Proceedings of 17th International Symposium on Advanced Vehicle Control (AVEC'26)
7-11 September 2026, Tsukuba, Japan

Vehicle Dynamics Identification and Control using Model-Structured Neural Networks

Gioele DEFRANCESCO [1)] Gastone Pietro ROSATI PAPINI [1)]

1) Department of Industrial Engineering, University of Trento
via Sommarive, 9 - 38123 Trento, ITALY
Corresponding author : gioele.defrancesco@unitn.it

Abstract: Vehicle dynamics modeling and control are challenging due to nonlinearities, uncertainty, and the trade-off between accuracy, interpretability, and computational cost. This work presents a unified framework for vehicle identification and lateral control based on Model-Structured Neural Networks (MS-NNs), which embed physical knowledge directly into the neural architecture. A structured lateral dynamics model is derived from vehicle handling analysis and trained on high-fidelity CarMaker telemetry to predict vehicle curvature. The identified model is then used as a differentiable plant to train a neural steering controller through recurrent optimization. The approach reduces complexity while maintaining interpretability and generalization. The model achieves a curvature prediction error of about $5 \times 10^{-3}\mathrm{m}^{-1}$, and the controller reaches a tracking error of $1.34 \times 10^{-3}\mathrm{m}^{-1}$. Exported to ONNX and validated in closed loop within CarMaker, the controller demonstrates improved curvature tracking compared with the reference controller under the considered case-study assumptions. These results suggest that MS-NNs provide an effective framework for vehicle identification and control with limited prior knowledge and a reduced number of trainable parameters.

Keywords: neural networks, model-structured neural networks, vehicle identification, vehicle control

1. Introduction

Accurate vehicle dynamics modeling and control are challenging due to nonlinearities, coupled lateral–longitudinal behavior, and varying operating conditions. Physics-based models, such as single- and double-track formulations, provide interpretability but require extensive parameter identification and costly experimental setups, while offering limited adaptability to changing conditions. Neural Network (NN) approaches enable data-driven identification of complex vehicle dynamics and have shown promising results in autonomous driving and racing [(4)]. However, general-purpose NNs are often black-box models, raising concerns on interpretability, physical consistency, and safety in automotive applications [(5)]. Model-Structured Neural Networks (MS-NNs) embed physical knowledge into the network architecture, combining learning flexibility with improved generalization and interpretability [(1,2)]. Although MS-NNs have been demonstrated to be an effective alternative for both system identification and control, their practical adoption remains limited, as their implementation within standard neural network frameworks is often complex and requires deep expertise in both vehicle modelling and neural network design to structure the architecture properly.

This paper presents a unified framework for vehicle dynamics identification and control using MS-NNs. The contributions include: (i) the synthesis of a structured neural model derived from vehicle dynamics equations; (ii) the use of the identified model for neural control synthesis and training; and (iii) the adoption of the open-source framework `nnodely` for design, training, and deployment. The approach is validated through simulation using data from a high-fidelity vehicle simulator (CarMaker).

1.1. Model-Structured Neural Networks

MS-NNs are neural architectures whose internal structure explicitly encodes domain knowledge through components derived from physical laws or control principles, introducing inductive biases that promote physically consistent learning. They belong to the broader class of model-based machine learning methods, which aim to integrate prior knowledge into data-driven models to improve generalization and interpretability. The effectiveness of MS-NNs has been demonstrated in automotive applications: [(3)] achieved a sideslip angle estimation error of 0.4°, outperforming recurrent NNs and classical observers, while an MS-NN reduced longitudinal acceleration RMSE to 0.105 m/s^2 compared to standard multilayer perceptrons [(1)].

Fig. 1: CarMaker simulation environment.

1.2. nnodely Framework

`nnodely`[1] is an open-source framework designed for the development of MS-NNs, providing a modular and domain-oriented interface that simplifies their design, training, validation and deployment. In the application below, the standard blocks of the framework are shown in bold. The `nnodely` code related to this work is reported in the repository[2].

