Model-Structured Neural Networks to Control the Steering Dynamics of Autonomous Race Cars

Mattia Piccinini[1*], Aniello Mungiello[2*], Georg Jank[3], Gastone Pietro Rosati Papini[4], Francesco Biral[4], Johannes Betz[1]

***Abstract*— Autonomous vehicle racing has attracted increasing attention in recent years, as a safe and controlled environment to accelerate the development of autonomous driving. This has led to the development of advanced algorithms in vehicle perception, motion planning, and control. Deep learning models, predominantly based on neural networks (NNs), have demonstrated significant potential in modeling the vehicle dynamics and in performing various tasks in autonomous driving. However, their black-box nature is critical in the context of autonomous racing, where safety and robustness demand a thorough understanding of the decision-making algorithms. To address this challenge, this paper proposes MS-NN-steer, a new Model-Structured Neural Network for vehicle steering control, integrating the prior knowledge of the nonlinear vehicle dynamics into the neural architecture. The proposed controller is validated using real-world data from the Abu Dhabi Autonomous Racing League (A2RL) competition, with full-scale autonomous race cars. In comparison with general-purpose NNs, MS-NN-steer is shown to achieve better accuracy and generalization with small training datasets, while being less sensitive to the weights' initialization. Also, MS-NN-steer outperforms the steering controller used by the A2RL winning team. Our implementation is available open-source in the following repository `https://github.com/tonegas/nnodely-applications/tree/main/vehicle/control_steer_dynamics_A2RL`.**

***Index Terms*— Autonomous racing, vehicle control, vehicle dynamics, neural networks, explainable artificial intelligence.**

I. Introduction

In the last 10 years, the field of autonomous driving has witnessed a continuous growth, as highlighted by the increasing number of publications addressing the wide range of tasks required for autonomous vehicle operation. Despite this progress, a substantial portion of the literature remains focused on standard urban driving conditions. While these are essential for real-world deployment, they do not fully capture the challenges that arise under extreme or safety-critical conditions, where the vehicle operates near its handling limits.

Fig. 1. Autonomous race car of the Technical University of Munich (TUM) at the 2024 Abu Dhabi Autonomous Racing League (A2RL) competition, on the Yas Marina circuit.

To address this gap, the domain of autonomous racing has gained increasing attention in recent years. Analogous to the role of Formula 1 in the evolution of conventional automotive technologies, autonomous racing serves as a high-performance testbed for the development and validation of advanced algorithms in perception, planning, and control [1]. As a vehicle operates near its handling limits, its dynamics become highly nonlinear and hard to predict, necessitating advanced nonlinear controllers to execute racing maneuvers. Additionally, the vehicle's dynamics are influenced by various factors, including tire characteristics, suspension systems, and aerodynamic effects, which can change over time due to wear, tire temperature and weather conditions. Thus, using traditional physics-based models for vehicle dynamics control is often impractical, as they require precise knowledge of the vehicle parameters [2]. On this account, an increasing number of studies have focused on data-driven approaches, particularly deep learning techniques [3], yet with several limitations in generalizing to unseen scenarios [4]. Let us now critically analyze the related work, to later highlight the contributions of this paper.

A. Related Work

Deep neural networks (NNs) have demonstrated significant potential in vehicle dynamics modeling [5], [6] and control [3], [4], [7]–[9]. However, large-scale NNs typically require extensive training datasets to perform reliably in unseen scenarios [3], [4]. Indeed, the general-purpose internal structure of NNs enables them to approximate a wide range of systems' behaviors, but at the price of requiring many trainable parameters and data. Moreover, deep NNs are often perceived as black boxes [4], making it difficult to assign

This work was partly funded by the Ministero Università e Ricerca (MUR), Italy, through the PNRR project Centro Nazionale per la Mobilità Sostenibile (CNMS) CUP E63C22000930007.

*These authors contributed equally to this work.

[1]Professorship of Autonomous Vehicle Systems, Technical University of Munich, 85748 Garching, Germany; Munich Institute of Robotics and Machine Intelligence (MIRMI), name.surname@tum.de

[2]Department of Information Technology and Electrical Engineering (DIETI), University of Naples Federico II, Naples 80125, Italy, aniello.mungiello@unina.it

[3]Chair of Automatic Control, Department of Mechanical Engineering, TUM School of Engineering and Design, Technical University of Munich, 85748 Garching, Germany, georg.jank@tum.de

[4]Department of Industrial Engineering, University of Trento, 38123 Trento, Italy, name.surname@unitn.it

Manuscript 261 submitted to 2025 IEEE 28th International Conference on Intelligent Transportation Systems (ITSC 2025). Received May 1, 2025.

physical meaning to their parameters and outputs. This is particularly critical in the context of autonomous driving, where safety and robustness demand a proper understanding of the underlying decision-making algorithms.

To address these concerns, recent research has begun to focus on enhancing the interpretability of deep learning models. For instance, the authors of [10] introduced a physics guard layer to constrain the learned parameters within physical bounds. Similarly, [11] combined physical prior knowledge with data-driven components for improved modeling of the vehicle's lateral dynamics.

Following a similar research line, the authors of [12]–[16] introduced Model-Structured Neural Networks (MS-NNs) to embed the vehicle dynamics laws inside their architectures, for modeling, control and estimation of the vehicle dynamics. Compared to general-purpose NNs, these physics-driven models showed enhanced interpretability and generalization.

To the best of our knowledge, the existing examples of NN-based steering controllers for autonomous vehicles are limited by at least one of the following:

- MS-NNs were used as steering controllers in simulation [14], but not with real-world data;
- The neural steering controllers had general-purpose architectures and required large training sets [3], [4];
- The neural steering controllers were conceived for urban driving and not for high-performance racing [13];
- Absence of open-source frameworks to easily replicate the proposed implementations.

B. Contributions

To address the previous limitations, this paper's contributions are the following:

- We present a new model-structured neural network for vehicle steering control, named MS-NN-steer, by extending the work of [14]. MS-NN-steer explicitly models the influence of the vehicle velocity on the steering characteristics: we show that this is a non-negligible effect in high-performance racing.
- We train and validate our MS-NN-steer using real-world data from the 2024 Abu Dhabi Autonomous Racing League (A2RL) competition.
- The proposed MS-NN-steer is compared against three benchmarks: the steering controller used by the winning team of the A2RL competition, a general-purpose neural network (G-NN), and the baseline MS-NN of [14].
- We release open-source our code to generate customized MS-NN architectures and to perform hyperparameters optimization.

