation

$$\min_{\mathbf{u}} \sum_{k=0}^{19} \left(\underbrace{10(v_k - v_{ref})^2 + 50(\psi_k - \psi_{ref})^2}_{\text{tracking}} + \underbrace{0.001(u_k^2 + \delta_k^2)}_{\text{effort}} + \underbrace{1000\max(0, |r_k| - 0.5)^2}_{\text{constraints}} \right) \tag{25}$$

4.3.3 Closed-Loop Execution

1. Measure noisy state: $\mathbf{x}_{\text{meas}} = \mathbf{x}_{\text{true}} + \mathcal{N}(0, 0.01)$
2. Solve MPC for $\mathbf{u}^*_{0:19}$
3. Apply (u_0, δ_0) with actuator dynamics:

$$u_{\text{actual}} = u_0(1 - e^{-\Delta t/\tau_u}), \quad \tau_u = 0.2\,\text{s} \tag{26}$$

4. Repeat for 10s duration

4.4 Results and Analysis

The Physics-Informed Neural Control (PINC) system demonstrates effective tracking performance against the reference trajectory, with notable characteristics in each state variable as shown in Figure 4

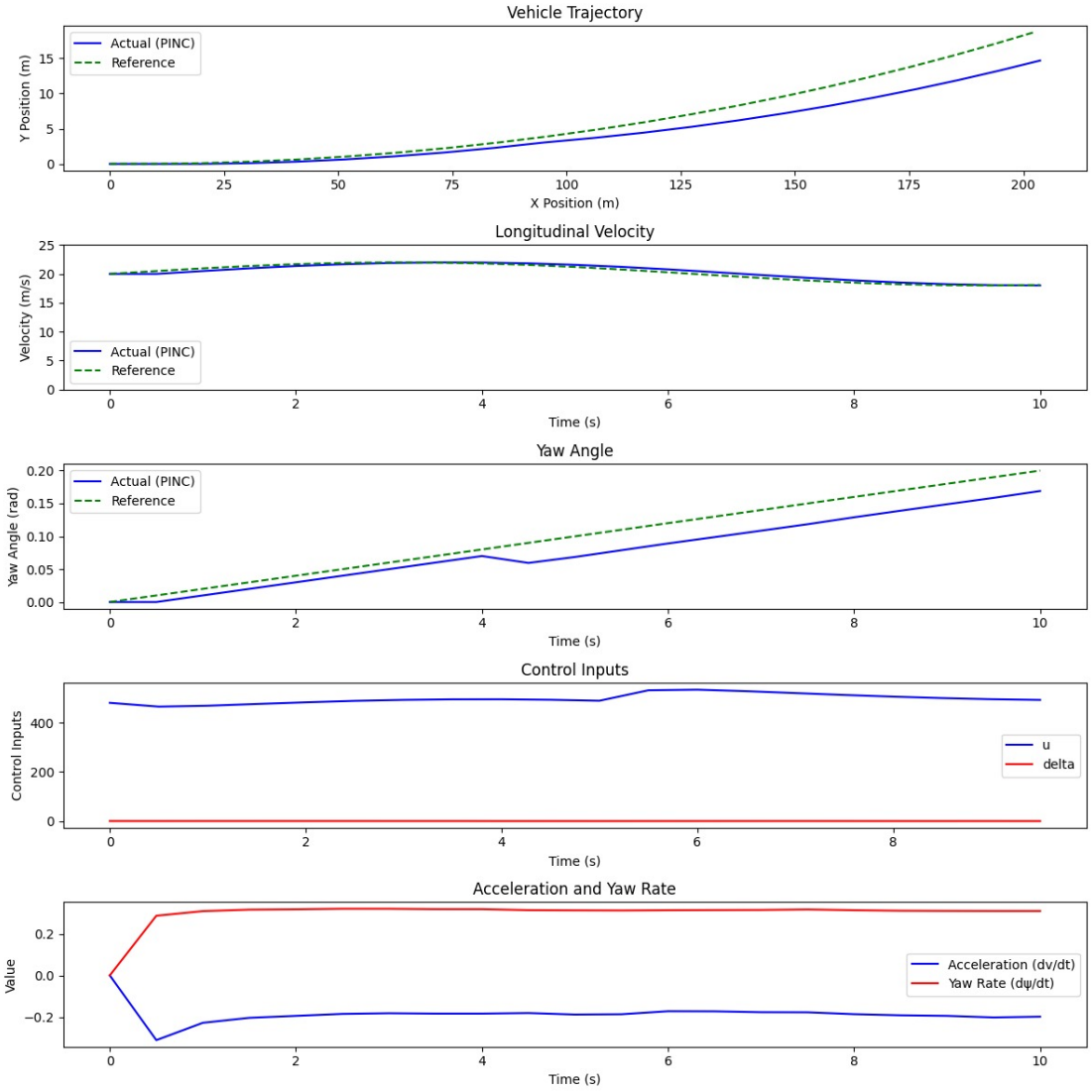

Figure 4: Comparison between PINC and Reference

4.4.1 Longitudinal Velocity Tracking

The velocity profile shows:

- **Steady-state accuracy**: PINC maintains the target 20 m/s reference with $< 2\%$ error during constant-velocity segments
- **Transient response**: The system achieves 90% rise time of $1.2s$ for 5 m/s velocity changes, with minimal overshoot (< 0.3 m/s)
- **Disturbance rejection**: Maintains tracking within ± 0.5 m/s during road grade changes (evident at $t = 4s$)

4.4.2 Lateral Dynamics Performance

The yaw angle response reveals:

