UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

Department of Industrial Engineering

Ph.D. Dissertation

Doctoral Program in Industrial Engineering

XXXVII Cycle

Ph.D. Coordinator: Prof. Eng. Michele Grassi

Towards Autonomous Driving: Integrating Data-Driven Approaches into Model-Based Framework for AI-Enhanced Vehicle Dynamics

Ph.D. Candidate

Raffaele Marotta

(DR995584)

Supervisors

Prof. Eng. Salvatore Strano

(University of Naples "Federico II")

Prof. Eng. Mario Terzo

(University of Naples "Federico II")

Reviewers

Prof. Eng. Umberto Montanaro

(University of Surrey)

Prof. Eng. Viktor Skrickij

(Vilnius Gediminas Technical University)

"Twenty years from now you will be more disappointed by the things that you didn't do than by the ones you did do. So, throw off the bowlines. Sail away from the safe harbor. Catch the trade winds in your sails. Explore. Dream. Discover."

(Mark Twain)

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my supervisors, Prof. Salvatore Strano and Prof. Mario Terzo, for guiding me throughout my PhD journey. Their unwavering belief in me, even in the most challenging moments, gave me the strength to persevere. Without their support and confidence, this doctoral path would not have reached its conclusion.

A heartfelt thank you to my colleague, friend, elder brother, and technical guide, Ciro Tordela, for being by my side throughout this journey.

I also extend my sincere thanks to Prof. Valentin Ivanov, who guided me with enthusiasm and expertise during my time at TU Ilmenau in Germany.

To Sebastiaan van Aalst, Kylian Praet, and Miguel Dhaens, thank you for welcoming me at Tenneco, Belgium, and giving me the first taste of the industrial world.

I would like to acknowledge my colleagues in Naples, with whom I always wished I could have worked more closely: Mario Spirto, Enrico Fornaro, Pierangelo Malfi, Armando Nicolella, Francesco Melluso, and Chiara Cosenza.

Last but certainly not least, I am deeply grateful to my family, friends, and to all those who believed in me.

Raffaele Marotta

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

The advancement of autonomous vehicle technology has the potential to revolutionize transportation, enhancing safety, efficiency, and convenience. At the core of this transformation lies the integration of model-based estimation and control techniques with artificial intelligence (AI). This integration is pivotal in addressing the complex challenges associated with vehicle dynamics and ensuring reliable and robust performance of autonomous vehicles. This thesis addresses this gap by developing novel methodologies that integrate these approaches to overcome the limitations of existing systems, with a particular emphasis on its significance for the future of autonomous driving.

Vehicle dynamics encompass the behavior of a vehicle in motion, governed by the interaction of various forces and moments acting on the vehicle. Accurate estimation and control of these dynamics are crucial for several reasons:

- *Safety*: Autonomous vehicles must operate safely in diverse and unpredictable environments. Precise control of vehicle dynamics is essential to maintain stability, avoid collisions, and respond effectively to emergencies.
- *Efficiency*: Optimizing vehicle dynamics can lead to more efficient driving patterns, reducing energy consumption and emissions. This is particularly important in the context of electric and hybrid vehicles, where range and energy efficiency are critical concerns.
- *Comfort*: Smooth and predictable vehicle behavior enhances passenger comfort. Proper control of dynamics ensures a pleasant

ride experience by minimizing sudden accelerations, decelerations, and lateral forces.

- *Reliability*: Robust estimation and control systems improve the reliability of autonomous vehicles. By accurately predicting and responding to dynamic changes, these systems ensure consistent performance under varying conditions.

Model-based estimation and control rely on mathematical models to describe the vehicle's dynamic behavior. These models can be derived from fundamental physical principles. Key advantages of model-based techniques include:

- *Predictive capability*: Models provide a predictive framework that enables proactive control strategies, anticipating changes in vehicle behavior and adjusting controls accordingly.
- *Systematic design*: Model-based methods allow for systematic design and tuning of control systems, leveraging well-established theories and tools from control engineering.
- *Transparency and interpretability*: Models offer a transparent representation of vehicle dynamics, facilitating analysis, debugging, and validation of control algorithms.

Common model-based techniques used in vehicle dynamics include Kalman filters for state estimation, Linear Quadratic Regulators (LQR) for optimal control, and Model Predictive Control (MPC) for handling constraints and optimizing performance over a prediction horizon.

AI, particularly machine learning (ML), has emerged as a powerful tool for enhancing vehicle dynamics estimation and control. AI techniques can complement model-based approaches in several ways:

- *Data-driven insights*: ML algorithms can extract patterns and insights from vast amounts of data, improving the accuracy of dynamic models and identifying previously unknown correlations.
- *Adaptive control*: AI enables adaptive control strategies that can learn and adjust to changing conditions, enhancing the robustness of control systems in real-world scenarios.
- *Complex decision-making*: AI can handle complex decision-making processes, such as path planning and obstacle avoidance, by considering a multitude of factors and optimizing actions in real-time.

The primary objective of this thesis is to bridge the research gap by proposing hybrid methodologies that leverage the strengths of both model-based and AI-driven techniques. Specifically, the work focuses on developing advanced solutions for critical tasks, including:

1. Predicting tire-road interaction forces using a combination of physical models and neural networks.
2. Estimating the sideslip angle through dynamic neural networks.
3. Measuring and controlling unsprung mass displacements using virtual sensors integrated with AI methods.
4. Addressing roll and pitch dynamics to improve vehicle stability on uneven terrains through a fusion of proportional control and reinforcement learning.

The relevance of advanced vehicle dynamics estimation and control is underscored by the Society of Automotive Engineers (SAE) classification of driving automation levels. The SAE defines six levels of driving automation, ranging from Level 0 (no automation) to Level 5 (full automation):

- *Level 0*: No Automation – The driver handles every aspect of driving.
- *Level 1*: Driver Assistance – Driver Assistance: The vehicle provides assistance with steering and braking, but the driver still has total control.
- *Level 2*: Partial Automation – The vehicle can steer and accelerate/decelerate, but the driver has to stay focused and aware of their surroundings.
- *Level 3*: Conditional Automation – In certain situations, the vehicle can drive itself; but the driver has to be prepared to take over when necessary.
- *Level 4*: High Automation – In certain contexts, the vehicle can drive itself and handle all driving duties without the driver's help, but in other circumstances, the driver may need to take control of the vehicle.
- *Level 5*: Full Automation – Without any input from the driver, the vehicle is capable of handling every driving duty in every situation.

As vehicles progress from Level 0 to Level 5, the demands on dynamic estimation and control systems increase significantly. For instance, at Level 4 and Level 5, vehicles must be capable of handling a wide range of dynamic scenarios autonomously, without human intervention. This requires highly reliable and precise control systems that can adapt to varying conditions, making the integration of model-based techniques and AI critical. By focusing on these high-stakes scenarios, this thesis lays the groundwork for systems capable of maintaining stability, efficiency, and comfort across a wide range of operating conditions.

1 | OVERVIEW OF ESTIMATION TECHNIQUES

Estimation is a crucial discipline in numerous fields of engineering, playing a vital role in control systems, signal processing, and robotics, among others. Two primary approaches dominate this field: model-based methods and data-driven techniques.

The state of a dynamic system is not typically measurable directly. Even when it is feasible, practical considerations, such as cost or complexity, often lead to avoiding direct measurement. Instead, measurements are taken that are in some way related to the state. These measurements are linear or nonlinear functions of elements within the state [1].

This chapter provides a detailed exploration of these two paradigms, focusing on the Kalman filter for model-based estimation and neural networks for data-driven estimation.

1.1 | Model-based estimation: Kalman filter

Model-based estimation relies on mathematical models that describe the system dynamics. One of the most prominent and widely used techniques in this category is the Kalman filter [2].

A Kalman filter is an algorithm that uses sensor measurements to infer the hidden internal state of a dynamic system.

Equation (1.1) shows a general dynamic model:

$$\begin{aligned} x_k &= f_k(x_{k-1}, u_{k-1}, w_{k-1}) \\ z_k &= h_k(x_k, u_k, v_k) \end{aligned} \tag{1.1}$$

Where:

- x_k is the system state vector at time index k.
- u_k is the measurable (deterministic) input to the system at time index k.
- w_k is the disturbance or process noise, an unmeasurable input that affects the state.
- z_k is the sensor measurement, related to x_k.
- v_k is sensor noise that corrupts the measurement.

The first equation is the "state equation" or "process equation" that describes how the state evolves over time. The second equation is the "output equation" or "measurement equation" which describes how the measured output relates to the state.

The goal is to estimate the values of the state vector x_k given all past measurements z_k, z_1,…, z_k.

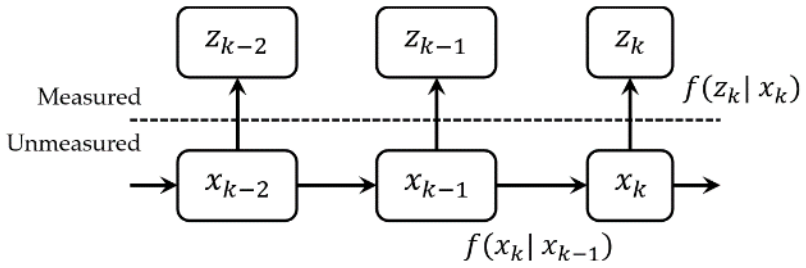


Figure 1.1 | Relationship between state and measurement in a dynamic system.

Kalman filters repeatedly execute two steps:

1. Infer the current state value from all of the historical data that is accessible.
2. By revising the prediction using all of the data that is currently available, estimate the value of the current state.

1.1.1 | Linear Kalman filter

Linear systems have the property that all variables of the system remain Gaussian if the stochastic inputs are Gaussian.

The linear Kalman filter assumes that the system being modeled can be represented in the state-space form:

$$\begin{aligned} x_k &= Ax_{k-1} + Bu_{k-1} + w_{k-1} \\ z_k &= Cx_k + Du_k + v_k \end{aligned} \tag{1.2}$$

Where:

- x_k is the system state vector at time index k.
- u_k is the measurable (deterministic) input to the system at time index k.
- w_k is the disturbance or process noise, an unmeasurable input that affects the state.
- z_k is the sensor measurement, related to x_k.
- v_k is sensor noise that corrupts the measurement.
- A, B, C, and D are matrices, specific to the observed system, that describe its dynamic behavior.

The two recursive in the linear Kalman filter steps are detailed:

- *Step 1a*: State estimate time update.

$$\hat{x}_k^- = A_{k-1}\hat{x}_{k-1}^+ + B_{k-1}u_{k-1} \tag{1.3}$$

The objective is to predict the current state utilizing the most recent state estimate and the system model. $\hat{x}_k^-$ is called a priori estimate.

- *Step 1b*: Error covariance time update.

$$\begin{aligned} \tilde{x}_k^- = x_k - \hat{x}_k^- &= A_{k-1}x_{k-1} + B_{k-1}u_{k-1} + w_{k-1} - A_{k-1}\hat{x}_{k-1}^+ \\ &- B_{k-1}u_{k-1} = A_{k-1}\tilde{x}_{k-1}^+ + w_{k-1} \end{aligned} \tag{1.4}$$

$$\Sigma_{\tilde{x},k}^- = A_{k-1}\Sigma_{\tilde{x},k-1}^+ A_{k-1}^T + \Sigma_{\tilde{w}} \tag{1.5}$$

The approach involves utilizing the most recent covariance estimate and propagating it forward in time.

For stable systems, the term $A_{k-1}\Sigma^{+}_{\tilde{x},k-1}A^{T}_{k-1}$ is contractive, indicating that the covariance diminishes. In the absence of input, the state of stable systems converges toward zero, or toward a known trajectory if $u_k \neq 0$. This contractive property implies that the accuracy of the state estimate increases over time.

Conversely, the process noise covariance $\Sigma_{\tilde{w}}$ contributes to the overall covariance. Uncertainty of the inputs introduces additional uncertainty into the state estimate by deviating the system's trajectory from the predicted path based solely on u_k.

- *Step 1c*: Predict system output.

$$\hat{z}_k = C\hat{x}^{-}_k + Du_k \tag{1.6}$$

The optimal approach is to estimate the output by utilizing the output equation of the system model and the most accurate estimate of the system state at the present time.

- *Step 2a*: Kalman gain matrix.

$$L_k = \frac{\Sigma^{-}_{\tilde{x},k}C^{T}_k}{C_k\Sigma^{-}_{\tilde{x},k}C^{T}_k + \Sigma_{\tilde{v}}} \tag{1.7}$$

The primary purpose of calculating covariance matrices is to facilitate the update of L_k. The gain L_k is time-varying, adapting to provide the optimal update to the state estimate based on current conditions.

The first component of L_k, $\Sigma^{-}_{\tilde{x},k}C^{T}_k$, indicates the relative need for correction to $\hat{x}_k$ and the degree to which individual states within $\hat{x}_k$ are coupled to the measurements.

The matrix $\Sigma^{-}_{\tilde{x},k}$ informs us about the state uncertainty at the present time. A large entry in $\Sigma^{-}_{\tilde{x},k}$ signifies that the corresponding state is highly

uncertain and would benefit from a substantial update. Conversely, a small entry in $\Sigma_{\tilde{x},k}^{-}$ indicates that the corresponding state is already well-known and requires a smaller update.

The term C_k^T provides the coupling between the state and the output. Entries that are zero indicate that a particular state has no direct influence on a specific output, and therefore an output prediction error should not directly update that state. Conversely, entries that are large indicate that a particular state is highly coupled to an output and thus contributes significantly to any measured output prediction error, warranting a substantial update to that state.

The matrix $\Sigma_{\tilde{v}}$ indicates the reliability of the measurement. If $\Sigma_{\tilde{v}}$ is large, we prefer small, gradual updates. Conversely, if $\Sigma_{\tilde{v}}$ is small, we opt for larger updates.

- *Step 2b*: State estimate measurement update.