Finally, the paper is organized as follows. The design of the proposed feedforward steering control is detailed in Section II, while the benchmarks are in Section III. The analysis of the results is performed in Section IV, and Section V draws the conclusions.

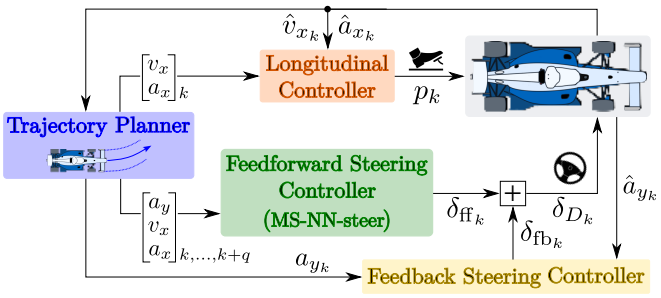

Fig. 2. Trajectory planning and control framework on the A2RL race car. The MS-NN-steer of this paper is a feedforward steering controller (green block), computing the steering angle to execute the planned trajectories.

II. Steering Control Design

A. Overview

Fig. 2 overviews the trajectory planning and control framework of the TUM Autonomous Motorsport software stack [17], which was employed in the A2RL competition. In this architecture, the trajectory planner uses a Tube Model Predictive Controller (Tube-MPC) to optimize the vehicle maneuver on a receding horizon. The planned trajectory is then executed by feedforward-feedback steering controllers and a longitudinal controller.

This paper develops a new neural network-based feedforward steering controller. The main methodological contribution is the design of a Model-Structured Neural Network (MS-NN), which integrates prior knowledge of the nonlinear vehicle dynamic laws into its architecture.

We consider two incremental versions of our MS-NN. The first version, named MS-NN-base and derived from [14], is our baseline, which models the combined lateral vehicle dynamics considering the effect of the longitudinal acceleration a_x. The second variant, MS-NN-steer, is a new extension of MS-NN-base and captures the influence of the longitudinal speed v_x on the vehicle's steering characteristics.

B. MS-NN: Inputs and Output

As shown in Fig. 2, our MS-NN computes the steering angle δ_k, where $k \in \mathbb{N}$ is the index of the current time step. The MS-NN's inputs are the trajectories planned by the Tube-MPC. Specifically, the inputs are windows of future planned values for the lateral and longitudinal accelerations $\{a_y, a_x\}$, and the longitudinal speed v_x:

$$\begin{cases} \boldsymbol{a_y}_k = \left[a_{y_k}, \dots, a_{y_{k+q}}\right] & \text{(1a)} \\ \boldsymbol{a_x}_k = \left[a_{x_k}, \dots, a_{x_{k+q}}\right] & \text{(1b)} \\ \boldsymbol{v_x}_k = \left[v_{x_k}, \dots, v_{x_{k+q}}\right] & \text{(1c)} \end{cases}$$

where q is the number of future time steps, the time step size is T, and both q and T are tunable parameters.

C. MS-NN-base: Baseline Version

The baseline version of our MS-NN, named MS-NN-base, is derived from [14], as is here briefly recalled. The internal architecture of MS-NN-base is shown in Fig. 4, *without* the

red term $k_y(v_x)$ inside the function $G(\cdot)$. This model is expressed as:

$$\delta_k = \sum_{j=1}^{n_v} \sum_{l=1}^{n_x} \Big(\mathbf{G}(\boldsymbol{a_{y}}_k, \boldsymbol{v_x}_k, \boldsymbol{a_x}_k) \odot \boldsymbol{\phi}_{jl}(\boldsymbol{v_x}_k, \boldsymbol{a_x}_k) \Big) \cdot \begin{bmatrix} F_{jl_1} \\ \vdots \\ F_{jl_{q+1}} \end{bmatrix} \tag{2}$$

where $\odot$ denotes the Hadamard element-wise product. The structure of MS-NN-base consists of two main parts. The first part learns the *quasi steady-state* lateral vehicle dynamics, while the second part captures the transient dynamics.

1) Quasi Steady-State Vehicle Behavior: The first part of MS-NN-base is the vector function $\mathbf{G}(\cdot)$ (blue block in Fig. 4), which is designed to learn the *quasi steady-state* lateral dynamics of the vehicle. The term *quasi* refers to the fact that the vehicle's dynamics are not strictly steady-state, as the longitudinal acceleration a_x can be non-zero. As shown in its expanded view in Fig. 4, $\mathbf{G}(\cdot)$ resembles a neuro-fuzzy system [18], [19], and is composed of $n_y \cdot n_x$ local models, each of which learns the dynamics in a predefined range of a_y and a_x.

To describe the internal design of $\mathbf{G}(\cdot)$, we first recall the handling diagram (HD), depicted in Fig. 3. The HD is a representation of a vehicle's steady-state nonlinear understeering and oversteering characteristics. It is generated by plotting the solution to the following equation:

$$\delta - \rho L = \delta - \frac{a_y}{v_x^2} L = \alpha_1 - \alpha_2 \tag{3}$$

where L is the vehicle wheelbase, and $\{\alpha_1, \alpha_2\}$ are the side slip angles of the front and rear tires. The HD quantifies the mismatch between the real steering angle δ (the average angle at the front wheels) and the kinematic steering angle ρL, with $\rho = a_y / v_x^2$ being the trajectory curvature.