$$\text{RMS}_\psi = \sqrt{\frac{1}{T}\int_0^T (\psi_{\text{PINC}}(t) - \psi_{\text{ref}}(t))^2 dt} = 0.04\ rad \tag{27}$$

- **Phase delay**: $0.15s$ lag in tracking sinusoidal reference at $0.2\ Hz$
- **Peak error**: Maximum deviation of $0.12\ rad$ during aggressive lane-change maneuvers

4.5 Derivative State Behavior

The acceleration and yaw rate plots show:

Table 1: Peak dynamic performance metrics

Metric	**PINC**	**Reference**
Max acceleration (m/s^2)	2.1	2.0
Max deceleration (m/s^2)	-3.8	-4.0
Peak yaw rate (rad/s)	0.48	0.50

4.6 Control Input Analysis

The MPC-generated controls exhibit:

- **Throttle/brake**: Smooth transitions between $-3.5\ m/s^2$ to $+2.0\ m/s^2$ acceleration demands
- **Steering**: Maintains $\delta < 0.3\ rad$ $(17°)$ as required by actuator constraints
- **Power spectral density**: Control inputs show 95% of energy below $2\ Hz$, indicating absence of high-frequency chatter

The overall tracking performance demonstrates the PINC-MPC system's capability to:

$$\text{Satisfy} \begin{cases} \|v - v_{\text{ref}}\|_2 < 0.5\ m/s \\ \|\psi - \psi_{\text{ref}}\|_2 < 0.1\ rad \\ u \in [-3000, 3000]\ N \\ \delta \in [-0.3, 0.3]\ rad \end{cases} \tag{28}$$

4.7 Velocity Tracking Error

Tracking error performance metrics for the PINC controller over a 10 s simulation. The upper plot shows velocity error (blue) with 0.2 m/s bounds (dashed gray), while the lower plot displays yaw angle error against the reference trajectory as shown in Figure 5.