The fifth step involves computing the a posteriori state estimate by updating the a priori estimate using the estimator gain and the output prediction error $z_k - \hat{z}_k$:

$$\hat{x}_k^+ = \hat{x}_k^- + L_k(z_k - \hat{z}_k) \tag{1.8}$$

The variable $\hat{z}_k$ represents the expected measurement based on the current state estimate. Therefore, $z_k - \hat{z}_k$ represents the unexpected or new information in the measurement. The error may arise from a flawed system model, or sensor noise. This new information should be used to update the state estimate, but it must be weighted according to its informational value. The term L_k is the blending factor.

- *Step 2c*: Error covariance measurement update.

$$\Sigma_{\tilde{x},k}^{+} = (I - L_k C_k)\Sigma_{\tilde{x},k}^{-} \tag{1.9}$$

The measurement update has reduced the uncertainty in our state estimate.

If a measurement is missed for any reason, then step 2 for that iteration is skipped. This means $L_k = 0$, $\hat{x}_k^+ = \hat{x}_k^-$, and $\Sigma_{\tilde{x},k}^+ = \Sigma_{\tilde{x},k}^-$.

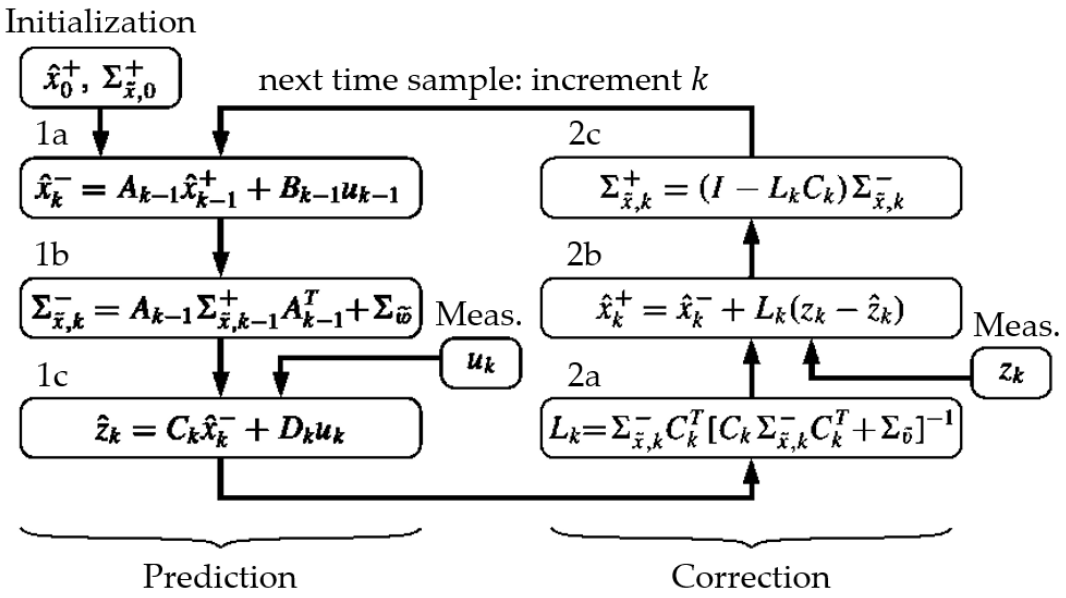


Figure 1.2 | Schematic representation of the recursive steps of a linear Kalman filter.

1.1.2 | Extended Kalman filter

The focus now shifts to the nonlinear case, characterized by system dynamics (1.1).

The Extended Kalman filter (EKF) is a nonlinear generalization of the Kalman filter. The EKF linearizes the system equations around the current operating parameter using Taylor series expansions in order to compute covariance estimations.

The two recursive in the EKF are detailed:

- *Step 1a*: State estimate time update.

$$\hat{x}_k^- \approx f_{k-1}(\hat{x}_{k-1}^+, u_{k-1}, w_{k-1}) \tag{1.10}$$

The expected value of the state is approximated by propagating $\hat{x}_{k-1}^+$ and through the state equation.

- *Step 1b*: Error covariance time update.

$$\tilde{x}_k^- = x_k - \hat{x}_k^- = f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1}) - f_{k-1}(\hat{x}_{k-1}^+, u_{k-1}, w_{k-1}) \tag{1.11}$$

$$\Sigma_{\tilde{x}_k}^- = A_{k-1}\Sigma_{\tilde{x},k-1}^+ A_{k-1}^T + \Sigma_{\tilde{w}} \tag{1.12}$$

The first term is expanded as a Taylor series around the prior operating point which is the set of values $\{\hat{x}_{k-1}^+, u_{k-1}, w_{k-1}\}$.

$$x_k \approx f_{k-1}(\hat{x}_{k-1}^+, u_{k-1}, w_{k-1}) + \frac{df_{k-1}(\hat{x}_{k-1}^+, u_{k-1}, w_{k-1})}{dx_{k-1}}(x_{k-1} + -\hat{x}_{k-1}^+) + \frac{df_{k-1}(\hat{x}_{k-1}^+, u_{k-1}, w_{k-1})}{dw_{k-1}}(w_{k-1} - \overline{w}_{k-1}) \tag{1.13}$$

This gives:

$$\tilde{x}_k^- \approx \hat{A}_{k-1}\tilde{x}_{k-1}^+ + \hat{B}_{k-1}\tilde{w}_{k-1} \tag{1.14}$$

Substituting this to find the predicted covariance:

$$\Sigma_{\tilde{x},k}^- \approx \hat{A}_{k-1}\Sigma_{\tilde{x},k-1}^+ \hat{A}_{k-1}^T + \hat{B}_{k-1}\Sigma_{\tilde{w}}\hat{B}_{k-1}^T \tag{1.15}$$

- *Step 1c*: Predict system output.

The system output is estimated as:

$$\hat{z}_k = h_k(\hat{x}_k^-, u_k, v_k) \tag{1.16}$$

This approach assumes that propagating $\hat{x}_k^+$ provides the best approximation for estimating the output.

- *Step 2a*: Kalman gain matrix.

$$L_k = \frac{\Sigma_{\tilde{x},k}^- \hat{C}_k^T}{\hat{C}_k \Sigma_{\tilde{x},k}^- \hat{C}_k^T + \hat{D}_k \Sigma_{\tilde{v}} \hat{D}_k^T}$$
$$\hat{C}_k = \frac{dh_k(\hat{x}_k^-, u_k, v_k)}{dx_k} \tag{1.17}$$
$$\hat{D}_k = \frac{dh_k(\hat{x}_k^-, u_k, v_k)}{dv_k}$$

- *Step 2b*: State estimate measurement update.

The fifth step involves computing the a posteriori state estimate by updating the a priori estimate using the estimator gain and the output prediction error $z_k - \hat{z}_k$:

$$\hat{x}_k^+ = \hat{x}_k^- + L_k(z_k - \hat{z}_k) \tag{1.18}$$

- *Step 2c*: Error covariance measurement update.

Finally, the updated covariance is computed as:

$$\Sigma_{\tilde{x},k}^+ = (I - L_k\hat{C}_k)\Sigma_{\tilde{x},k}^- \tag{1.19}$$

The EKF has some disadvantages:

- EKF relies on the linearization of nonlinear functions around the current estimate, which can lead to inaccuracies. If the state transition and observation models are highly nonlinear, the linear approximation might not be accurate, resulting in poor filter performance. Linearization errors can propagate through the iterations of the filter, leading to accumulated inaccuracies over time. This is particularly problematic in long-term applications where small errors can grow and affect the overall estimation.
- Linearization requires the computation of Jacobian matrices, which can be computationally intensive, especially for high-dimensional systems. This can make EKF less suitable for real-time applications with limited computational resources.

1.1.3 | Unscented Kalman filter

Instead of linearizing the functions, the Unscented Kalman filter (UKF), also known as Central Difference Kalman Filter (CDKF), selects a set of carefully chosen sample points, called sigma points, around the mean. These points are then propagated through the nonlinear functions. The resulting mean and covariance are computed based on the transformed

sigma points, providing a more accurate representation of the true nonlinear transformation. This approach captures the true mean and covariance to the second order, at least, for any nonlinearity, offering a more precise estimate compared to EKF. Although generating and propagating sigma points involves some computational effort, it is often more efficient and straightforward than calculating Jacobians, especially for systems where deriving Jacobians is complex or infeasible.

The filter uses the parameters shown in Table 1.1.

γ	$\alpha_0^{(m)}$	$\alpha_k^{(m)}$	$\alpha_0^{(c)}$	$\alpha_k^{(c)}$
h	$\frac{h^2 - L}{h^2}$	$\frac{1}{2h^2}$	$\frac{h^2 - L}{h^2}$	$\frac{1}{2h^2}$

Table 1.1 | Parameter of the UKF

For Gaussian random variables:

$$h = \sqrt{3} \tag{1.20}$$

L represents the augmented state vector's size.

The two recursive in the UKF are detailed:

- *Step 1a*: State estimate time update.

First, form the augmented a posteriori state estimate vector for the previous time interval:

$$\hat{x}_{k+1}^{a,+} = [(\hat{x}_{k-1}^{+})^T, \bar{w}, \bar{v}] \tag{1.21}$$

Where w and v are the uncertainties of the process and of the measures (output), and the augmented a posteriori covariance estimate:

$$\Sigma_{\tilde{x},k-1}^{a,+} = diag(\Sigma_{\tilde{x},k-1}^{+}, \Sigma_{\tilde{w}}, \Sigma_{\tilde{v}}) \tag{1.22}$$

These factors are used to generate the $p + 1$ augmented sigma points:

$$\chi_{k-1}^{a,+} = \left\{\hat{x}_{k-1}^{a,+}, \hat{x}_{k-1}^{a,+} + \gamma\sqrt{\Sigma_{\tilde{x},k-1}^{a,+}}, \hat{x}_{k-1}^{a,+} - \gamma\sqrt{\Sigma_{\tilde{x},k-1}^{a,+}}\right\} \Sigma_{\tilde{x},k-1}^{a,+} \tag{1.23}$$

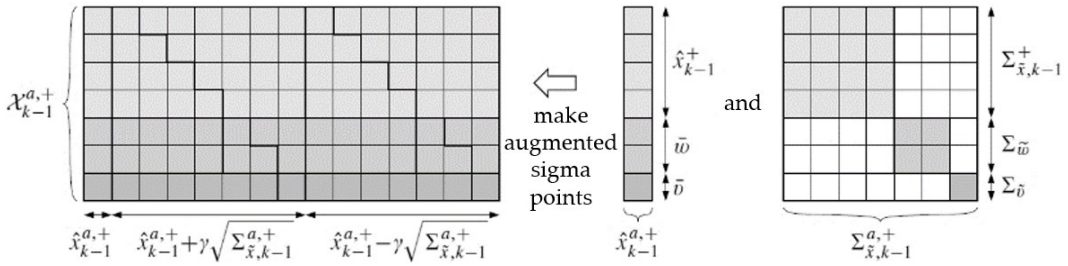


Figure 1.3 | Schematic representation of the generation of augmented sigma points.

These sigma-points are divided into state portion $\chi_{k-1}^{x,+}$, process-noise portion $\chi_{k-1}^{w,+}$ and sensor-noise portion $\chi_{k-1}^{v,+}$.

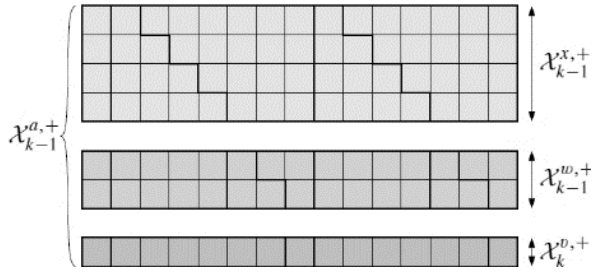


Figure 1.4 | Schematic representation of the division among state portion $\boldsymbol{\chi_{k-1}^{x,+}}$, process-noise portion $\boldsymbol{\chi_{k-1}^{w,+}}$ and sensor-noise portion $\boldsymbol{\chi_{k-1}^{v,+}}$.

The state equation is evaluated using all pairs of $\chi_{k-1,i}^{x,+}$ and $\chi_{k-1,i}^{w,+}$ (where subscript i denotes that the $i-$th vector is being extracted from the original set), yielding the a priori sigma points $\chi_{k,i}^{x,-}$.

$$\chi_{k,i}^{x,-} = f(\chi_{k-1,i}^{x,+}, u_{k-1}, \chi_{k-1,i}^{w,+}) \tag{1.24}$$

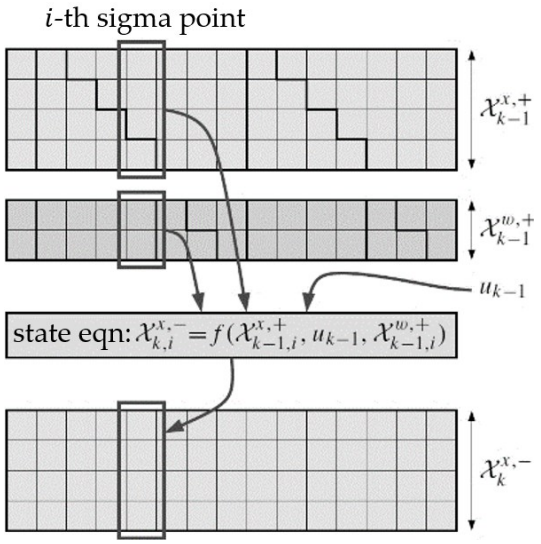


Figure 1.5 | A priori sigma points $\boldsymbol{\mathcal{X}_{k,i}^{x,-}}$ evaluation.

Finally, the a priori state estimate is computed as:

$$\hat{x}_{k,i}^{x,-} \approx \sum_{i=0}^{p} \alpha_i^{(m)} \mathcal{X}_{k,i}^{x,-} \tag{1.25}$$

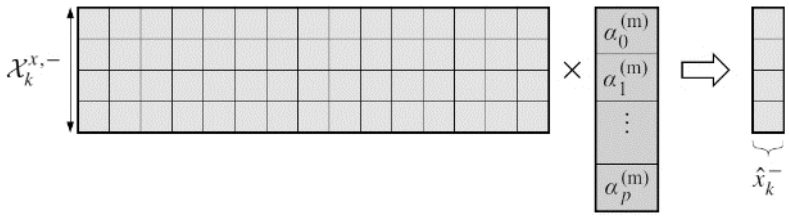


Figure 1.6 | A priori state estimation $\hat{\boldsymbol{x}}_{\boldsymbol{k,i}}^{\boldsymbol{x,-}}$ evaluation.