As depicted in Fig. 4, the function $\mathbf{G}(\cdot)$ in (2) consists of local neuro-fuzzy models $g_{il}(\cdot)$, where $i \in \{1, \dots, n_y\}$ and $l \in \{1, \dots, n_x\}$ are the indices of the local models. To design $g_{il}(\cdot)$ using (3), we need a relation between $\{\alpha_1, \alpha_2\}$ and the accelerations $\{a_y, a_x\}$ in the vicinity of a local model's center $\{a_y = a_{y_{0_i}}, a_x = a_{x_{0_l}}\}$. As shown in [14], this relation can be obtained by locally approximating the equations of a nonlinear double-track vehicle model, around $\{a_y = a_{y_{0_i}}, a_x = a_{x_{0_l}}\}$. After performing the mathematical derivation outlined in [14], the steering angle δ is locally expressed as follows:

$$\begin{aligned} & g_{il}(a_y, v_x, a_x) \\ & = \frac{a_y}{v_x^2} L + k_{y1_i} \text{sign}(a_y) + k_{y2_i}(a_y - a_{y,0_i} \text{sign}(a_y)) + \\ & k_{y3_i} k_{x1_l} \cdot (a_y - (a_{y0_i} + k_{y4_i}) \text{sign}(a_y))(k_{x2_l} + a_x + a_{x0_l}) \cdot \\ & [1 + k_{y5_i}(a_y - a_{y0_i} \text{sign}(a_y)) \\ & + k_{x3_l} \cdot (a_x - a_{x0_l}) + k_{x4_l}(a_x - a_{x0_l})^2 + \\ & k_{y6_i} k_{x5_l}(a_y) - a_{y0_i} \text{sign}(a_y))(a_x - a_{x0_l})] \end{aligned} \tag{4}$$

The function $\mathbf{g}_{il}(\boldsymbol{a_{y}}_k, \boldsymbol{v_x}_k, \boldsymbol{a_x}_k)$ in Fig. 4 is the vector form of (4), which is omitted for brevity. Each of the local models $g_{il}(\cdot)$ has $6 + 5 = 11$ learnable parameters, namely the set $\{k_{y1_i}, \dots, k_{y6_i}, k_{x1_l}, \dots, k_{x5_l}\}$.

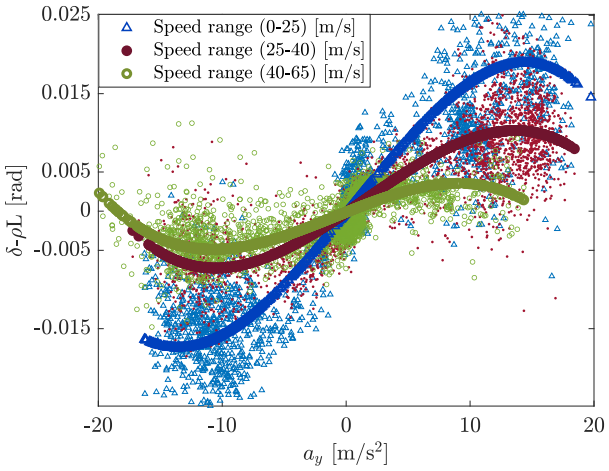

Fig. 3. Handling diagram (HD) derived from the real telemetry data of the A2RL car. The plot highlights the dependency of the HD on the vehicle speed v_x, which is a crucial physical phenomenon that we aim to capture in our extended MS-NN-steer architecture. The solid lines are polynomial fits of the data points.

Each local model $\mathbf{g}_{il}(\cdot)$ is activated by a vector function $\boldsymbol{\phi}_{il}(\boldsymbol{a_{y}}_k, \boldsymbol{a_x}_k) = \boldsymbol{\phi}_i(\boldsymbol{a_{y}}_k) \odot \boldsymbol{\phi}_l(\boldsymbol{a_x}_k)$ with $i \in \{1, \dots, n_y\}$ and $l \in \{1, \dots, n_x\}$ (see Fig. 4).

$\mathbf{G}(\boldsymbol{a_{y}}_k, \boldsymbol{v_x}_k, \boldsymbol{a_x}_k)$ returns a vector of $q+1$ future steady-state steering angle predictions.

2) Transient Vehicle Dynamics: From the classical theory of vehicle dynamics [20], the transient lateral dynamics depend on the vehicle speed v_x and acceleration a_x. In MS-NN-steer, these dynamics are captured by the fully connected layers $\boldsymbol{F}_{jl}$ (green block in Fig. 4), which learn linear combinations of the $\mathbf{G}(\cdot)$ vector entries across different ranges of v_x and a_x. This is done using the activation functions $\boldsymbol{\phi}_{jl}(\boldsymbol{v_x}_k, \boldsymbol{a_x}_k)$, where $l \in \{1, \dots, n_x\}$, $j \in \{1, \dots, n_v\}$, and n_v is the number of local models in the range of v_x.

D. MS-NN-steer: Extended Version

The methodological novelty of this work lies in the improvement of the $g_{il}(\cdot)$ function in Fig. 4 and equation (4). To illustrate this, let us examine the experimental handling diagram shown in Fig. 3, which displays three different third-degree polynomial fitted curves corresponding to distinct speed ranges. The shape of the handling diagram varies as a function of the vehicle speed, both in the linear region (i.e., for low a_y) and in the nonlinear region. However, this speed-dependent behavior was not accounted for in the MS-NN-base of Section II-C.

To address this, we introduce an extended version of the local models $g_{il}(\cdot)$ in (4), where some of the parameters k_y become now functions of the speed. Specifically, to capture the effect of v_x, we extend the local models by replacing the parameters $\{k_{y1_i}, k_{y2_i}\}$ with learnable

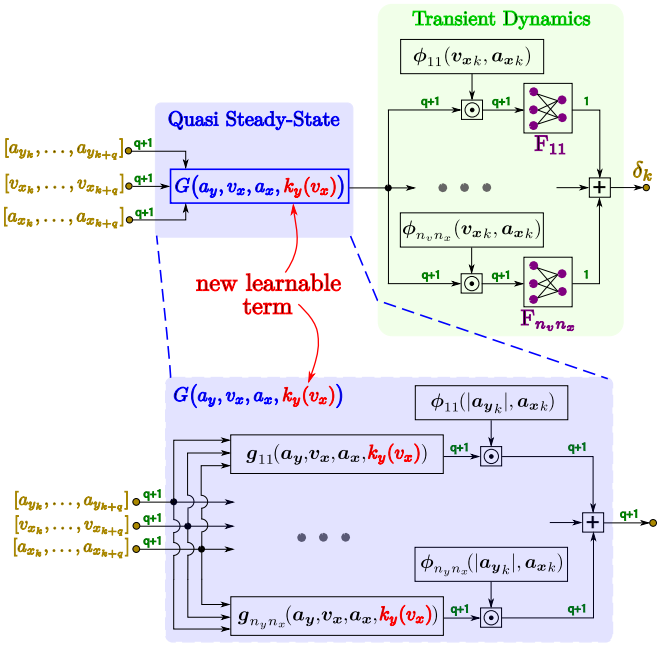

Fig. 4. Proposed MS-NN-steer architecture. The red learnable term $k_y(v_x)$ is part of the novelty of this work, as it captures the influence of the vehicle speed v_x on the steering characteristics and the handling diagram (Fig. 3).