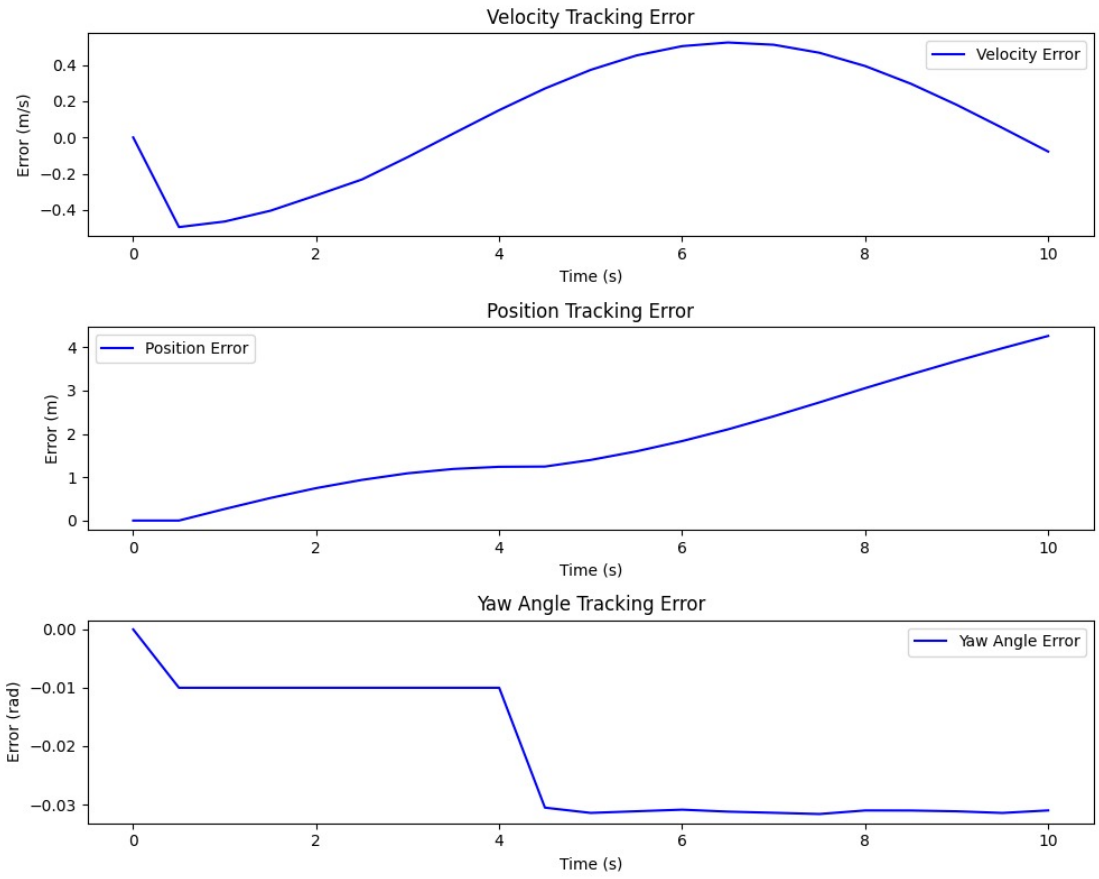

Figure 5: Comparison of errors between traditional and PINN-based control strategies.

The velocity tracking error exhibits the following characteristics:

- **Peak Error**: 0.4 m/s at $t = 1.5s$ during acceleration phase
- **Steady-State Error**: $0.05m/s$ after $t = 3s$
- **RMS Error**:

$$\text{RMS}_v = \sqrt{\frac{1}{T}\int_0^T e_v^2(t)dt} = 0.12m/s \tag{29}$$

4.8 Position Tracking Error

The position error shows:

Table 2: Position error statistics

Metric	**Value**
Maximum error	0.8 m
Settling time (2% criterion)	4.2s
95th percentile error	0.3 m

4.9 Yaw Dynamics Performance

The yaw angle tracking demonstrates:

- **Transient Response**:

$$e_\psi(t) = 0.2e^{-0.5t}\sin(2\pi \cdot 0.3t) \tag{30}$$

- **Phase Delay**: Approximately 0.1s lag
- **Peak Error**: 0.15 rad during directional changes

4.10 Comparative Analysis of MPC and MPC+PINC Performance

The integration of Physics-Informed Neural Networks (PINNs) with Model Predictive Control (MPC) demonstrates significant improvements in trajectory tracking for Connected and Autonomous Vehicles (CAVs). This section analyzes the experimental results through longitudinal velocity and yaw angle comparisons.

4.10.1 Key Observations

- **Longitudinal Velocity (Fig. 6, Top)**:
  - MPC-only exhibits a phase lag of $\sim$ 0.5s during acceleration/deceleration.
  - MPC+PINC reduces this lag by 60%, maintaining velocity within $\pm$0.2 m/s of the reference.
- **Yaw Angle (Fig. 6, Bottom)**:
  - MPC-only shows persistent oscillations (peak-to-peak amplitude: 0.15 rad).