[1] `https://github.com/tonegas/nnodely`
[2] `https://github.com/tonegas/nnodely-applications`

2. Vehicle Dynamics Control via Model-Structured Neural Networks

A two-step approach is adopted for the design of a lateral controller. First, a MS-NN forward model is identified from experimental data. Then, the controller is connected to the forward model and trained through episodic simulations, resulting ideally in a unitary plant. This strategy allows the designed controller to generalise beyond the training data while reducing the need for additional measurements, provided that the forward model is sufficiently accurate. Furthermore, the physical interpretability enhances robustness against overfitting.

2.1. Vehicle Lateral Dynamics Model

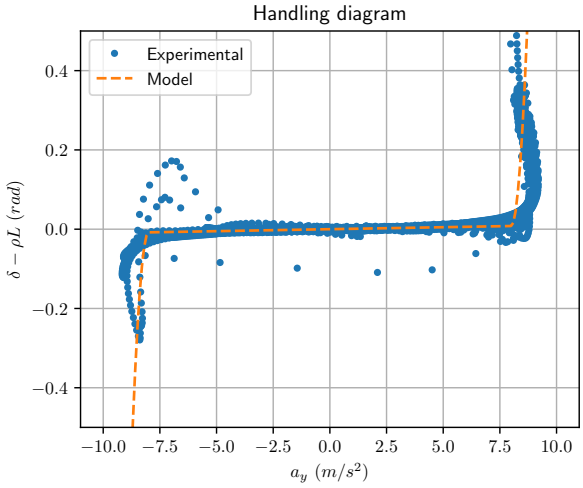

Fig. 2: Comparison of the experimental handling diagram of the vehicle and proposed model function with $K_{us_0} = 0.001, a_{y_0} = 8, n = 3$.

The structured neural network is inspired by the vehicle handling diagram (HD), defined as the difference between the actual steering angle and the kinematic steering angle, $\delta - \rho L$, where δ, ρ, and L denote the steering angle, steady-state curvature, and wheelbase, respectively. Since the shape of the handling diagram strongly depends on the vehicle characteristics (e.g., understeering or oversteering behavior), a preliminary analysis of the dataset was conducted to identify the dominant trends and formulate a network architecture tailored to the vehicle under consideration. The analyzed vehicle exhibited a pronounced understeering behavior, and the dataset covered the entire lateral dynamic range up to the saturation of the front axle. Under these conditions, the handling diagram showed a predominantly nonlinear dependence on lateral acceleration, while the influence of vehicle speed was comparatively limited. Based on these observations, a relationship linking the steady-state curvature to the steering angle can be derived as follows:

$$\rho(\delta, v_x, a_y) = \frac{\delta - f(a_y, v_x)}{L} \tag{1}$$

The proposed neural network predicts the vehicle curvature at the next time step ρ_{k+1} (5) using a window of $n = 20$ past steering values sampled at 20 Hz, $\boldsymbol{\delta}_k = [\delta_{k-n+1}, \ldots, \delta_k]$, together with current longitudinal speed v_k and lateral acceleration a_k. The model is structured as the product of a dynamic term (2), which captures the steering-to-curvature relationship through a speed-dependent **FIR filter** ($\boldsymbol{w}_\delta$ are the weights), and a **parametric function** (3). $\phi_{*i}(\cdot)$ denotes triangular **fuzzy** membership function. The number of channels is $N_v = 3$. The neural model's equations are:

$$D_v(v_k, \boldsymbol{\delta}_k) = \sum_{i=1}^{N_v} \phi_{v_i}(v_k)\, (\boldsymbol{w}_{\boldsymbol{\delta}_i}^\top \boldsymbol{\delta}_k), \tag{2}$$

$$F_v(a_k, v_k) = \sum_{i=1}^{N_v} \phi_{v_i}(v_k)\, F_v^i(a_k), \tag{3}$$

$$\begin{aligned} F_v^i(a_k) = {} & K_{us_0}^i a_k + \max(a_k - a_{y_0}^i, 0)^n \\ & - \max(-a_k - a_{y_0}^i, 0)^n \end{aligned} \tag{4}$$

$$\rho_{k+1} = D_v(v_k, \boldsymbol{\delta}_k) F_a(a_k, v_x). \tag{5}$$

where (4) represents the proposed model for the handling diagram of an understeering vehicle, as shown in Fig. 2. The total number of training parameters is 72.