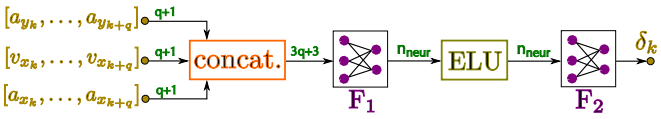

Fig. 5. General-purpose neural networl (G-NN) used as a benchmark.

functions $\{k_{y1_i}(v_x), k_{y2_i}(v_x)\}$:

$$
\begin{aligned}
& g_{il}\big(a_y, v_x, a_x, \overbrace{k_y(v_x)}^{\text{new term}}\big) \\
&= \frac{a_y}{v_x^2} L + \overbrace{k_{y1_i}(v_x)}^{\text{new term}} \operatorname{sign}(a_y) + \overbrace{k_{y2_i}(v_x)}^{\text{new term}} (a_y - a_{y,0_i}\operatorname{sign}(a_y)) + \\
& k_{y3_i} k_{x1_l} \cdot (a_y - (a_{y0_i} + k_{y4_i})\operatorname{sign}(a_y))(k_{x2_l} + a_x + a_{x0_l}) \cdot \\
& \qquad [1 + k_{y5_i}(a_y - a_{y0_i}\operatorname{sign}(a_y)) \\
& \qquad + k_{x3_l} \cdot (a_x - a_{x0_l}) + k_{x4_l}(a_x - a_{x0_l})^2 + \\
& \qquad k_{y6_i} k_{x5_l}(a_y) - a_{y0_i}\operatorname{sign}(a_y))(a_x - a_{x0_l})] \qquad (5)
\end{aligned}
$$

The functions $k_{y1_i}(v_x)$ and $k_{y2_i}(v_x)$ in (5) are modeled as learnable polynomials of the form:

$$
k_{y1i}(v_x) = \sum_{s=0}^{n_{p_1}} c_{1_{is}} v_x^s \qquad k_{y2_i}(v_x) = \sum_{w=0}^{n_{p_2}} c_{2_{iw}} v_x^w
$$

where $c_{1_{is}}, c_{2_{iw}}$ are learnable parameters. The polynomial degrees $\{n_{p_1}, n_{p_2}\}$ for $k_{1_i}(v_x)$ and $k_{2_i}(v_x)$ are optimized through experimental validation, which results in $n_{p_1} = 3$ and $n_{p_2} = 1$.

III. Benchmarks

Besides the MS-NN-base of [14], we benchmark our new MS-NN-steer against the following techniques.

A. General-Purpose Neural Network

The first benchmark is a general-purpose neural network (G-NN), shown in Fig. 5. G-NN has the same inputs and output as our MS-NN-steer. The input vectors are first concatenated and then passed through a two-layer feedforward NN, with an ELU (exponential linear unit) activation function. The first layer has n_{neur} neurons, while the second layer has 1 neuron, which produces the scalar output δ_k. The total number of learnable parameters in G-NN nearly matches the one of our MS-NN-steer. This allows for a fair comparison, and will highlight the performance and generalization advantages of MS-NNs.

B. Baseline A2RL Steering Controller

As an additional benchmark, we use the feedforward steering controller of the TUM Autonomous Motorsport software stack [17] that won the A2RL competition in 2024 (Fig. 1). This controller, hereafter named A2RL-control, computes the steering angle δ_k as a sum of four terms:

$$
\delta_k = \delta_{\text{kin}}(a_{y_k}, v_{x_k}) + \delta_{\text{us}}(a_{y_k}) + \delta_{\text{ax}}(a_{y_k}, a_{x_k}) + \delta_{\text{off}} \qquad (6)
$$

where $\delta_{\text{kin}} = a_y\, L / v_x^2$ is the kinematic steering angle. δ_{us} considers the understeering behavior and uses a first-order dynamics to produce an additional steering angle $\delta_{\text{us}_k} = \delta_{\text{us}_{k-1}} + (K_{\text{us}}\, a_{y_k} - \delta_{\text{us}_{k-1}}) \frac{\Delta t}{T_{\text{us}}}$, where K_{us} is a tunable understeering gradient, Δt is the sampling time, and T_{us} is the tunable time constant of the first-order dynamics. δ_{ax} is a compensation term that accounts for the weight transfers and aerodynamic effects, and is modeled as $\delta_{\text{ax}_k} = \big(\delta_{\text{ax}_{k-1}} + (K_{\text{ax}}\, a_{x_k} - \delta_{\text{ax}_{k-1}}) \frac{\Delta t}{T_{\text{ax}}}\big)\, a_{y_k}$, where K_{ax} and T_{ax} are tunable parameters. Finally, δ_{off} is a static offset due to asymmetries in the car setup.

IV. Results with Experimental Data

A. Real-World Autonomous Race Car Dataset

1) A2RL Autonomous Race Car: The test platform, EAV24, is an autonomous open-wheel race car based on a Dallara Super Formula SF23 chassis. The telemetry data used in this paper was collected during the 2024 Yas Marina Event of the A2RL racing series. This was an autonomous race between eight teams running their cars in single-vehicle fastest laps and multi-vehicle racing sessions, with speeds up to 260 km/h. The training datasets of this paper contain the velocity and acceleration signals computed by a state estimator, which fuses IMU, GNSS, LiDar, and Radar sensor signals [17]. The steering angle signal is measured by the steer-by-wire system of the car.

2) Training, Validation, and Test Datasets: Fig. 6 shows the racetrack layout with the sectors considered for the generation of our datasets. The data in this paper originated from two pre-qualifying laps.