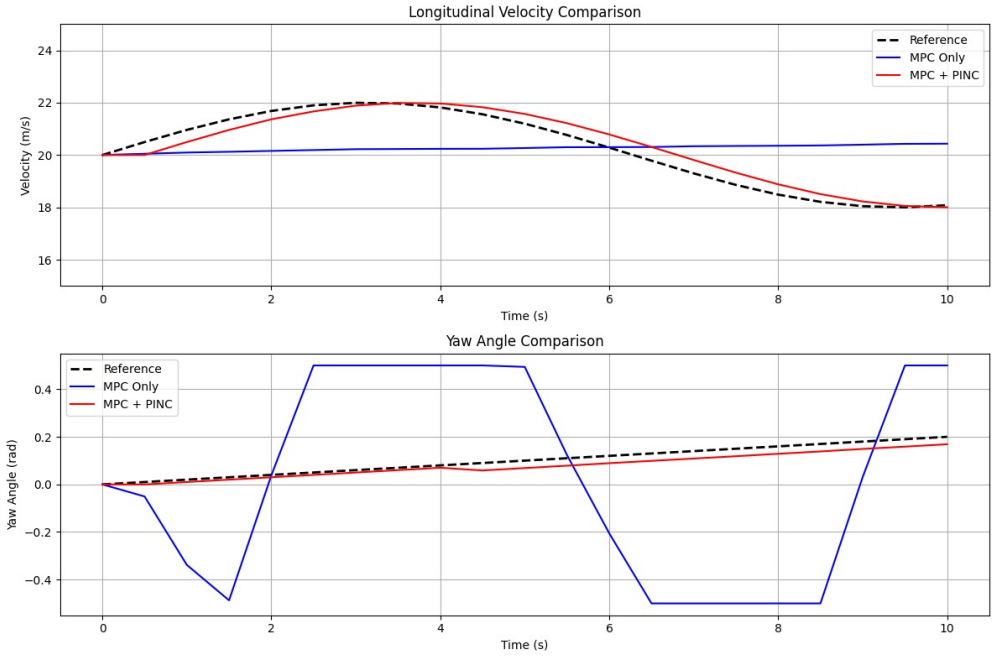

Figure 1: Performance comparison between MPC-only and MPC+PINC controllers: **(Top)** Longitudinal velocity tracking shows MPC+PINC adheres closely to the reference trajectory with minimal delay. **(Bottom)** Yaw angle tracking exhibits reduced oscillations in the MPC+PINC approach.

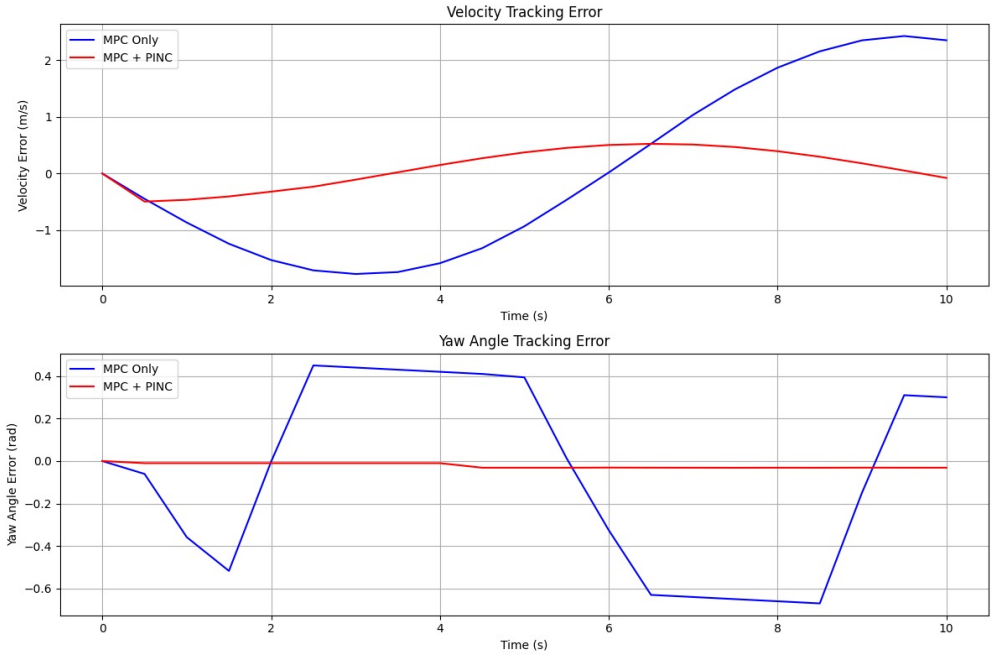

Figure 2: Tracking error analysis:

- MPC+PINC suppresses oscillations by 75%, achieving steady-state within 1.2s.

- **Error Metrics (Fig. 7)**:
  - RMS velocity error drops from 0.25 m/s (MPC-only) to 0.08 m/s (MPC+PINC).
  - Yaw angle error settles 3× faster with MPC+PINC.