- *Step 1b*: Error covariance time update.

Using the a priori sigma points from step 1a, the a priori covariance estimate is computed as:

$$\begin{aligned} \Sigma_{\tilde{x},k}^{-} &= \sum_{i=0}^{p} \alpha_i^{(c)} (\mathcal{X}_{k,i}^{x,-} - \hat{x}_k^-)(\mathcal{X}_{k,i}^{x,-} - \hat{x}_k^-)^T \\ &= \alpha_0^{(c)} (\mathcal{X}_{k,0}^{x,-} - \hat{x}_k^-)(\mathcal{X}_{k,0}^{x,-} - \hat{x}_k^-)^T + \sum_{i=1}^{p} \left[\sqrt{\alpha_i^{(c)}} (\mathcal{X}_{k,i}^{x,-} + \right. \\ &\quad \left. - \hat{x}_k^-) \right] \left[\sqrt{\alpha_i^{(c)}} (\mathcal{X}_{k,i}^{x,-} - \hat{x}_k^-)^T \right] \end{aligned} \tag{1.26}$$

$$= \alpha_0^{(c)} \left(\chi_{k,0}^{x,-} - \hat{x}_k^-\right)\left(\chi_{k,0}^{x,-} - \hat{x}_k^-\right)^T + \chi_\alpha \chi_\alpha^T$$

- *Step 1c*: Predict system output.

The output z_k is estimated by evaluating the model output equation using the sigma points describing the state and sensor noise.

First, the points are evaluated:

$$\mathrm{Z}_{k,i} = h(\chi_{k,i}^{x,-}, u_k, \chi_{k-1,i}^{v,+}) \tag{1.27}$$

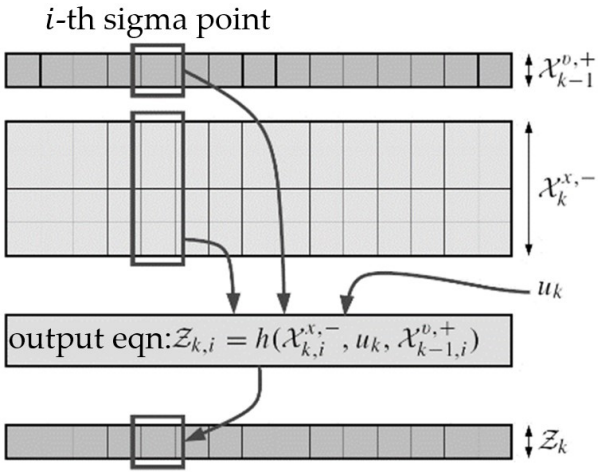


Figure 1.7 | Sigma points $\boldsymbol{Z_{k,i}}$ evaluation.

The output estimate is then:

$$\hat{z}_k = \sum_{i=0}^{p} \alpha_i^{(m)} \mathrm{Z}_{k,i} \tag{1.28}$$

- *Step 2a*: Kalman gain matrix.

To compute the estimator gain matrix, the required covariance matrices are computed:

$$\Sigma_{\tilde{z},k} = \sum_{i=0}^{p} \alpha_i^{(c)} \left(\mathrm{Z}_{k,i} - \hat{z}_k\right)\left(\mathrm{Z}_{k,i} - \hat{z}_k\right)^T \tag{1.29}$$

$$\Sigma_{\tilde{x}\tilde{z},k}^- = \sum_{i=0}^{p} \alpha_i^{(c)} \left(\chi_{k,i}^{x,-} - \hat{x}_k^-\right)\left(\mathrm{Z}_{k,i} - \hat{z}_k\right)^T \tag{1.30}$$

Then, L_k is computed:

$$L_k = \Sigma_{\tilde{x}\tilde{z},k}^{-}\Sigma_{\tilde{z},k}^{-1} \tag{1.31}$$

- *Step 2b*: State estimate measurement update.

Updating the a priori estimate to generate the a posteriori state estimate is the fifth step:

$$\hat{x}_k^{+} = \hat{x}_k^{-} + L_k(z_k - \hat{z}_k) \tag{1.32}$$

- *Step 2c*: Error covariance measurement update.

$$\Sigma_{\tilde{x},k}^{+} = \Sigma_{\tilde{x},k}^{-} - L_k\Sigma_{\tilde{z},k}L_k^T \tag{1.33}$$

1.1.4 | Strengths and Weaknesses

The Kalman filter is a widely used technique for state estimation due to its mathematical rigor and ability to provide optimal estimates under Gaussian noise assumptions. Its predictive nature makes it well-suited for vehicle dynamics applications, such as estimating sideslip angle or tire-road forces. However, it struggles with highly nonlinear dynamics and non-Gaussian noise, which are common in real-world scenarios. Variants like the EKF and UKF address these issues to some extent, but their reliance on accurate system models and computational intensity can limit their practical applicability. While Kalman filters provide transparency and interpretability, their dependence on accurate models is a significant limitation. In scenarios where model parameters are uncertain or external disturbances are prevalent, purely model-based techniques may fail to deliver reliable estimates, necessitating hybrid approaches that incorporate data-driven elements.

1.2 | Data-driven estimation

Data-driven estimation techniques leverage large volumes of data to make predictions or estimations about unknown values or future events. These techniques are essential across various fields such as finance, healthcare, marketing, and engineering.

ML is a subset that involves the development of algorithms capable of learning patterns from data and making decisions or predictions based on that data. The central idea is to enable computers to learn from experience. ML can be divided into two categories: supervised learning and unsupervised learning.

Supervised learning involves training a model on a labeled dataset, meaning that each training example includes the input data and the corresponding correct output. The goal is for the model to learn the mapping from inputs to outputs so that it can make accurate predictions on new, unseen data [3] [4]. Key techniques include:

- *Linear regression*: Used for predicting a continuous outcome. It models the relationship between input features and the target variable as a linear function [5].
- *Logistic regression*: Used for classification problems. It predicts the probability of a binary outcome and models the relationship between input features and a binary target variable using a logistic function [6].
- *Decision trees*: Tree-like models that make decisions based on a series of feature splits. They are interpretable but prone to overfitting [7].
- *Random forests*: An ensemble method that uses multiple decision trees to improve prediction accuracy and control overfitting [8].

- *Support vector machines (SVM)***:** Used in regression and classification problems. SVMs locate the hyperplane in a high-dimensional space that best divides data points of various classes. [9].
- *Neural Networks (NNs)***:** Composed of layers of interconnected nodes or neurons. They can model complex relationships in the data and are particularly powerful for image and speech recognition tasks [10].

Training a model using unlabelled data entails unsupervised learning. The goal is to find hidden patterns or intrinsic structures in the input data [11] [12]. Key techniques include:

- *Clustering*: Groups similar data points together. Common algorithms include K-Means, Hierarchical Clustering, and DBSCAN [13].
- *Anomaly detection***:** Finds outliers, or odd data points, that don't match the data's overall trend. Techniques include Isolation Forest and One-Class SVM [14].

In the studies presented in this thesis, extensive use was made of neural networks.

1.2.1 | Neural networks

NNs are computational models inspired by the human brain's structure and function. They are designed to recognize patterns, classify data, and make predictions. At their core, NNs consist of layers of interconnected nodes (neurons), where each node represents a mathematical function.

1.2.1.1 | Data collection

To develop a NN, it is necessary to have three sets of data:

- *Training dataset*: It is a set of samples that is utilized throughout the learning process to fit the model parameters.

- *Validation dataset*: It is a set of samples that is used to fine-tune the hyperparameters.
- *Test dataset*: It is independent of the training and validation dataset, yet it has the same probability distribution as the training dataset. If an NN that fits the training and validation dataset likewise fits the test dataset well, there has been minimal overfitting.

Each of these datasets is characterized by a collection of input-output pairs.

1.2.1.2 | Basic structure of a NN

A typical NN consists of three types of layers:

1. *Input layer*: Receives the input data.
2. *Hidden layers*: Middle layers that handle the inputs from the input layer. There can be one or more hidden layers.
3. *Output layer*: Produces the final output of the network.

Each neuron in a layer receives inputs, processes them using weights and biases, and applies an activation function to produce an output.

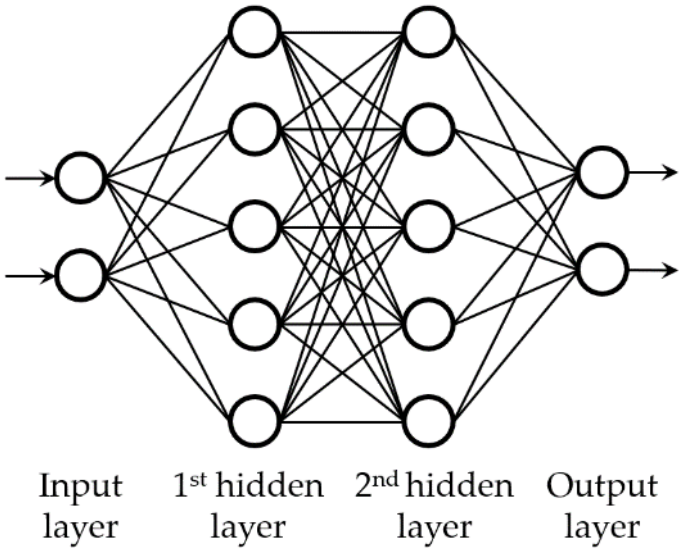


Figure 1.8 | Basic structure of a NN.

1.2.1.3 | Mathematical representation

A neural network can be represented mathematically:

1. *Inputs and weights*: Each neuron receives multiple inputs x_i and each input has an associated weight w_i.
2. *Weighted sum*: The neuron computes a weighted sum of its inputs:

$$z = \sum_{i=1}^{n} w_i x_i + b \tag{1.34}$$

Where b is the bias term.

3. *Activation function*: The weighted sum z is passed through an activation function f to produce the neuron's output:

$$a = f(z) \tag{1.35}$$

1.2.1.4 | Common activation functions

By introducing non-linearity into the model, activation functions in neural networks enable complicated pattern recognition and learning. They help decide whether a neuron should be activated or not, based on the input it receives. Without activation functions, the network would behave like a linear regression model, limiting its ability to solve complex problems [15]. Listed below are the most common activation functions:

- *Sigmoid:*

$$f(z) = \frac{1}{1 + e^{-z}} \tag{1.36}$$

  - *Pros*: It provides a probabilistic interpretation, often used for binary classification problems.
  - *Cons*: It suffers from vanishing gradient problems and saturation for extreme values of z.

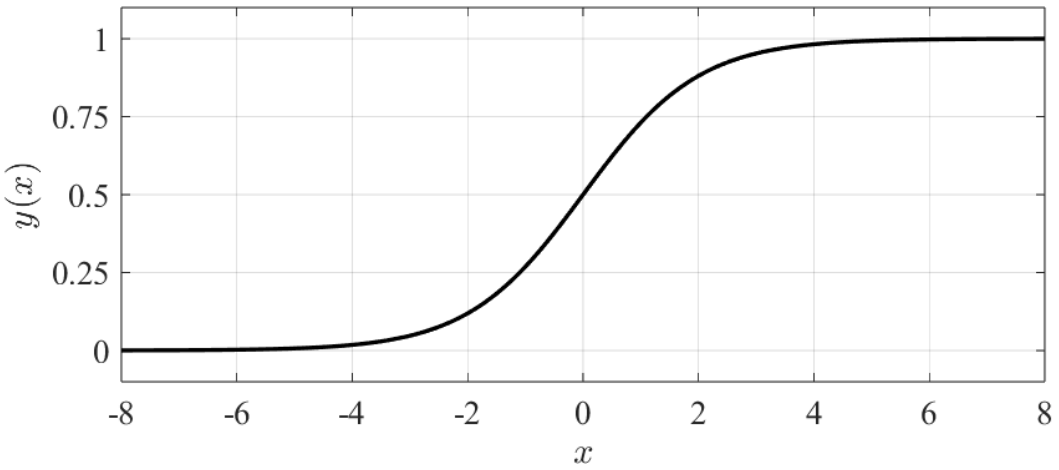


Figure 1.9 | Sigmoid activation function.

- *Hyperbolic tangent (tanh):*

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{1.37}$$

  - *Pros*: Values are centered around zero, which can help with convergence.
  - *Cons*: Also suffers from vanishing gradient problems.

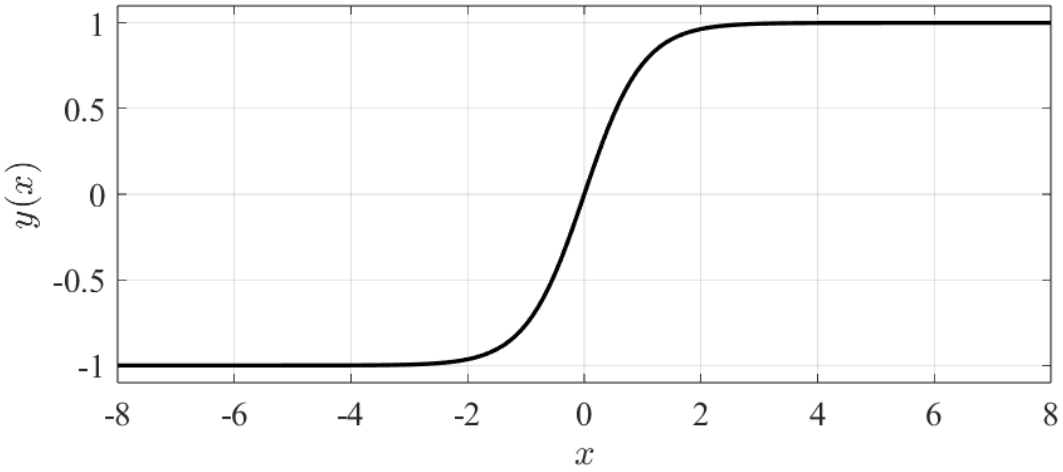


Figure 1.10 | Hyperbolic Tangent activation function.