As reported in Table I, the training set uses the first lap, while the validation set uses the second one. The reference speed was progressively increased from the first to the second lap, and the vehicle had different tire sets in each lap. Thus, the validation set is used to assess the generalization potential

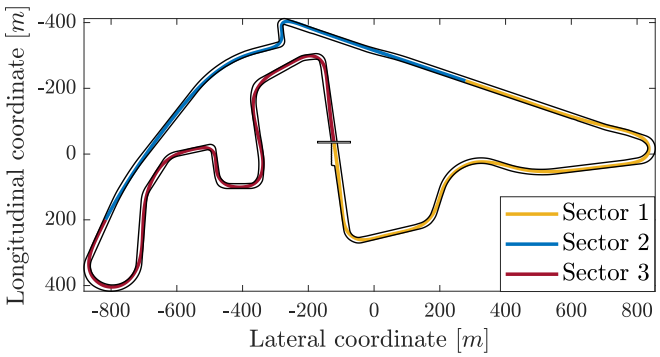

Fig. 6. Yas Marina track sectors used to generate training and validation datasets with different sizes, for the neural network controllers of this paper.

TABLE I

TRAINING AND VALIDATION DATASETS, USING THE TRACK SECTORS IN FIG. 6.

	Training sets			**Validation set**
	Small	Medium	Large	
Track sectors	3	1+3	1+2+3	1+2+3
Lap	1	1	1	2

of the neural controllers in unseen conditions. To further evaluate the capability to learn with limited data, we generate three different training sets of increasing size, as reported in Table I.

B. MS-NN Implementation and Training

Designing highly structured neural networks (e.g., MS-NNs) poses challenges when using general-purpose machine learning frameworks like PyTorch or TensorFlow. Although powerful and flexible, these libraries typically demand substantial expertise to embed domain-specific structures and physical priors, leading to a complex and time-consuming development process.

To address this challenge, we used `nnodely`, a new open-source software tool that streamlines the implementation of MS-NNs, available at `https://github.com/tonegas/nnodely.git`.

`nnodely` provides an intuitive and modular environment tailored for rapid prototyping and evaluation of MS-NN architectures. It lowers the entry barrier for users with classical modeling backgrounds (e.g., engineers, physicists) by streamlining the integration of physical principles and facilitating deployment.

The training of the proposed and benchmark neural models is conducted in `nnodely`, using supervised learning to minimize the Root Mean Square Error (RMSE) between the predicted steering angles δ and the corresponding measured values δ_{meas}, obtained from the experimental data. Training is performed using the Adam algorithm. To prevent overfitting, we employ the early stopping technique: the training process is stopped if the validation loss does not improve after a predefined number of epochs.

The training hyperparameters are: learning rate 10^{-3}, 8000 epochs, batch size 1000, and early stopping with a patience of 1500 epochs.

C. Evaluation Metrics

In this paper, we consider the following three key performance indicators. The Root Mean Square Error (RMSE) is used to quantify the model accuracy, while the Fraction of Variance Unexplained (FVU) quantifies the proportion of variance in the target data that remains unexplained by the model (lower FVU values indicate better performance). Finally, the Akaike Information Criterion (AIC) [21] is a metric for model selection which trades off the model accuracy versus its complexity (number of learnable parameters), with lower AIC values indicating more sample-efficient models.

D. Hyperparameters Tuning

We tune the MS-NN-steer's hyperparameters with a grid-search approach, to minimize the AIC metric (Section IV-C) on the validation dataset. We consider the following variables: the number of future time steps q in the input vectors, the numbers of local models $\{n_y, n_x, n_v\}$ for the lateral accelerations, the longitudinal accelerations and the longitudinal speed, respectively. The following ranges are assumed: $q \in (4, 9, 14)$, $n_y \in (3, 5, 7)$, $n_x \in (3, \dots, 7)$ and $n_v \in (3, \dots, 7)$.

The results are shown in Fig. 7, which displays a heatmap of the AIC values computed for each combination of hyperparameters. Each subplot includes a white dot that marks the local minimum of the AIC within the corresponding parameter configuration. By examining these local minima and identifying the lowest overall value across all configurations, the best model corresponds to the setting with $q = 9$, $n_y = 5$, $n_x = 3$ and $n_v = 3$. This configuration yields the lowest AIC value, and thus the best balance between model complexity and predictive performance.

Note that the MS-NN-steer's input vectors span a time window of $q \cdot T$ seconds, where the sampling time T is 0.05 s in our implementation. Thus, in the best setting, the input vectors span 0.45 s, which is enough to cover most of the time-scale of the lateral vehicle dynamics [14].

Finally, the number of learnable parameters for MS-NN-steer is:

$$\begin{aligned} N_{\text{pars}} &= (n_{p_1} + 1 + n_{p_2} + 1) \cdot n_y + \\ &+ k_{y3,\cdots,6} \cdot n_y + k_{x1,\cdots,5} \cdot n_x + (q+1) \cdot n_x \cdot n_v = \\ &= (4+1+1+1) \cdot 5 + 4 \cdot 5 + 5 \cdot 3 + (9+1) \cdot 3 \cdot 3 = 160 \end{aligned} \tag{7}$$

E. Performance Evaluation

After tuning the numbers of parameters, the focus shifts to evaluating the model's accuracy and its ability to generalize on new data. Table II, Fig. 8 and 9 compare the performance of our MS-NN-steer and the benchmarks, using the large training set of Table I. The proposed MS-NN-steer outperforms all the benchmarks, improving

Manuscript 261 submitted to 2025 IEEE 28th International Conference on Intelligent Transportation Systems (ITSC 2025). Received May 1, 2025.