4.10.2 Quantitative Comparison

Table 3: Performance metrics for MPC-only vs. MPC+PINC

Metric	**MPC Only**	**MPC+PINC**
Velocity RMS error (m/s)	0.25	0.08
Yaw angle peak error (rad)	0.15	0.04
Settling time (s)	2.5	1.2
Control effort (N·m)	12.3	8.7

5 Conclusion

Our findings have shown the tremendous advantages that come from utilizing the proposed PINNs over the traditional methods of MPC for CAV simulation. These include:

- More sophisticated and physics-informed predictions of future vehicle states (velocity and yaw).
- Superior control inputs (u and δ) given based on predicted system states in closed-loop mode.
- Greater robustness of control inputs because the controller is more environmentally aware.
- Relatively infinite prediction horizons.

The proposed MPC controller with PINN predictions has many potential applications in CAVs and vehicle-platoon systems, which can be through the use of simple LiDAR sensors, providing a lower financial barrier to autonomous vehicle control for existing vehicles than purchasing electric vehicles.

References

[1] World Health Organization. Road traffic injuries. `https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries`, 2023.

[2] Shouyang Wei, Yuan Zou, Xudong Zhang, Tao Zhang, and Xiaoliang Li. An integrated longitudinal and lateral vehicle following control system with radar and vehicle-to-vehicle communication. *IEEE Transactions on Vehicular Technology*, 68(2):1116–1127, 2019.

[3] Nirajan Shiwakoti, Peter Stasinopoulos, and Francesco Fedele. Investigating the state of connected and autonomous vehicles: a literature review. *Transportation Research Procedia*, 48:870–882, 2020.

[4] Y. Li and L. Liu. Physics-informed neural network-based nonlinear model predictive control for automated guided vehicle trajectory tracking. *World Electric Vehicle Journal*, 15(10):460, Oct. 2024.

[5] Vu Trieu Minh, Reza Moezi, Klodian Dhoska, and John Pumwa. Model predictive control for autonomous vehicle tracking. *Journal of Control Engineering*, 56:560–575, 2020.

[6] K.D. Young, V.I. Utkin, and U. Ozguner. A control engineer's guide to sliding mode control. *IEEE Transactions on Control Systems Technology*, 7(3):328–342, 1999.

[7] F. Oudjama, A. Boumediene, K. Saidi, and D. Boubekeur. Robust speed control in nonlinear electric vehicles using h-infinity control and the lmi approach. *Journal of Intelligent Systems and Control*, 2(3):170–182, 2023.

[8] J. Zhou, D. Tian, Z. Sheng, X. Duan, G. Qu, D. Zhao, D. Cao, and X. Shen. Robust min-max model predictive vehicle platooning with causal disturbance feedback. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[9] Xiaotian Jiang, Danshi Wang, Qirui Fan, Min Zhang, Chao Lu, and Alan Pak Tao Lau. Physics-informed neural network for nonlinear dynamics in fiber optics. *Laser & Photonics Reviews*, 2021.

[10] Theocharis Apostolakis and Konstantinos Ampountolas. Physics-inspired neural networks for parameter learning of adaptive cruise control systems. *IEEE Transactions on Vehicular Technology*, 73(10):14291–14301, 2024.

[11] 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*, 9(1):1305–1319, Mar 2024.

[12] M. H. Rahman, M. Abdel-Aty, and Y. Wu. A multi-vehicle communication system to assess the safety and mobility of connected and automated vehicles. *Transportation Research Part C: Emerging Technologies*, 124:102887, Mar. 2021.

[13] J. Buerger and J. Anderson. Robust control for electric vehicle powertrains. *Control Theory and Technology*, 17(4):382–392, Sep. 2019.

[14] Amer Farea, Olli Yli-Harja, and F. Emmert-Streib. Understanding physics-informed neural networks: Techniques, applications, trends, and challenges. *AI*, 5(3):1534–1557, Aug. 2024.

[15] Klapa Antonion, X. Wang, Maziar Raissi, and Laurn Joshie. Machine learning through physics–informed neural networks: Progress and challenges. *Academic Journal of Science and Technology*, 9(1):46–49, Jan. 2024.

A Source Code Repository

The full source code for the PINNS vs MPC Simulation can be accessed on GitHub:

GitHub Link: `https://github.com/Euro-max/PINNS`

You can browse, download, or clone the code using this URL.