2.2. Vehicle Steer Control

The controller receives two future windows of length $n = 20$ for the target velocity $\boldsymbol{v_{t}}_k$ and curvature $\boldsymbol{\rho_{t}}_k$, along with a past window of realized steering angles $\boldsymbol{\delta_{c}}_k = [\delta_{k-n+1}, \ldots, \delta_{k-1}]$ to account for initial steering conditions. A **parametric function** (7), parameterized by A, is introduced to compensate for understeer effects (here assumed to be linear in a_y), while a **FIR filter** (6) ($\boldsymbol{w}_\rho$) is used to capture the vehicle dynamic response. The predicted steering angle $\tilde{\delta}_k$ (8) is obtained as the sum of the contribution from the target trajectory and the past steering history.

$$K_t(a_k, \boldsymbol{v_{t}}_k, \boldsymbol{\rho_{t}}_k) = \sum_{i=1}^{N_v} \phi_{v_i}(v_k)\, (\boldsymbol{w}_{\rho_i}^\top \nabla(a_k, \boldsymbol{v_{t}}_k, \boldsymbol{\rho_{t}}_k)) \tag{6}$$

$$\nabla(a_k, \boldsymbol{v_{t}}_k, \boldsymbol{\rho_{t}}_k) = \boldsymbol{\rho_{t}}_k \odot \left(1 + A\, \boldsymbol{v_{t}}_k^2\right) \tag{7}$$

$$\tilde{\delta}_k = K_t(a_k, v_k, \boldsymbol{v_{t}}_k, \boldsymbol{\rho_{t}}_k) + \boldsymbol{w_{\delta_c}}^\top \boldsymbol{\delta_{c}}_k \tag{8}$$

The controller has a total of 39 learnable parameters.

2.3. Training and Validation

The datasets used for training and validation were generated in CarMaker using a 14-DoF electric vehicle model and collected over several circuits, including both synthetic tracks and real racing circuits such as Hockenheim. The selected scenarios span a wide range of curvature profiles and vehicle operating conditions, including maneuvers performed close to the handling limits. This diversity was intentionally introduced to expose the networks to the vehicle's nonlinear characteristics over the entire operating envelope. A total of 7,868 samples were collected and subsequently divided into training (70%), validation (20%), and test (10%) subsets. The test set was used exclusively for the final performance evaluation, while the validation set was employed for hyperparameter tuning and for monitoring the training process through an early-stopping criterion.

The forward model was trained to predict the steady-state vehicle curvature. Since curvature is directly related to yaw rate and lateral acceleration through kinematic relationships, it represents a compact quantity for characterizing the vehicle response. The model was trained for 48 epochs, until the early-stopping criterion was reached, using the Adam optimizer with a learning rate of 10^{-3}. Batch sizes of 16 and 64 samples were adopted for training and validation, respectively. The model accuracy was evaluated on the test dataset using the root mean squared error (RMSE). The resulting curvature prediction error was approximately 5×10^{-4} m^{-1}, corresponding to a lateral acceleration error of approximately 0.24 m/s^2. Figure 3 reports a comparison between the predicted and reference values,

highlighting the capability of the network to accurately reproduce the steady-state vehicle behavior across the considered operating range.

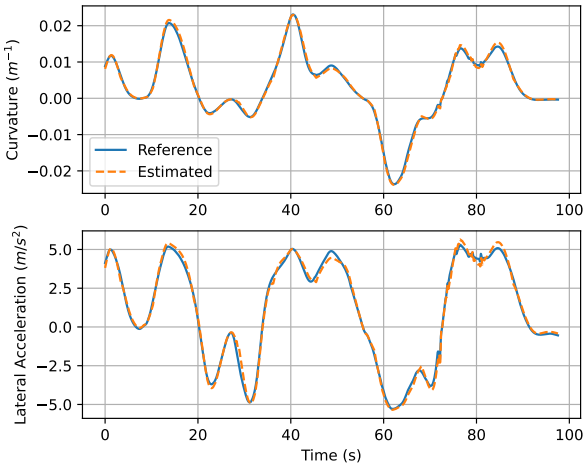

Fig. 3: Comparison between the trained forward model and telemetry data on the test set.