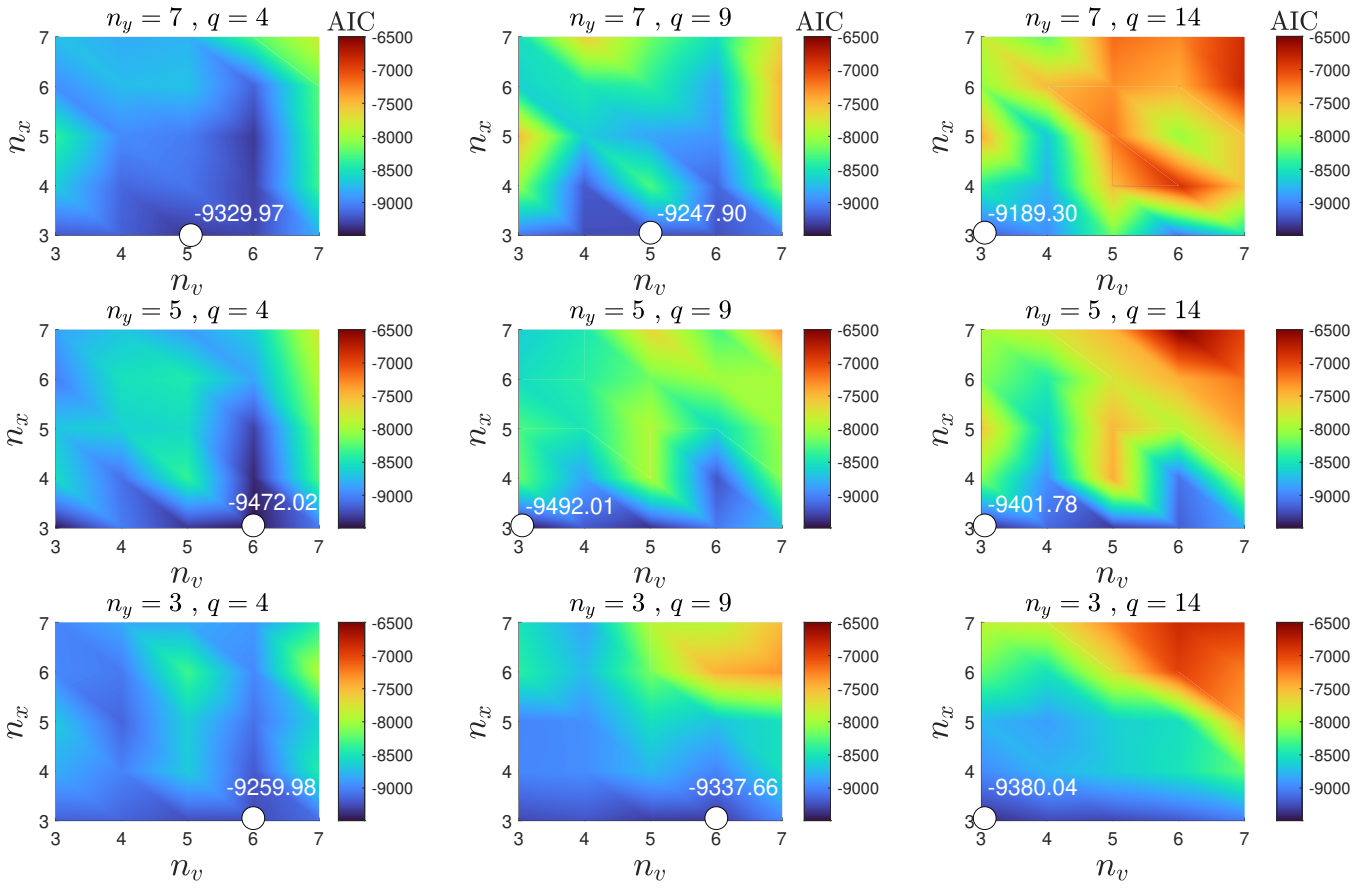

Fig. 7. Grid search to tune the MS-NN-steer's hyperparameters. The figure shows the Akaike Information Criterion (AIC) for each combination of (q, n_y, n_x, n_v) in their ranges. The AIC metric is a trade-off between model complexity, quantified by the number of learnable parameters, and the accuracy on the validation dataset. The lowest value, i.e., the most sample-efficient model, results in $q = 9$, $n_y = 5$, $n_x = 3$ and $n_v = 3$.

TABLE II

RESULTS WITH THE LARGE TRAINING DATASET.

	Metrics	Training Set	Validation Set
G-NN	RMSE [deg]	0.189	0.217
	FVU ($\cdot 10^3$)	6.1	10.0
A2RL-control [17]	RMSE [deg]	0.280	0.259
	FVU ($\cdot 10^3$)	13.0	14.0
MS-NN-base [14]	RMSE [deg]	0.097	0.109
	FVU ($\cdot 10^3$)	2.3	2.2
MS-NN-steer (ours)	RMSE [deg]	**0.079**	**0.103**
	FVU ($\cdot 10^3$)	**1.2**	**1.6**

the RMSE and FVU both on the training and validation sets. In validation, the performance gain over the G-NN, A2RL-control and MS-NN-base are 53%, 60% and 6% in terms of RMSE, respectively, while the FVU improves by 27% compared to the MS-NN-base. Also, MS-NN-steer reduces the peak steering errors by more than 60% compared to the A2RL-control (Fig. 9). Finally, A2RL-control shows high-frequency steer oscillations in many parts of the dataset (Fig. 8(a)), while our MS-NN-steer yields smoother profiles and better robustness to noise on the input signals. Smoother steering profiles favor the vehicle's stability and reduce the risk of dangerous oversteering situations. Overall, our MS-NN-steer can more accurately execute the planned trajectories, with a potentially large impact on the vehicle's overall performance and lap times.

When trained with small datasets (Table III), the benchmark G-NN significantly degrades its performance on the validation set, with an RMSE of 0.327 deg. In contrast, our MS-NN-steer achieves an RMSE of 0.097 deg, thus demonstrating better generalization capabilities even with limited training data. This is a crucial aspect for real-world applications, where collecting data can be expensive and time-consuming, and the environment may change over time.

Thus, the physics-based internal structure of our MS-NN-steer allows it to learn the underlying dynamics of the vehicle more effectively, reducing the risk of overfitting the training data and improving the performance on unseen scenarios.