- *Rectified Linear Unit (ReLU):*

$$f(z) = \max(0, z) \tag{1.38}$$

  - *Pros*: Simple and computationally efficient.

- *Cons*: It can suffer from "dead neurons" where neurons never activate for certain inputs (outputs remain zero).

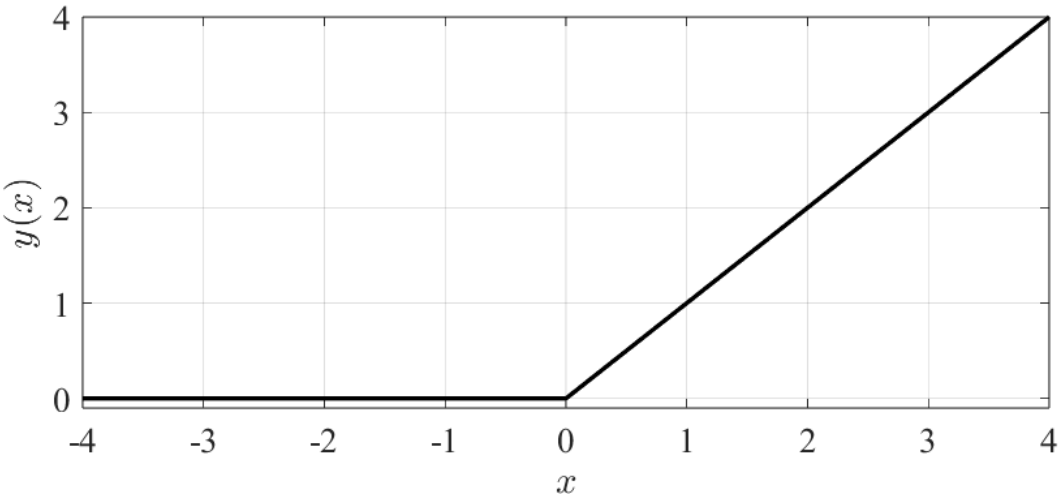


Figure 1.11 | Rectified Linear Unit activation function.

- *Leaky ReLU:*

$$f(z) = \begin{cases} z & \text{if } z > 0 \\ \alpha z & \text{if } z \leq 0 \end{cases} \tag{1.39}$$

  - *Pros*: It addresses the problem of dead neurons by allowing a small, non-zero gradient when $z \leq 0$.
  - *Cons*: It requires the selection of an appropriate α parameter.

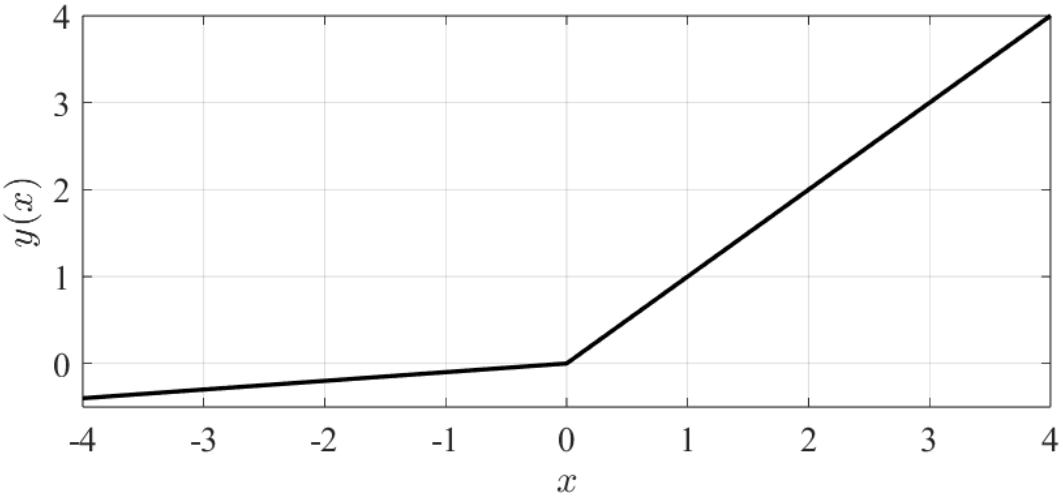


Figure 1.12 | Leaky ReLU activation function.

- *Parametric ReLU (PReLU):*

$$f(z) = \begin{cases} z & \text{if } z > 0 \\ \alpha z & \text{if } z \leq 0 \end{cases} \tag{1.40}$$

  - *Pros*: α is learned during training, potentially leading to better performance.
  - *Cons*: It adds additional parameters to the model, which may increase computational complexity.

- *Swish activation function:*

$$f(z) = \frac{z}{1 - e^{-z}} \tag{1.41}$$

  - *Pros*: It has a smoother gradient compared to ReLU, which can facilitate convergence during training.
  - *Cons*: It may incur higher computational costs compared to simpler alternatives like ReLU.

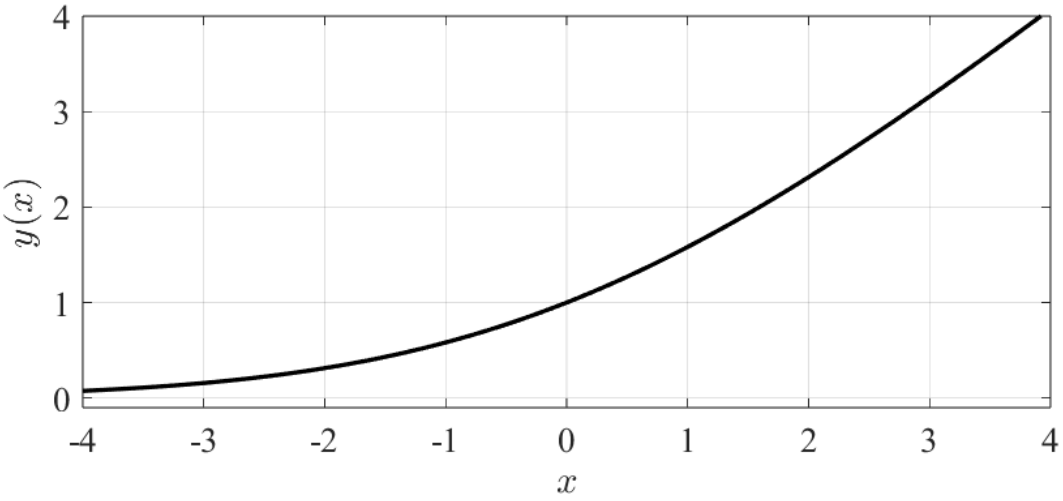


Figure 1.13 | Swish activation function.

- *Exponential Linear Unit (ELU):*

$$f(z) = \begin{cases} z & \text{if } z > 0 \\ \alpha(e^z - 1) & \text{if } z \leq 0 \end{cases} \tag{1.42}$$

  - *Pros*: It helps mitigate the vanishing gradient problem and allows for negative outputs.

- *Cons*: Computationally more expensive than ReLU.

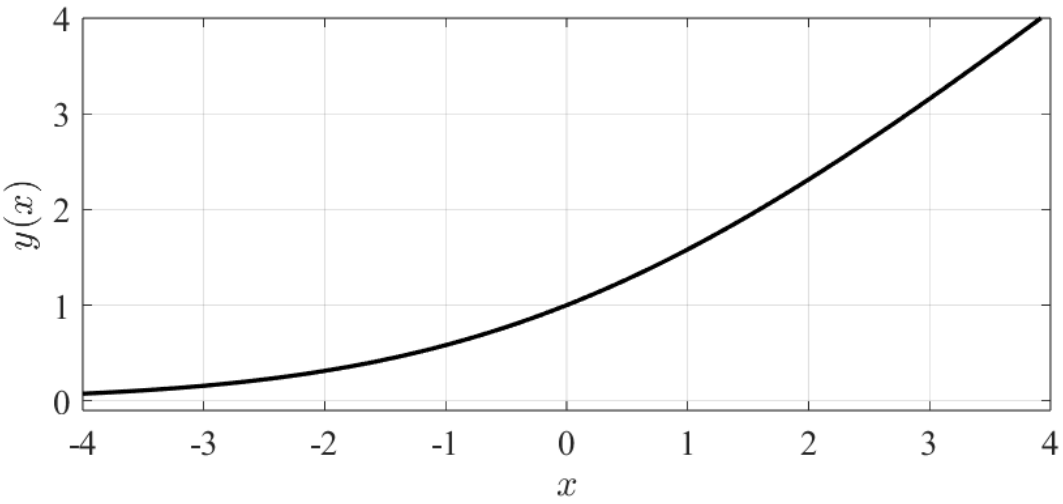


Figure 1.14 | Exponential Linear Unit activation function.

- *Scaled Exponential Linear Unit (SELU):*

$$f(z) = \lambda \begin{cases} z & \text{if } z > 0 \\ \alpha(e^z - 1) & \text{if } z \leq 0 \end{cases} \tag{1.43}$$

  - *Pros*: Self-normalizing properties that can lead to better performance in deep networks.
  - *Cons*: It requires careful initialization and use of specific architectures.

- *Softmax:*

$$f(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{1.44}$$

  - *Pros:* Converts a vector of values into a probability distribution, often used in the output layer for multi-class classification problems.
  - *Cons:* It can be sensitive to outliers due to the exponential function.

1.2.1.5 | Forward propagation

Forward propagation entails the transmission of data from the input layer through the hidden layers to the output layer. At each layer, the

neurons process the inputs, apply the weights, biases, and activation functions, and pass the outputs to the next layer [16].

Mathematically, for a NN with multiple layers, the output of layer l is given by:

$$a^{(l)} = f\left(W^{(l)}a^{(l-1)} + b^{(l)}\right) \tag{1.45}$$

Where:

- $a^{(l)}$ is the activation of the neurons in layer l.
- $W^{(l)}$ is the weight matrix for layer l.
- $b^{(l)}$ is the bias vector for layer l.

1.2.1.6 | Loss function

To train a NN, it is necessary to measure how well the network's predictions match the actual targets. This is done using a loss function L [17].

For example, the mean squared error (MSE) loss for regression tasks is:

$$L(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1.46}$$

Where y is the actual output and $\hat{y}$ is the predicted output.

1.2.1.7 | Backpropagation

Backpropagation is the process of calculating the gradient of the loss function concerning each weight and bias using the chain rule [18].

The backpropagation algorithm consists of computing the gradient of the loss function with respect to each weight and bias using the chain rule:

$$\delta^{(L)} = \nabla_a L \odot f'\left(z^{(L)}\right) \tag{1.47}$$

Where $\delta^{(L)}$ is the error term for the output layer, $\nabla_a L$ is the gradient of the loss concerning the output activations, and $f'\left(z^{(L)}\right)$ is the activation function derivative at the output layer.

For hidden layers:

$$\delta^{(l)} = \left(W^{(l+1)}\right)^T \delta^{(l+1)} \odot f'\left(z^{(l)}\right) \tag{1.48}$$

Where $\delta^{(l)}$ is the error term for layer l.

1.2.1.8 | Gradient descent

In gradient descent, weights are updated as follows:

$$\begin{aligned} W^{(l)} &= W^{(l)} - \alpha \frac{\partial L}{\partial W^{(l)}} \\ b^{(l)} &= b^{(l)} - \alpha \frac{\partial L}{\partial b^{(l)}} \end{aligned} \tag{1.49}$$

Where α is the learning rate [19].

There are three main variants of gradient descent:

- *Batch gradient descent*: Computes the cost function's gradient using the complete training dataset. It's computationally expensive for large datasets but converges smoothly.
- *Stochastic gradient descent (SGD)*: Computes the gradient using a single, randomly chosen data point. It's computationally efficient but has a high variance in the parameter updates, which can lead to noisy convergence.
- *Mini-batch gradient descent*: Strikes a balance between batch and stochastic gradient descent by using a small subset (mini-batch) of the training data to compute the gradient. It reduces the variance of parameter updates (compared to SGD) and is computationally more efficient than batch gradient descent. It allows for efficient use of vectorized operations and parallel processing on modern hardware (e.g., GPUs). The generic sizes of a mini-batch are typically chosen as powers of two, such as 32, 64, 128, 256. Powers of two are often efficient for memory allocation and utilization in computer systems. Larger mini-batches can provide a more

accurate estimate of the gradient but may be computationally slower. Smaller mini-batches introduce more noise into the gradient estimate but can converge faster due to more frequent updates.

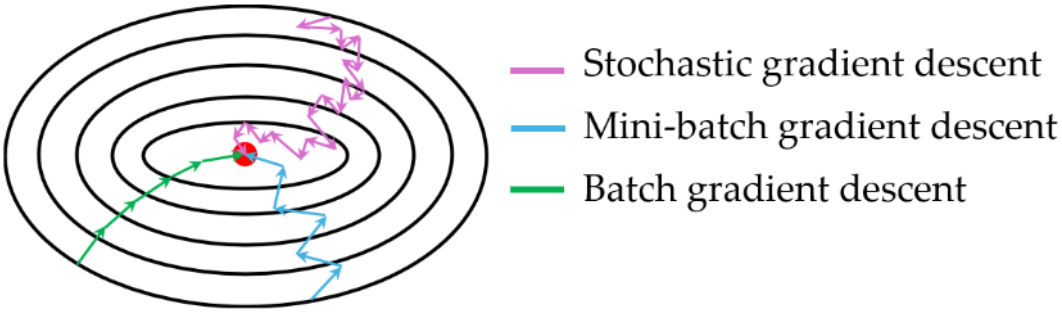


Figure 1.15 | Graphic representation of stochastic, mini-batch and batch gradient descent.

1.2.1.9 | Batch normalization

Batch normalization (BN) is a technique used in NNs to improve training speed and model stability [20] [21]. BN lowers the internal covariate shift by normalizing each layer's inputs. This means that the distribution of the inputs to each layer remains more stable, allowing for higher learning rates and faster convergence. Networks with batch normalization are less sensitive to the initial weights, as the normalization process mitigates the effects of poor initializations. Normalizing the inputs helps in maintaining the gradient magnitudes within a reasonable range, improving the gradient flow through the network and reducing the chances of vanishing or exploding gradients.