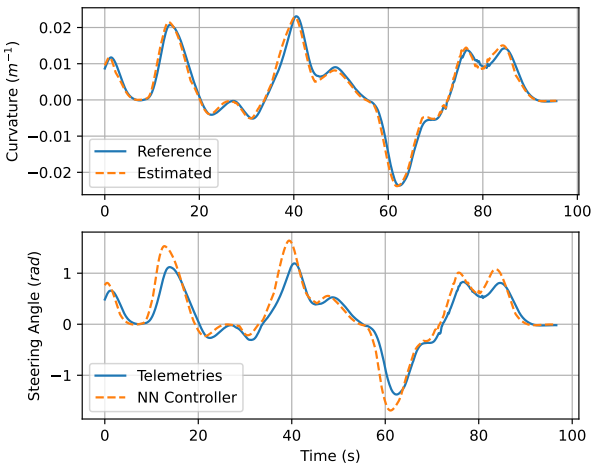

Fig. 4: Estimated curvature obtained by feeding the controller steering output into the trained forward model, and comparison between the controller-generated and telemetry steering angles.

The controller was trained by connecting its output to the previously identified forward model and minimizing the error on the predicted curvature. This choice was found to be more robust than directly minimizing the error on the steering angle. Indeed, when the vehicle operates close to the front-axle saturation limit, the relationship between steering angle and generated lateral acceleration becomes highly nonlinear and may even locally reverse, such that an increase in steering angle results in a reduction of the achievable curvature. Learning this behavior directly through a steering-based loss is unnecessary and may adversely affect the training process. By evaluating the loss on the resulting curvature, the controller is instead optimized with respect to the actual vehicle response. Since the forward model already captures the nonlinear steady-state characteristics and the physical limits of the vehicle, the controller can maintain a relatively simple structure while still achieving accurate performance over the entire operating range. Since the controller model has a branch depending on a window of past steering angles, the controller was trained recurrently. The training process was divided into two stages to progressively increase the prediction horizon and improve convergence. During the first stage, the controller was trained for 5 epochs using a prediction horizon of 5 samples. Subsequently, the prediction horizon was increased to 30 samples, and the training continued for an additional 10 epochs. The controller was optimized using the Adam optimizer with a learning rate of 10^{-2}. The same batch sizes adopted for the forward model were employed, namely 16 samples for training and 64 samples for validation. On the test dataset, the controller achieved an RMSE of 1.34×10^{-3} m^{-1}. Figure 4 presents the curvature tracking performance of the proposed approach and the comparison between the steering angle generated by the controller and the reference steering input.

2.4. Results

To test the performance of the controller, the network was exported to ONNX and evaluated within the CarMaker environment by applying the controller on top of the vehicle model. Since a planner was not included at this stage, the vector of future curvature values provided to the controller was populated using the future road curvature. A dedicated test circuit was designed such that the road curvature variations remained compatible with the vehicle's dynamic limits. To enable a direct comparison with CarMaker curvature, the driver configuration was set with a corner-cutting coefficient equal to 0, forcing the vehicle to follow the centerline of the road.

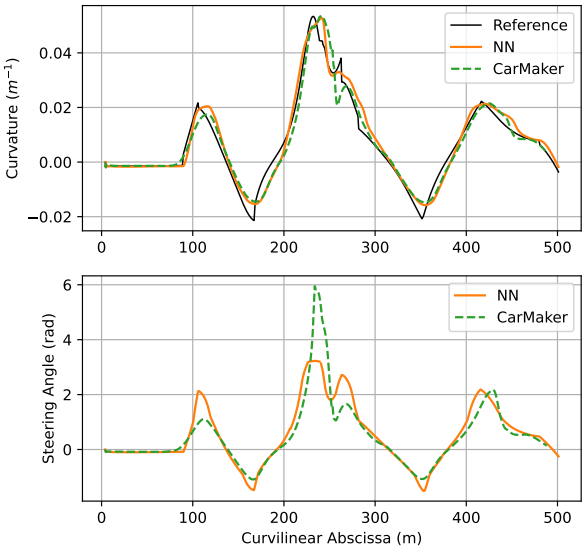

Fig. 5: Comparison of the neural lateral controller and the CarMaker controller on a test-track snapshot. The upper plot shows reference and achieved curvature, while the lower plot compares steering angles.