F. Sensitivity to the Weights' Initialization

This section evaluates the sensitivity of the proposed MS-NN-steer and the benchmark G-NN to the initialization of the learnable parameters during training. To perform this analysis, the initial guesses for the learnable parameters

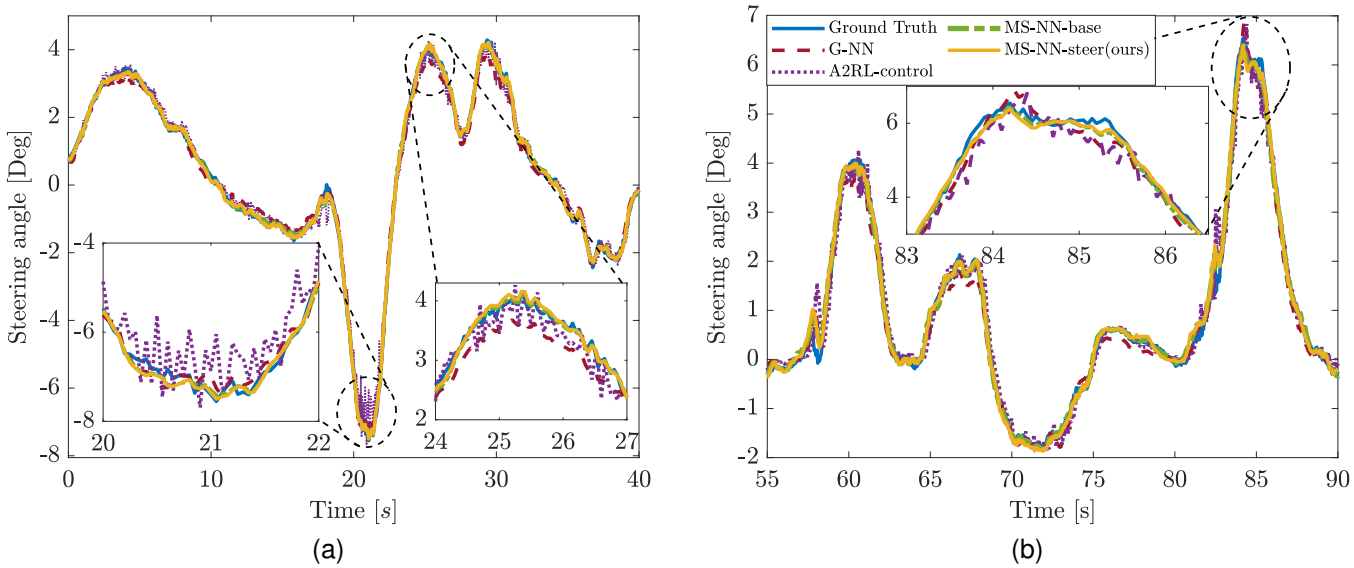

Fig. 8. Steering angle profiles computed by the proposed MS-NN-steer and the benchmarks, when trained with the large training set of Table I. Results on the training set (a) and validation set (b). Note that the plots show around 40 s of the full datasets, for improved visibility.

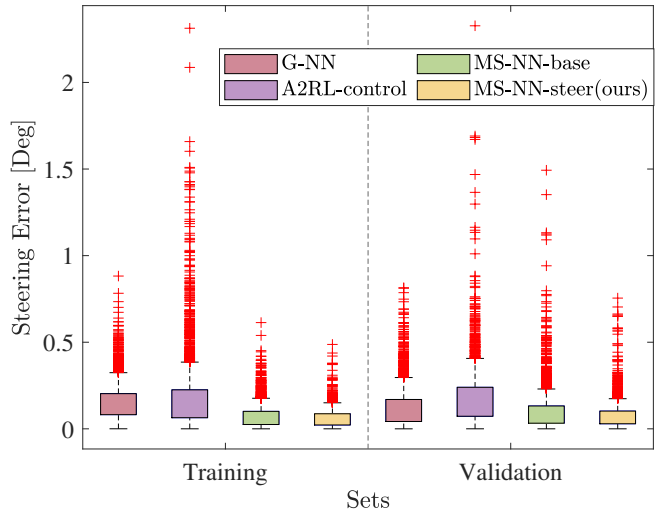

Fig. 9. Boxplot of the steering angle errors, when using the large training dataset of Table I. The proposed MS-NN-steer outperforms the benchmarks both in training and validation, with lower mean, variance, and peak errors.

TABLE III

GENERALIZATION PERFORMANCE USING MEDIUM AND SMALL TRAINING DATASETS. THE VALIDATION SET IS THE SAME IN ALL CASES AND THE SAME AS IN TABLE II.

		Medium Train. Set		Small Train. Set	
	Metrics	Train.	Valid.	Train.	Valid.
G-NN	RMSE [deg]	0.160	0.229	0.132	0.327
	FVU ($\cdot 10^3$)	5.9	15.0	2.5	79.0
MS-NN-base [14]	RMSE [deg]	0.079	0.097	0.069	0.115
	FVU ($\cdot 10^3$)	1.0	2.0	0.6	2.6
MS-NN-steer (ours)	RMSE [deg]	**0.068**	**0.091**	**0.063**	**0.097**
	FVU ($\cdot 10^3$)	**0.8**	**1.8**	**0.6**	**2.0**

of the fully connected layers are randomized by sampling from a Gaussian distribution, with zero mean and a standard deviation of 10^{-4}. For each network, 10 different configurations are tested by varying the random seed used to initialize the weights.

The results are shown in the boxplots of Fig. 10. Our MS-NN-Steer is almost insensitive to the initialization of its weights, as indicated by the very small variance of the RMSE values across different seeds. This is a significant advantage, as it suggests that the model can achieve consistent performance regardless of the initial conditions. In contrast, the G-NN exhibits a very high variance in its RMSE values, indicating that its performance is more sensitive to the choice of initial weights, and thus it needs to be re-trained multiple times to achieve a satisfactory performance.

Additionally, we conduct a related analysis by varying the learning rate in the range from 10^{-3} to 10^{-5}. This investigation revealed a similar trend, confirming that our architecture maintains stable performance across a wide range of training conditions.

Hence, the proposed MS-NN-steer is robust to variations in its initial weights and learning rate, which is a desirable property since it reduces the need to perform extensive hyperparameter tuning and multiple training runs.