For each mini-batch, batch normalization computes the mean μ_B and variance σ_B^2 of the activations.

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{1.50}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$$

Where m is the number of samples in the mini-batch, and x_i represents the activation values.

The calculated mean and variance are used to normalize each activation. x_i is normalized using the computed mean and variance:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \tag{1.51}$$

The small constant ε is added for numerical stability, preventing division by zero.

After normalization, the activations are scaled and shifted using learnable parameters γ and β:

$$y_i = \gamma\hat{x}_i + \beta \tag{1.52}$$

γ and β allow the model to undo the normalization if necessary, ensuring that the transformed outputs can represent any possible distribution.

1.2.1.10 | Bias-variance trade-off

The bias-variance trade-off is a fundamental concept in ML that describes the relationship between a model's ability to fit the training data (bias) and its ability to generalize to unseen data (variance). This concept is particularly relevant in NNs, where it is crucial to balance these two aspects to achieve optimal performance.

The inaccuracy that results from using a simple model to approximate a complicated real-world problem is known as bias. High bias typically means that the model is too simple to capture the underlying patterns in the data. In the context of NNs, this might occur if the network has too few layers or neurons, leading to underfitting. Underfitting happens

when the model cannot capture the complexity of the training data, resulting in poor performance on both the training and test datasets.

Variance is the error that results from the model's sensitivity to small variations in the training set of data. High variance means that the model is too complex and fits the training data too closely, including its noise. This often happens when the NN has too many layers or neurons, or if it is trained for too many epochs. Such a model will perform very well on the training data but poorly on unseen data, a situation known as overfitting.

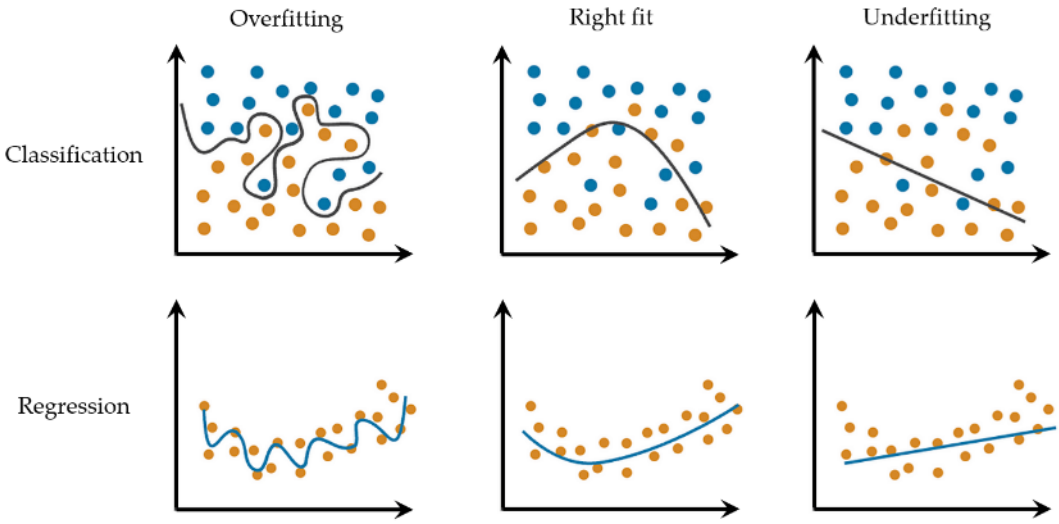


Figure 1.16 | Graphical representation of overfitting and underfitting issues for classification and regression problems.

The trade-off between these two types of error is known as the bias-variance trade-off:

- *High bias, low variance*: The model is too simple and does not capture the complexity of the data (underfitting). It has a high training error and does not improve significantly on the test set.
- *Low bias, high variance*: The model is overly complex and captures the noise in the training data (overfitting). It has a low training error but a high test error.

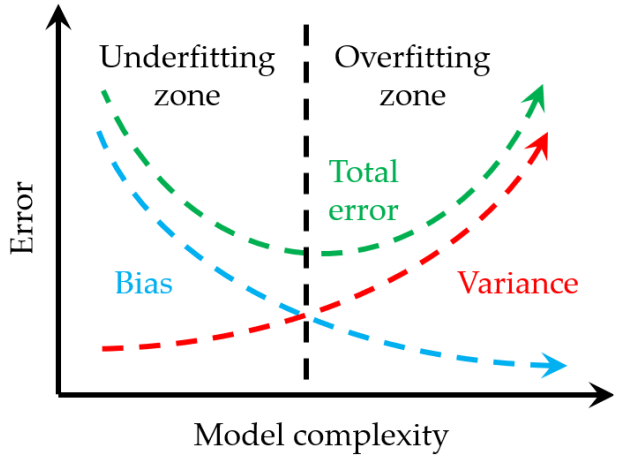


Figure 1.17 | Graphic representation of the bias-variance trade-off.

To manage the bias-variance trade-off in NNs, several strategies can be employed:

- *Model complexity*: Adjusting the number of layers and neurons. More layers and neurons increase the model's capacity to learn complex patterns but also increase the risk of overfitting.
- *Regularization*: Techniques such as L1/L2 regularization, dropout, and batch normalization can help prevent overfitting by adding constraints to the model.
- *Training data*: Increasing the amount of training data can help improve the model's generalization by providing more examples from which to learn.
- *Cross-validation*: Using techniques like *k*-fold cross-validation can provide a better estimate of model performance and help in selecting the optimal model complexity.
- *Early stopping*: Monitoring the model's performance on a validation set and stopping the training process when performance starts to deteriorate can prevent overfitting.

1.2.1.11 | Training, validation, and test phases

Three main phases are involved in identifying the optimal parameters of an AI model.

- *Training phase*:

  During training, the model learns from the training data. The goal is to minimize the training loss, which might lead to the following scenarios:

  - *High bias (underfitting)*: If the model is too simple (e.g., a linear model for a non-linear problem), it will have high bias and will not capture the underlying pattern of the data well. This can be detected if the training loss is high and the model performs poorly on both the training and validation datasets.
  - *High variance (overfitting)*: If the model is too complex (e.g., a very deep NN for a small dataset), it will have low training loss but might perform poorly on the validation dataset. This indicates that the model is fitting the noise in the training data rather than the actual signal.

- *Validation phase*:

  The validation dataset helps in monitoring the model's performance during training and tuning the model to avoid overfitting:

  - *Detecting overfitting*: If the validation loss starts to increase while the training loss continues to decrease, it indicates that the model is overfitting the training data.
  - *Hyperparameter tuning*: The validation loss is used to fine-tune the model's hyperparameters to find a balance between bias and variance.

- *Test phase*:

The test dataset provides a final unbiased evaluation of the model's performance. It helps to estimate how well the model generalizes to new, unseen data:

- *Generalization performance*: A well-balanced model with a good trade-off between bias and variance will have comparable performance (low loss and high accuracy) on both the validation and test datasets.
- *Final assessment*: If the test loss is significantly higher than the validation loss, it could indicate that the validation set was not representative, or there is still some overfitting.

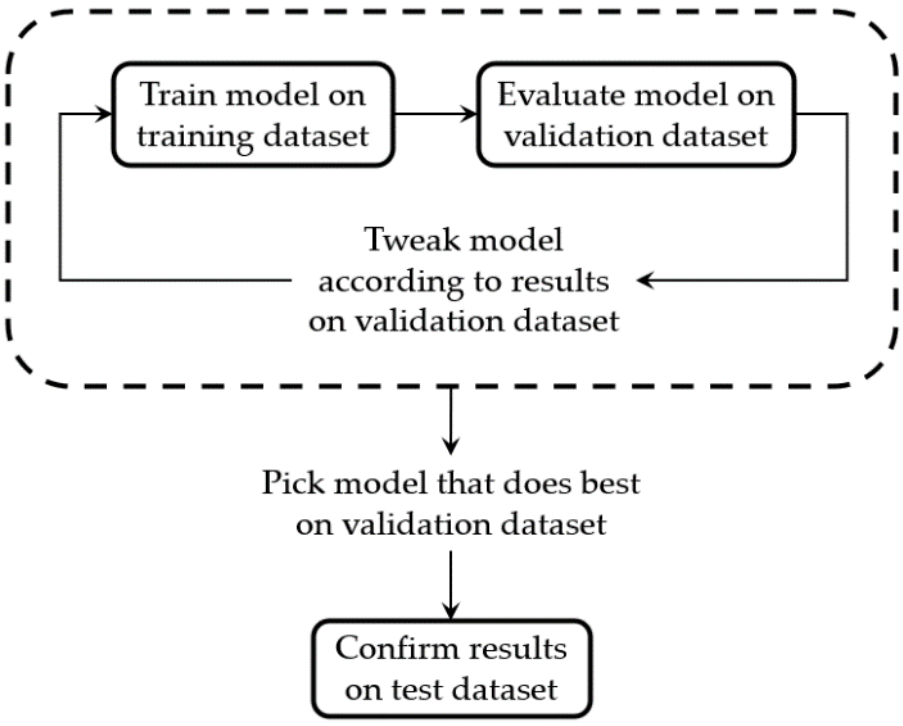


Figure 1.18 | Schematic representation of training, validation, and test phases.

1.2.1.12 | **Regularization**

To prevent overfitting, regularization techniques can be applied:

- *L1 regularization*: Adds a penalty proportional to the absolute values of the weights to the loss function:

$$L_{reg} = L + \lambda \sum_i |w_i| \tag{1.53}$$

Where λ is the regularization parameter that controls the strength of the regularization. A higher λ value increases the penalty, leading to more weights being driven towards zero.

This function is not differentiable at zero, which can complicate the optimization process. L1 Regularization tends to drive many weights to exactly zero, resulting in a sparse model. While this can be beneficial for feature selection, it may lead to models that discard potentially useful features if the regularization parameter λ is not chosen carefully. This can sometimes result in underfitting.

- *L2 regularization*: Adds a penalty proportional to the square of the magnitude of the weights to the loss function:

$$L_{reg} = L + \lambda \sum_i w_i^2 \tag{1.54}$$

The penalty term in L2 regularization is based on the square of the weights. This function is smooth and differentiable everywhere, which makes the optimization process more stable and efficient. L2 regularization shrinks all weights but does not drive them to zero. This typically results in a more balanced reduction of weights, preserving more features.

- *Dropout regularization*: Randomly sets a fraction of the activations to zero during training to prevent the network from becoming too reliant on any single neuron.

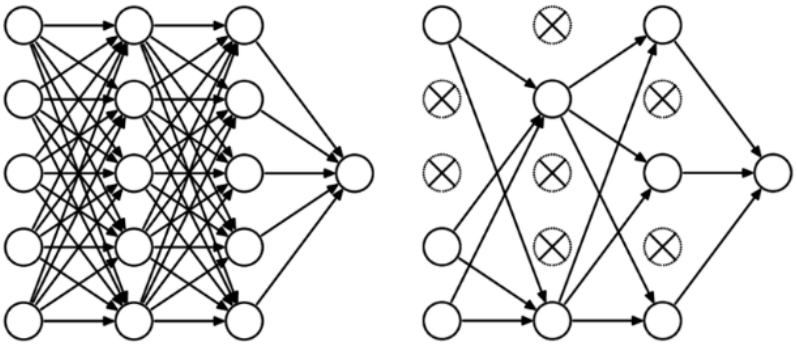


Figure 1.19 | Schematic representation of dropout regularization.

1.2.1.13 | Hyperparameters

Hyperparameters are parameters whose values govern the learning process and dictate the values of model parameters learned by a learning algorithm. When the learning algorithm is undergoing training, hyperparameters are employed. They are established before the start of the model training. At the end of the training, the model parameters are obtained [22]. Setting the proper hyperparameter values is critical since it has a direct influence on the model's performance. Hyperparameter optimization refers to the process of selecting the optimal hyperparameters to train the model [23] [24]. Examples of hyperparameters are:

- *Number of epochs* [25]: The number of complete passes through the training dataset. More epochs can lead to better learning but also increase the risk of overfitting.
- *Initial learn rate* [26]: The Learn Rate governs the rate at which the algorithm learns the NN model parameters. At the beginning of the learning process the model parameters are far from those that minimize the cost function [27], so it will be useful to have a high initial learning rate for a faster decreasing.
- *Learn rate drop factor* [28]: As the value of the cost function approaches the minimum, it is necessary to decrease the learning rate to avoid oscillations around the minimum and allow convergence of the

model parameters. It is a factor to drop the learning rate that is expressed as a scalar ranging from 0 to 1.

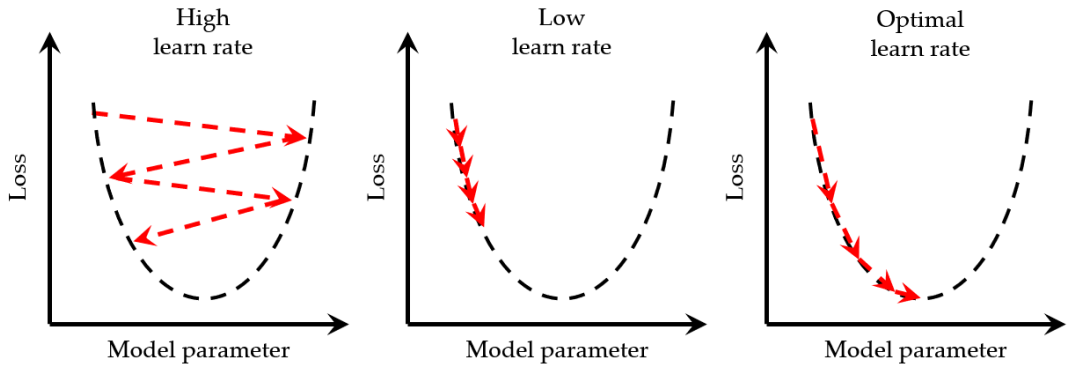


Figure 1.20 | Decrease in loss with respect to a model parameter as approaching toward the minimum for different values of the learn rate.