Figure 5 reports the curvature and steering angle obtained with the two controllers, while the curvature tracking error along the entire test track is shown in Fig. 6. The proposed neural controller is able to accurately track the desired curvature throughout the considered maneuver. The snapshot in Fig. 5 highlights the different behavior of the two approaches: the trained controller achieves the target curvature while maintaining a moderate steering demand, whereas the original CarMaker controller requires significantly larger steering angles. This behavior suggests that the

latter operates closer to the front-axle saturation region, while the proposed controller exploits the learned vehicle model to generate steering commands that more effectively utilize the available tire force, thereby achieving the desired curvature without excessive steering inputs. This qualitative observation is confirmed by the tracking performance over the entire test track, where the proposed controller achieved a curvature RMSE of 3.4×10^{-3} m^{-1}, compared to 6.8×10^{-3} m^{-1} for the CarMaker controller.

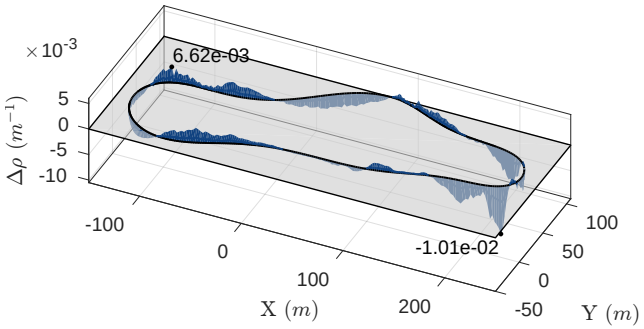

Fig. 6: Curvature tracking error $\Delta\rho$ along the test track.

3. Conclusion

This paper highlights the advantages of adopting an MS-NN architecture for vehicle control. With a limited number of parameters and only partial knowledge of the vehicle dynamics, it is possible to train a robust and effective controller. Moreover, this approach improves model interpretability by embedding structural knowledge into the learning process. In particular, it was shown that, starting from a simplified analysis of the considered model, it is possible to impose a meaningful structure on the problem and significantly reduce the number of parameters required to represent behaviors that would otherwise be highly complex to model directly.

Future work will focus on extending the proposed framework to more complex vehicle dynamics representations. In particular, models exhibiting oversteering behavior introduce instability and therefore require more advanced modeling and control strategies. At the same time, the results obtained suggest that robust controllers can still retain relatively simple structures despite the increased system complexity. A further extension of this work would involve the development of a longitudinal vehicle model to complement the lateral dynamics model. The integration of both components could enable the construction of a digital twin of the vehicle, suitable for long-term prediction and for the estimation of unobservable states and system parameters.

Acknowledgments

This work was supported by the Italian FIS 2 Call, Grant Assignment Decree No. 1236 adopted on 01/08/2023 by the Italian Ministry of University and Research (MUR), for the project FIS-2023-03684 "Structured neural network framework for modeling and control of autonomous systems - Neu4mes", CUP E53C24003800001.

References

(1) M. Da Lio, D. Bortoluzzi, and G. P. Rosati Papini. Modelling longitudinal vehicle dynamics with neural networks. *VSD*, 58(11):1675–1693, 2020.

(2) M. Da Lio, R. Donà, G. P. Rosati Papini, F. Biral, and H. Svensson. A mental simulation approach for learning neural-network predictive control (in self-driving cars). *IEEE Access*, 8:192041–192064, 2020.

(3) M. Da Lio, M. Piccinini, and F. Biral. Robust and sample-efficient estimation of vehicle lateral velocity using neural networks with explainable structure informed by kinematic principles. *IEEE Trans. ITS*, 24(12):13670–13684, 2023.

(4) L. Hermansdorfer, R. Trauth, J. Betz, and M. Lienkamp. End-to-end neural network for vehicle dynamics modeling. In *2020 6th IEEE Congress on Information Science and Technology (CiSt)*, pages 407–412, 2020.

(5) S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah. A survey of deep learning applications to autonomous vehicle control. *Trans. ITS*, 22(2):712–733, 2020.