V. CONCLUSION

This paper presented a novel model-structured neural network (MS-NN-steer) for the steering control of a full-scale autonomous race car. The proposed architecture is based on the MS-NN framework, which incorporates physical priors and structured local models to improve the interpretability and generalization capabilities of the resulting neural network. We designed MS-NN-steer to capture the effect of the vehicle speed on the steering dynamics and handling diagram, and we showed that this

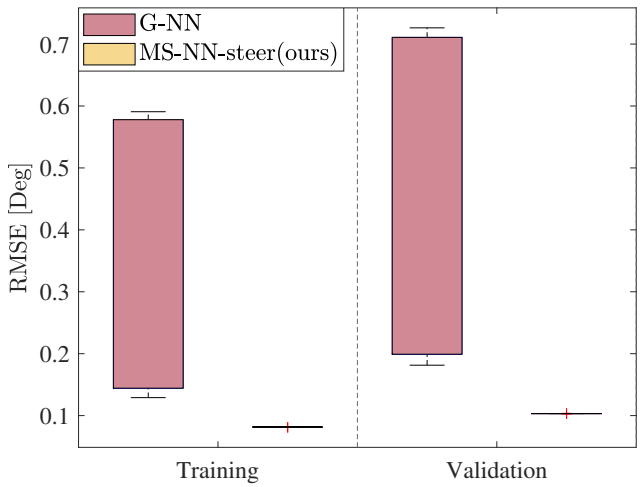

Fig. 10. Boxplot of the RMSE values for the MS-NN-steer and G-NN, when trained with different weights' initialization. Our MS-NN-steer shows a very low variance in the RMSE values, suggesting that it is almost insensitive to the weights' initialization. In contrast, G-NN exhibits a high variance, indicating that its performance is sensitive to the choice of initial weights.

leads to improved performance compared to the previous MS-NN-base architecture.

The proposed MS-NN-steer was trained and validated using real-world telemetry data from the TUM Autonomous Motorsport team, which won the A2RL competition in 2024. Our results demonstrate that the MS-NN-steer outperforms a general-purpose neural network, the A2RL baseline steering controller, and the previous MS-NN-base architecture. When trained with small datasets, our MS-NN-steer can still generalize well to unseen data, achieving a lower RMSE and FVU compared to the benchmarks. Finally, the MS-NN-steer's training is less sensitive to the initialization of the learnable parameters, which speeds up the training process and reduces the need for extensive hyperparameter tuning.

REFERENCES

[1] J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam, "Autonomous vehicles on the edge: A survey on autonomous vehicle racing," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 458–488, 2022.

[2] T. Zhang, Y. Sun, Y. Wang, B. Li, Y. Tian, and F.-Y. Wang, "A survey of vehicle dynamics modeling methods for autonomous racing: Theoretical models, physical/virtual platforms, and perspectives," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 3, pp. 4312–4334, 2024.

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

[4] M. Piccinini, M. Larcher, E. Pagot, D. Piscini, L. Pasquato, and F. Biral, "A predictive neural hierarchical framework for on-line time-optimal motion planning and control of black-box vehicle models," *Vehicle system dynamics*, vol. 61, no. 1, pp. 83–110, 2023.

[5] 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)*. IEEE, 2021, pp. 407–412.

[6] X. Nie, C. Min, Y. Pan, K. Li, and Z. Li, "Deep-neural-network-based modelling of longitudinal-lateral dynamics to predict the vehicle states for autonomous driving," *Sensors*, vol. 22, no. 5, p. 2013, 2022.

[7] N. A. Spielberg, M. Brown, N. R. Kapania, J. C. Kegelman, and J. C. Gerdes, "Neural network vehicle models for high-performance automated driving," *Science Robotics*, vol. 4, no. 28, p. eaaw1975, 2019.

[8] J. Liu, Y. Cui, J. Duan, Z. Jiang, Z. Pan, K. Xu, and H. Li, "Reinforcement learning-based high-speed path following control for autonomous vehicles," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 6, pp. 7603–7615, 2024.

[9] T. Weiss and M. Behl, "Deepracing: A framework for autonomous racing," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020, pp. 1163–1168.

[10] J. Chrosniak, J. Ning, and M. Behl, "Deep dynamics: Vehicle dynamics modeling with a physics-constrained neural network for autonomous racing," *IEEE Robotics and Automation Letters*, 2024.

[11] Z. Zhou, Y. Wang, G. Zhou, X. Liu, M. Wu, and K. Dai, "Vehicle lateral dynamics-inspired hybrid model using neural network for parameter identification and error characterization," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 11, pp. 16 173–16 186, 2024.

[12] M. Da Lio, D. Bortoluzzi, and G. P. Rosati Papini, "Modelling longitudinal vehicle dynamics with neural networks," *Vehicle System Dynamics*, vol. 58, no. 11, pp. 1675–1693, 2020.

[13] M. Da Lio, R. Donà, G. P. R. Papini, F. Biral, and H. Svensson, "A mental simulation approach for learning neural-network predictive control (in self-driving cars)," *Ieee Access*, vol. 8, pp. 192 041–192 064, 2020.

[14] 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.

[15] M. Piccinini, M. Zumerle, J. Betz, and G. P. R. Papini, "A road friction-aware anti-lock braking system based on model-structured neural networks," *IEEE Open Journal of Intelligent Transportation Systems*, pp. 1–1, 2025.

[16] M. D. 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 Transactions on Intelligent Transportation Systems*, vol. 24, no. 12, pp. 13 670–13 684, 2023.

[17] J. Betz, T. Betz, F. Fent, M. Geisslinger, A. Heilmeier, L. Hermansdorfer, T. Herrmann, S. Huch, P. Karle, M. Lienkamp *et al.*, "Tum autonomous motorsport: An autonomous racing software for the indy autonomous challenge," *Journal of Field Robotics*, vol. 40, no. 4, pp. 783–809, 2023.

[18] O. Nelles, A. Fink, and R. Isermann, "Local linear model trees (lolimot) toolbox for nonlinear system identification," *IFAC Proceedings Volumes*, vol. 33, no. 15, pp. 845–850, 2000.

[19] O. Nelles and O. Nelles, *Nonlinear dynamic system identification*. Springer, 2020.

[20] M. Guiggiani, *The Science of Vehicle Dynamics: Handling, Braking, and Ride of Road and Race Cars*. Springer, 2018.

[21] H. Bozdogan, "Model selection and akaike's information criterion (aic): The general theory and its analytical extensions," *Psychometrika*, vol. 52, no. 3, pp. 345–370, 1987.