- *Learn rate drop period* [28]: It is a positive integer that specifies the number of epochs [29] for dropping the learning rate.

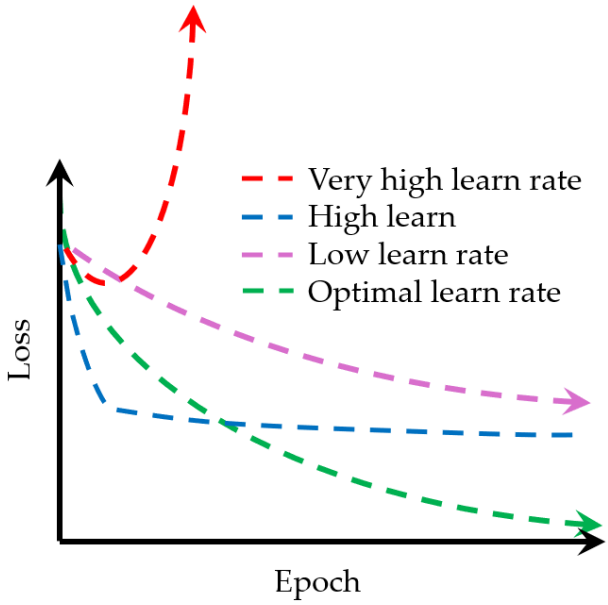


Figure 1.21 | Loss decrease as epochs increase for different values of learn rate.

- *Hidden units* [30]: It is the number of values calculated as output from a hidden layer and input to the next layer. The same number of nodes was chosen for each hidden layer.

- *L1 regularization coefficient* [31]: It penalizes the absolute sum of the model parameters. Overfitting of the model on the training set is prevented by this regularisation parameter.
- *L2 regularization coefficient* [32]: It penalizes the quadratic sum of the model parameters. This regularisation parameter keeps the model from overfitting to the training set.
- *Dropout probability* [33]: Dropout is a regularisation method. During training, some layer outputs are dropped or ignored at random. It reduces the reliance of each unit in the hidden layer on other units in the hidden layers, which helps to prevent overfitting on the training dataset.
- *Mini batch size* [34]: it is the number of samples used in each portion of the epoch with which the model parameters are updated approaching the minimum of the cost function.

1.2.1.14 | Hyperparameter tuning

Hyperparameter tuning is a crucial aspect of ML model optimization, where the goal is to find the best set of hyperparameters that maximize the model's performance. Three methods used for hyperparameter tuning are:

- *Grid search*: Grid search is a systematic method where you define a grid of hyperparameter values and evaluate the model performance for each combination of values within the grid [35]. It exhaustively searches through all the specified hyperparameter values to find the optimal set based on a predefined performance metric.
  - *Process*:
    1. Define ranges or discrete values for each hyperparameter of interest.

2. Construct a grid where each point represents a unique combination of these values.
3. Train and evaluate the model for each combination
4. Select the combination that yields the best performance metric.

- *Pros*: Straightforward and easy to implement. It guarantees to find the optimal combination within the specified grid.
- *Cons*: Computationally expensive, especially with a large number of hyperparameters or wide ranges of values. It may not efficiently allocate resources to more promising areas of the hyperparameter space.

- *Random search*: Random search differs from grid search by sampling hyperparameter values randomly from predefined distributions [36]. Instead of exploring every combination exhaustively, random search focuses on randomly selected points in the hyperparameter space.
  - *Process*:
    1. Define probability distributions for each hyperparameter.
    2. Sample a specified number of combinations randomly from these distributions.
    3. Train and evaluate the model for each randomly selected combination.
  - *Pros*:
    1. More efficient than grid search in many cases.
    2. It allows the exploration of a broader range of hyperparameter values.

- *Cons*:

  No guarantee of finding the optimal combination, although it often performs better than grid search in practice.

- *Bayesian optimization* [37] [38]: Bayesian optimization is an advanced technique that uses probabilistic models to model the objective function (model performance) and decides where to sample the next set of hyperparameters based on the outcomes of previous evaluations. It balances exploration (searching in less explored regions) and exploitation (focusing on regions likely to yield high performance).
  - *Process*:
    1. Define the objective function: The objective function $f(x)$ to be optimized is defined. In the context of neural networks, x represents the set of hyperparameters, while $f(x)$ could be the performance metric, like accuracy or validation loss. The goal is to find the optimal set of hyperparameters that maximizes (or minimizes) $f(x)$.
    2. Build a probabilistic surrogate model: A Gaussian Process (GP) is typically used as the surrogate model to approximate the unknown objective function. This model estimates both the expected value of $f(x)$ and the uncertainty about this estimate. The GP provides a probabilistic view of the function by modeling the mean $\mu(x)$ and the variance $\sigma(x)$, allowing the optimization to be more informed when selecting new points to evaluate. This uncertainty helps the optimizer decide whether to try

new hyperparameters (exploration) or refine good ones (exploitation).

3. Choose the acquisition function: The activation function gives a way to decide which set of hyperparameters to evaluate next. A common acquisition function is the Upper Confidence Bound (UCB), which selects points by considering both the predicted mean $\mu(x)$ and the uncertainty $\sigma(x)$.

$$UCB(x) = \mu(x) + \zeta \cdot \sigma(x) \tag{1.55}$$

ζ controls the balance between exploration and exploitation. A large value of ζ tends to prioritize regions with limited or no samples (exploration). A small value of ζ value leads to concentrated sampling around the current best solution (exploitation). An intermediate value of ζ strikes a balance between well-known optimal areas and unexplored regions.

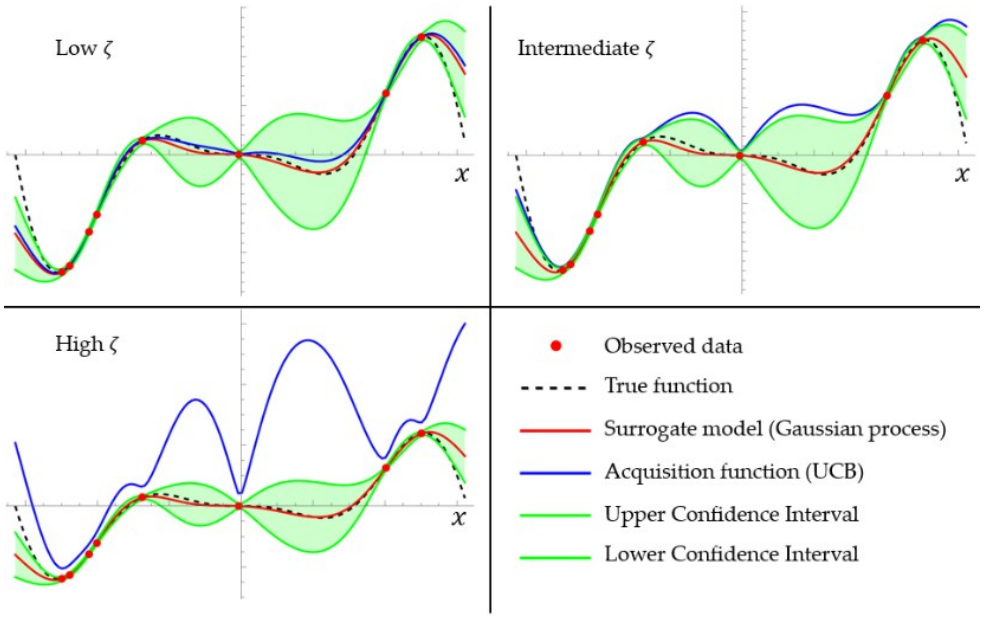


Figure 1.22 | Graphical representation of the effect of ζ variation in the UCB acuisition function.

4. Evaluate the objective function: After selecting a new set of hyperparameters based on the acquisition function, the true objective function $f(x)$ is evaluated. For neural networks, this involves training the model with the chosen hyperparameters and obtaining the performance metric (e.g., accuracy). This step is computationally expensive, as it requires full training or validation of the model.
5. Update the surrogate model: The Gaussian Process model is updated with the newly acquired data point from the previous step. This means that the surrogate model's predictions and uncertainties are refined, making it more accurate. The new data allows the GP to better approximate the underlying objective function, guiding future decisions on where to evaluate next.
6. Iterate and converge: The process of selecting hyperparameters via the acquisition function, evaluating them, and updating the surrogate model is repeated. This loop continues until a stopping criterion is met, such as a maximum number of iterations or achieving a satisfactory performance level. Over time, the Bayesian Optimization algorithm converges to the optimal set of hyperparameters, reducing the need for excessive evaluations.

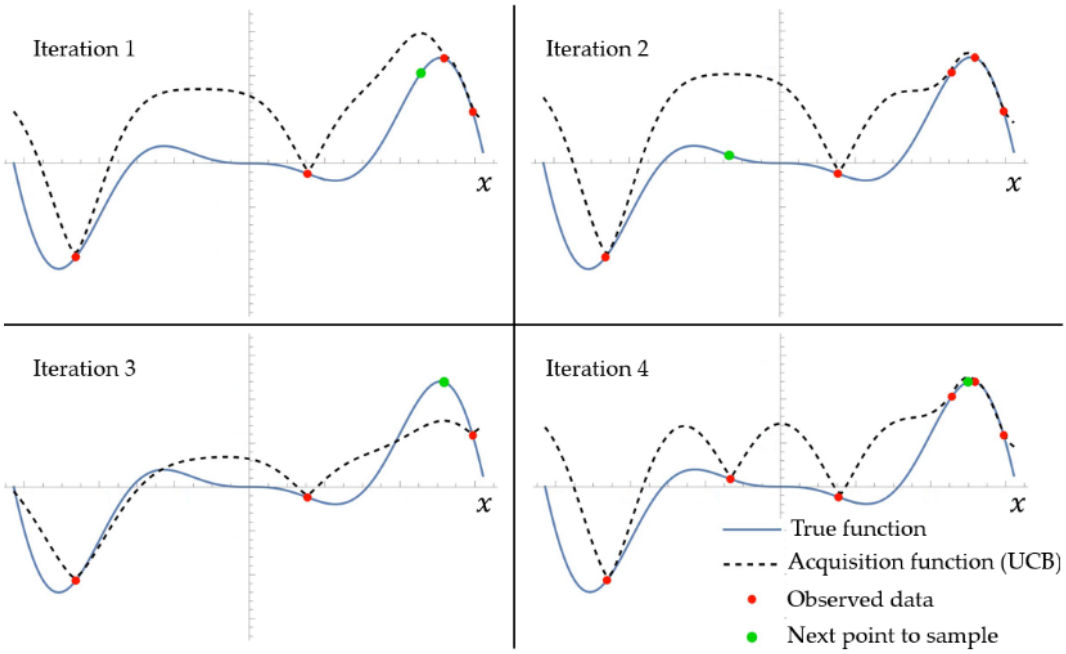


Figure 1.23 | Graphical representation of the recursive steps of Bayesian Optimization.

- *Pros*:
  1. It efficiently finds the optimal hyperparameters with fewer evaluations compared to grid and random search.
  2. It can adaptively allocate resources to more promising areas of the hyperparameter space.
- *Cons*:
  1. Requires tuning of its own hyperparameters, such as the choice of surrogate model and acquisition function.
  2. Computationally more intensive than grid and random search.

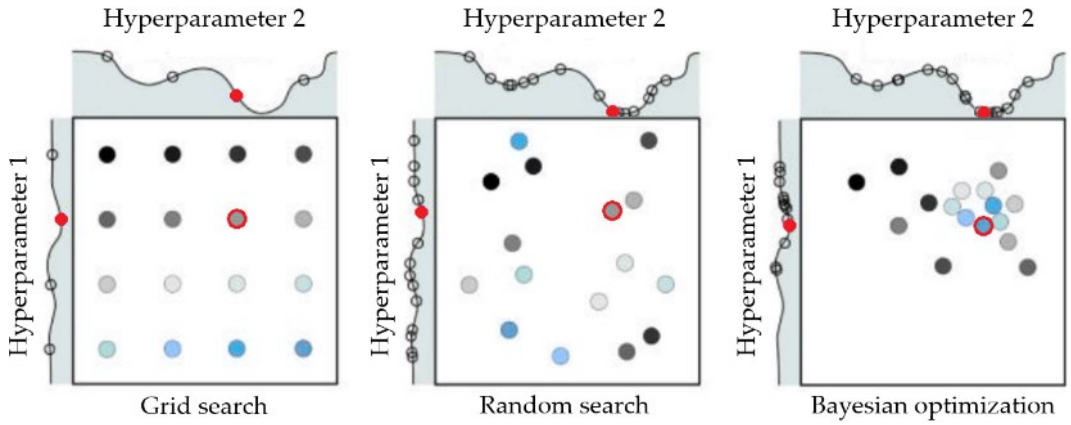


Figure 1.24 | Graphical comparison of tuning techniques for hyperparameters: grid search, random search and Bayesian optimization.

Figure 1.24 graphically shows the decrease in the loss on the validation dataset when optimizing two hyperparameters with the three techniques of grid search, random search, and bayesian optimization. The loss on the validation dataset with respect to the two hyperparameters is identified by a surface, on the horizontal and vertical axes are shown the values with respect to only one of the hyperparameters and the corresponding calculated values with empty black circles. The attempts are identified in ascending order from black to blue color by the filled inner circles. The attempts that resulted in the calculation of the minimum value of the loss on the validation set are identified by the red color. Given the same number of attempts, Bayesian optimization is able to identify the set of hyperparameters that returns the lowest value of calculated loss on the validation set compared with both random search and grid search.

1.2.1.15 | Setting up an optimization problem

Three concepts are key to setting up the training and optimization of a NN:

- *Normalizing inputs*: Normalizing inputs involves adjusting the values of input features to a standard scale, typically to improve the efficiency and stability of the training process for neural networks.

The main goal of normalizing inputs is to ensure that each input feature contributes equally to the learning process. Without normalization, features with larger numerical ranges can dominate the learning process, making it difficult for the network to converge. Normalization helps in faster convergence during training, reduces the risk of getting stuck in local minima, and improves the overall performance and accuracy of the model.

Common normalization methods include:

- *Min-max scaling*: Rescaling the features to lie within a specific range, usually $[0, 1]$ or $[-1, 1]$.
- *Standardization (Z-score normalization)*: Transforming the data to have a mean of 0 and a standard deviation of 1.

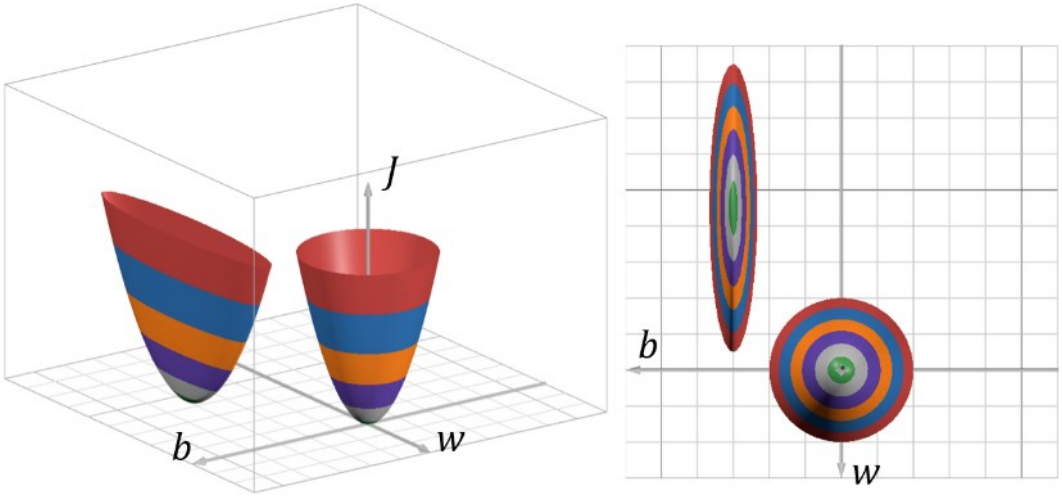


Figure 1.25 | Graphical representation of the effect of input normalization on the cost function assuming a model consisting of only two parameters.

- *Normalizing outputs*: Normalizing outputs involves adjusting the values of output variables to a standard scale, which is crucial especially when dealing with models predicting multiple output variables simultaneously. When a model needs to predict multiple output variables, it's essential to normalize these outputs. This

ensures that each output variable contributes fairly to the learning process, even if they have different value ranges. Normalizing multiple outputs helps ensure that each output variable influences the model's learning process proportionally. This improves the stability of the learning process, model convergence, and prediction accuracy. Similar normalization methods used for inputs can be applied to multiple outputs, such as Min-Max scaling or standardization.

- *Weight initialization*: Weight initialization refers to the process of setting the initial values of the weights in a NN before training begins. Proper weight initialization is crucial for training deep networks effectively. Poor initialization can lead to issues such as vanishing or exploding gradients, which hinder the learning process. Common methods for initialization include [39] [40] [41]:
    - *Zero initialization*: Setting all weights to zero, which is generally not used because it prevents the network from learning effectively.
    - *Random initialization*: Initializing weights with small random values from a normal or uniform distribution.
    - *Kaiming He initialization*: Initializing weights based on the size of the previous layer (specifically, using a normal distribution with mean 0 and variance $\frac{2}{n}$, where n is the number of input units).
    - *Xavier/Glorot initialization*: Initializing weights with a distribution that depends on the size of the input and output layers, typically using a normal distribution with mean 0 and

variance $\frac{2}{n_{in}+n_{out}}$, where the numbers for the input and output units are, respectively, n_{in} and n_{out}.

1.2.1.16 | Optimization algorithms: Momentum

The momentum optimization algorithm uses an exponentially weighted moving average of the gradients to adjust the NN weights more efficiently than simple methods like stochastic gradient descent.

$$E[g]_t = \mathrm{B}E[g]_{t-1} + (1 - \mathrm{B})g_t \tag{1.56}$$

Where:

- g_t is the gradient of the weights at step t.
- $E[g]_t$ is the exponentially weighted average of the gradients at step t.
- B is the decay parameter (typically between 0.9 and 0.999) that controls the decay rate of the moving average.

The weights θ are updated using the momentum:

$$\theta_{t+1} = \theta_t - \alpha E[g]_t \tag{1.57}$$

Where:

- α is the learning rate.

Momentum helps in accelerating convergence by accumulating past gradients, enabling smoother and faster updates to the weights. It reduces oscillations and improves stability in gradient descent by smoothing out the update process.

1.2.1.17 | Optimization algorithms: RMSProp

RMSProp (Root Mean Square Propagation) uses an exponentially weighted moving average of the squared gradients to adjust the NN weights more efficiently than simple methods like stochastic gradient descent.

The exponential moving average is computed as:

$$E[g^2]_t = \Gamma E[g^2]_{t-1} + (1-\Gamma) g_t^2 \tag{1.58}$$

Where:

- $E[g^2]_t$ is the exponentially weighted average of the squared gradients at step t.
- Γ is the decay parameter (typically between 0.9 and 0.999) that controls the decay rate of the moving average.

The weights θ are updated using the normalized gradient divided by the square root of the accumulated squared gradients:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{E[g^2]_t + \varepsilon}} g_t \tag{1.59}$$

where:

- ε is a small term (e.g., 10^{-8}) added for numerical stability to avoid division by zero.

RMSProp adjusts the learning rate for each parameter based on the magnitude of its gradients, which can lead to faster convergence. It performs well in scenarios where gradients have varying scales.

1.2.1.18 | Optimization algorithms: ADAM

Adam (Adaptive Moment Estimation) is an optimization algorithm that combines the advantages of both momentum optimization and RMSProp. It dynamically adjusts the learning rate for each parameter, providing faster convergence and better handling of sparse gradients in NN training.

At each time step t Adam computes the first and the second moment:

$$\begin{aligned} E[g]_t &= \mathrm{B} E[g]_{t-1} + (1-\mathrm{B}) g_t \\ E[g^2]_t &= \Gamma E[g^2]_{t-1} + (1-\Gamma) g_t^2 \end{aligned} \tag{1.60}$$

The weights θ are updated using both momentum:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{E[g^2]_t + \varepsilon}} E[g]_t \tag{1.61}$$

Adam combines the benefits of both momentum optimization and RMSProp, leveraging momentum to accelerate gradients in relevant directions and RMSProp to scale learning rates based on the magnitudes of recent gradients.

Adam's hyperparameters B, Γ, and ε typically have default values that work well across various applications, but fine-tuning may be necessary for specific problems.

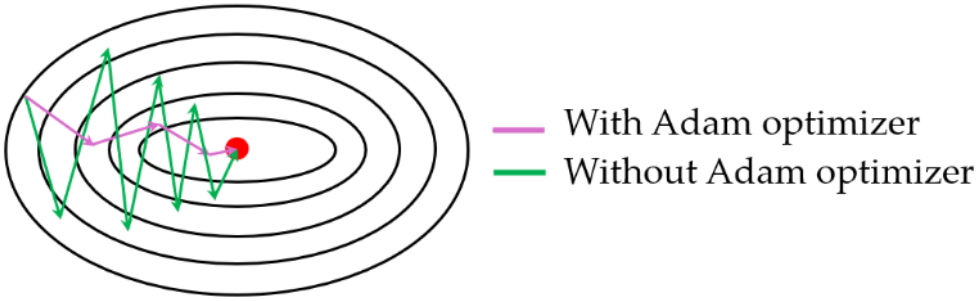


Figure 1.26 | Graphical representation of the effect of the Adam optimizer during the training process.

1.2.2 | Strengths and Weaknesses

ML-based estimation methods, such as neural networks, have gained traction for their ability to handle nonlinearities and learn from large datasets. For instance, dynamic neural networks can predict complex vehicle dynamics without requiring explicit system models. However, their black-box nature and lack of theoretical guarantees pose challenges for safety-critical applications. Additionally, these methods often require extensive training data, which may not always be available or cover the full range of operating conditions. Data-driven techniques offer adaptability and robustness in uncertain environments, but their opacity and dependency on data quality limit their standalone applicability. Integrating these methods with model-based techniques can address these weaknesses, providing both reliability and adaptability.

2 | OVERVIEW OF CONTROL TECHNIQUES

Control systems are integral to modern technology, governing the behavior of dynamic systems. They are designed to regulate and manage the performance of various systems. The two primary types of control systems are open-loop and closed-loop (feedback) systems [42] [43] [44].

The control action and the output are unrelated in an open-loop control system. These systems do not use feedback to adjust the input, relying solely on pre-determined instructions. Open-loop systems are simpler and more cost-effective but less accurate and adaptable.

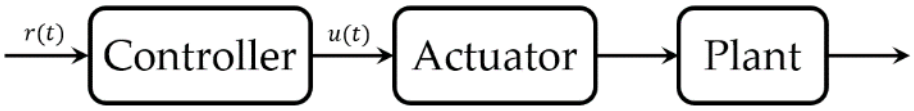


Figure 2.1 | Open loop control system.

Closed-loop control systems use feedback to compare the actual output with the desired output (reference). The difference between these values (error) is used to adjust the input to minimize the error and achieve the desired output.

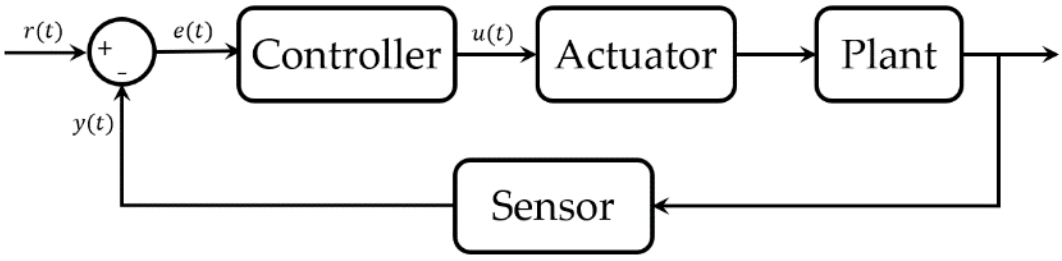


Figure 2.2 | Closed loop control system.

2.1 | Proportional-Integral-Derivative Control (PID-Control)

PID control is the most widely used control technique, combining the advantages of Proportional (P), Integral (I), and Derivative (D) controls [45] [46]. It provides a balanced approach to improve stability, response time, and accuracy.

$$e(t) = r(t) - y(t)$$
$$u(t) = K_p \cdot e(t) + K_i \int_{t_0}^{t} e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{2.1}$$

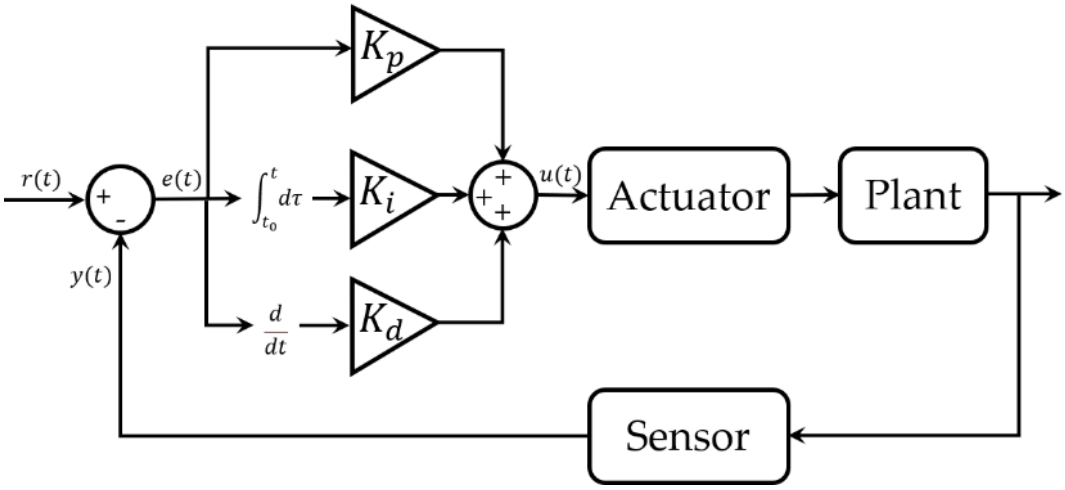


Figure 2.3 | Schematic representation of a PID control system.

The difference between the expected output, $r(t)$, and the actual output, $y(t)$, is the tracking error, denoted by the variable $e(t)$.

This error signal $e(t)$ is fed to the PID controller, and the controller computes both the derivative and the integral of this error signal concerning time. The control signal $u(t)$ to the plant is equal to the proportional gain K_p times the magnitude of the error plus the integral gain K_i times the integral of the error plus the derivative gain K_d times the derivative of the error.

This control signal $u(t)$ is fed to the plant and the new output $y(t)$ is obtained. The new output $y(t)$ is then fed back and compared to the reference to find the new error signal $e(t)$. After receiving this incoming error signal, the controller updates the control input. As long as the controller is active, this operation keeps going.

Finding the Laplace transform of Equation () yields the transfer function of a PID controller.

$$K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s} \tag{2.2}$$

Increasing the proportional gain (K_p) proportionally increases the control signal for a given level of error. This means the controller will respond more aggressively to an error, causing the closed-loop system to react more quickly. However, this often results in more overshoot. Additionally, increasing K_p tends to reduce, but not eliminate, steady-state error.

Adding a derivative term to the controller (K_d) enables the controller to anticipate the error. With only proportional control and a fixed K_p, the control signal increases only if the error increases. However, with derivative control, the control signal can become large if the error starts to slope upward, even if the magnitude of the error is still relatively small. This anticipation effect tends to add damping to the system, thereby decreasing overshoot. However, the derivative term does not affect the steady-state error. Incorporating an integral term into the controller (K_i) helps to reduce steady-state error. If there is a persistent, steady error, the integrator accumulates this error, increasing the control signal and driving the error down. A drawback of the integral term is that it can make the system more sluggish and oscillatory, as it may take some time for the integrator to unwind when the error signal changes sign. The general effects of each

controller parameter (K_p, K_d, K_i) on a closed-loop system are summarized in the Table 2.1.

Gain	Rise time	Overshoot	Settling time	Steady-state error
K_p	Decrease	Increase	Small change	Decrease
K_i	Decrease	Increase	Increase	Decrease
K_d	Small change	Decrease	Decrease	No change

Table 2.1 | Effects of variation of the coefficients $\boldsymbol{K_p}$, $\boldsymbol{K_i}$, and $\boldsymbol{K_d}$ on the dynamic behavior of the system.

Where:

- *Rise time*: The rise time is the time it takes for a system's response to go from a specific lower percentage to a higher percentage of its final value. Typically, it's measured between 10% and 90% of the final value. This metric indicates how fast the system responds initially to an input or disturbance.
- *Overshoot*: Overshoot refers to the extent to which the system's response exceeds its final or desired value. It is typically expressed as a percentage of the final value. Overshoot occurs due to system inertia and is usually undesirable, as it indicates temporary deviation beyond the target.
- *Settling time*: Settling time is the time required for the system's response to remain within a specific tolerance band around its final value (usually within 2% or 5% of the final value) without further oscillations. It indicates how long it takes for the system to stabilize after a disturbance or change in input.
- *Steady-state error*: Steady-state error is the difference between the system's final output and the desired input when the system has settled and all transient effects have dissipated. It indicates how

accurately the system follows the desired input over time and represents a system's long-term accuracy.

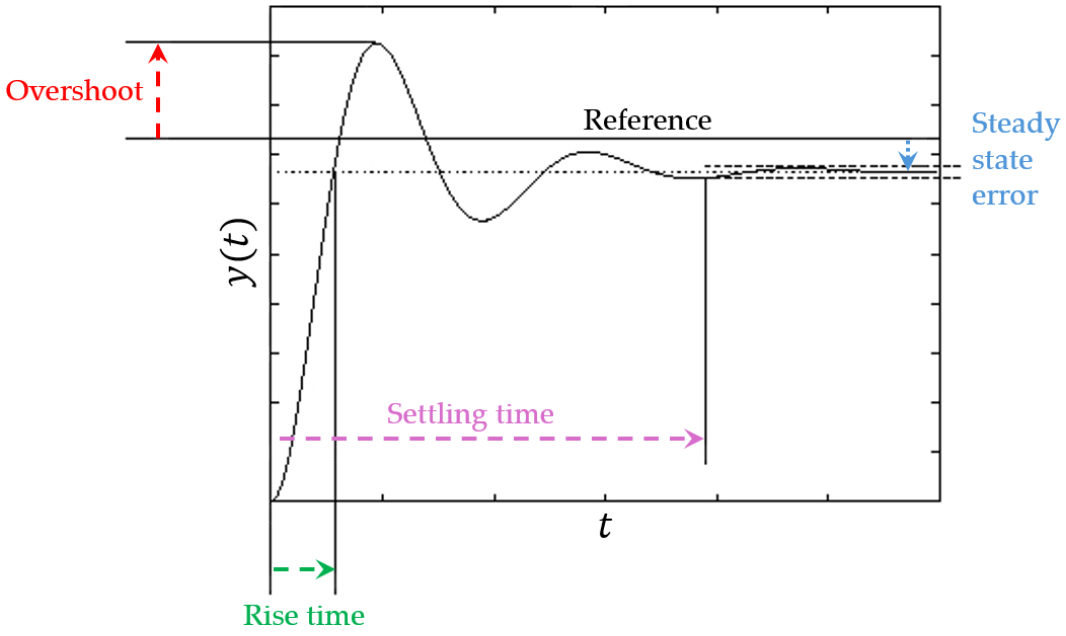


Figure 2.4 | Graphical representation of rise time, overshoot, settling time and steady state error.

These guidelines apply in many cases, but not all. To truly understand the impact of tuning the individual gains, more analysis or testing on the actual system is necessary.

2.1.1 | Strengths and Weaknesses

PID control is a simple and effective technique widely used in vehicle dynamics for tasks such as maintaining speed or controlling suspension systems. Its primary advantage lies in its ease of implementation and tuning. However, it struggles with handling multivariable systems, time delays, and constraints, which are critical in autonomous vehicle applications. PID control is well-suited for basic tasks but falls short in addressing the complexity of modern vehicle dynamics. Its limitations highlight the need for more advanced methods, such as Model Predictive

Control (MPC), in applications requiring higher levels of precision and adaptability.

2.2 | Linear Quadratic Regulator (LQR)

The Linear Quadratic Regulator (LQR) is an optimal control strategy that operates within the framework of linear dynamic systems. It aims to regulate the behavior of a system to minimize a cost function, typically a quadratic function representing the trade-off between the system's performance and control effort. The LQR is renowned for its robust performance and mathematical elegance in handling linear systems.

The LQR controller operates on systems described by linear time-invariant (LTI) state-space models:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.3}$$

Where:

- $x(t)$ is the state vector at time t.
- $u(t)$ is the control input vector.
- A is the state matrix.
- B is the input matrix.

The goal of the LQR is to determine the control input u(t)u(t)u(t) that minimizes a quadratic cost function of the form:

$$J = \int_0^\infty (x^T Q x + u^T R u) dt \tag{2.4}$$

Where:

- Q is a positive semi-definite matrix that penalizes the state x.
- R is a positive definite matrix that penalizes the control input u.

The cost function J represents a trade-off between state regulation and control effort. By adjusting Q and R, different performance characteristics

can be prioritized. For example, increasing the entries of Q relative to R emphasizes state accuracy, while increasing R relative to Q emphasizes control effort minimization.

The matrices Q and R are designed to reflect the relative importance of state deviation and control effort.

The optimal control law for the LQR problem is given by a state feedback law:

$$u(t) = -Kx(t) \tag{2.5}$$

where K is the feedback gain matrix. This gain matrix K is determined by solving the Algebraic Riccati Equation (ARE):

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \tag{2.6}$$

Here, P is a positive definite matrix, known as the solution to the ARE. The ARE can be solved efficiently using numerical algorithms. This makes LQR suitable for real-time applications. Once P is found, the optimal gain matrix K is computed as:

$$K = R^{-1}B^T P \tag{2.7}$$

The LQR controller guarantees that the closed-loop system $\dot{x}(t) = (A - BK)x(t)$ is stable if the pair (A, B) is controllable and the matrices Q and R are chosen appropriately.

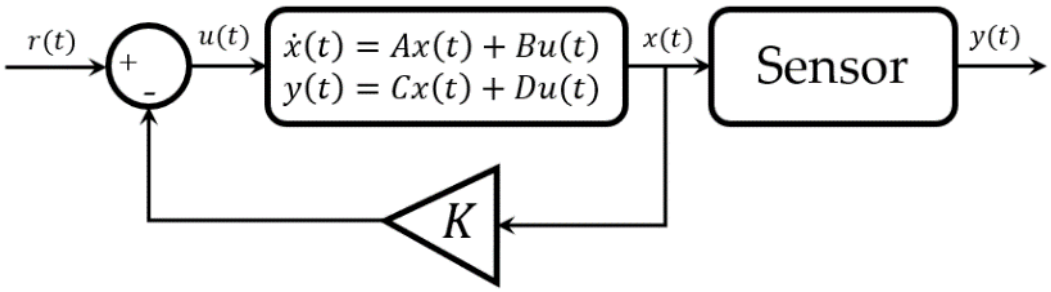


Figure 2.5 | Schematic representation of a Linear Quadratic Regulator.

2.2.1 | Strengths and Weaknesses

LQR provides optimal control by minimizing a quadratic cost function, making it an effective tool for balancing performance and energy efficiency. Its analytical formulation ensures stability and robustness under certain conditions. However, its applicability is constrained by its reliance on linear models, which may not adequately capture the nonlinearities inherent in vehicle dynamics. While LQR excels in providing systematic and theoretically sound solutions, its reliance on linearization can lead to suboptimal performance in highly nonlinear systems. Hybrid approaches or extensions such as nonlinear LQR can partially mitigate these drawbacks.

2.3 | Model Predictive Control (MPC)

Model Predictive Control (MPC) is an advanced control strategy used widely in the field of control engineering for complex dynamic systems [47] [48] [49]. Unlike traditional control methods, MPC explicitly utilizes a model of the system to predict future states and optimize control actions. This predictive capability allows MPC to handle multi-variable control problems with constraints, making it highly effective for industrial processes, robotics, automotive applications, and more.

The cornerstone of MPC is its reliance on a mathematical model of the system. This model describes the relationship between the inputs, states, and outputs of the system. The model can be derived from first principles, empirical data, or a combination of both. The types of models commonly used in MPC include linear time-invariant (LTI), linear time-varying (LTV), and nonlinear models.

MPC predicts the future behavior of the system over a finite prediction horizon, which is a sequence of future time steps. At each time step, the

controller solves an optimization problem to find the optimal control inputs that drive the system towards desired future states while minimizing a cost function. This cost function typically includes terms for tracking error (difference between predicted outputs and desired outputs) and control effort.

The optimization in MPC is performed in a receding horizon fashion. This means that only the first control action of the optimal sequence is implemented, and the process is repeated at the next time step with updated state information. This approach provides robustness to modeling errors and disturbances because the optimization is continuously updated based on the latest system measurements.

MPC is particularly powerful because it can handle constraints on inputs, states, and outputs directly within the optimization problem. These constraints can represent physical limitations, safety requirements, or operational boundaries. The ability to manage constraints explicitly makes MPC suitable for real-world applications where such limitations are common.

2.3.1 | Mathematical formulation of MPC

A discrete-time system is represented by the following state-space equations:

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) \\ y_k &= h(x_k, u_k) \end{aligned} \tag{2.8}$$

where x_k is the state vector, u_k is the control input vector, and y_k is the output vector at time step k. For linear systems, these equations can be simplified to:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k \end{aligned} \tag{2.9}$$

Where A, B, C, and D are matrices of appropriate dimensions.

The cost function J to be minimized over the prediction horizon N is typically defined as:

$$J = \sum_{i=0}^{N-1} (\|y_{k+i} - y_{k+i}^{ref}\|_Q^2 + \|u_{k+i}\|_R^2) \tag{2.10}$$

where y_{k+i}^{ref} is the reference output, Q and R are weighting matrices for the tracking error and control effort, respectively.

The constraints are incorporated as:

$$\begin{aligned} x_{k+i} &\in \chi \\ u_{k+i} &\in \mathrm{U} \\ y_{k+i} &\in \mathrm{Y} \end{aligned} \tag{2.11}$$

Where χ, U, and Y represent feasible sets for states, inputs, and outputs, respectively.

The MPC optimization problem at each time step can be formulated as:

$$\min_{u_k, u_{k+1}, \ldots, u_{k+N-1}} \in \chi \tag{2.12}$$

subject to:

$$\begin{aligned} x_{k+i+1} &= f(x_{k+i}, u_{k+i}), i = 0, \ldots, N-1 \\ x_{k+i} &\in \chi \\ u_{k+i} &\in \mathrm{U} \\ y_{k+i} &\in \mathrm{Y} \end{aligned} \tag{2.13}$$

The solution to this problem provides the optimal control inputs u_k, $u_{k+1}, \ldots, u_{k+N-1}$. The first one, u_k, is applied to the plant.

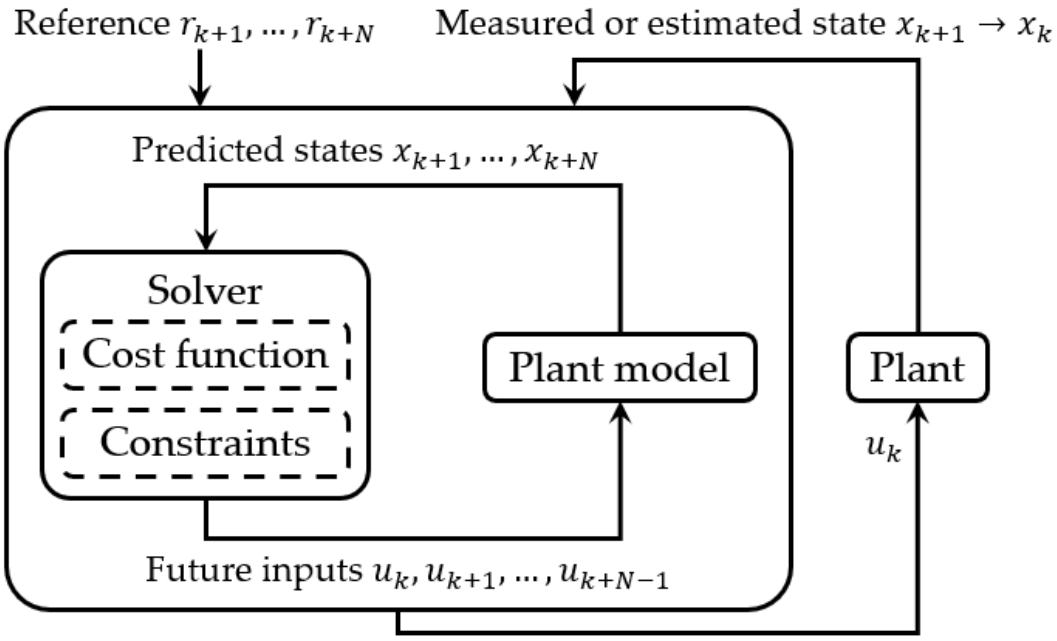


Figure 2.6 | Schematic representation of an MPC.

In practice, the true state x_k of the system may not be directly measurable. Therefore, state estimation techniques, such as the Kalman filter, are often used to estimate the state from available measurements.

The optimization problem in MPC can be computationally intensive, especially for large-scale systems with long prediction horizons or nonlinear models. Efficient solvers and numerical algorithms, such as quadratic programming (QP) for linear MPC or sequential quadratic programming (SQP) for nonlinear MPC, are employed to solve these problems in real-time. Real-time implementation of MPC requires the controller to solve the optimization problem within each sampling period. This necessitates fast and reliable computational resources. Advances in embedded systems and parallel computing have made real-time MPC feasible for many applications.

2.3.2 | Strengths and Weaknesses

MPC is a powerful control technique that explicitly accounts for system constraints and predicts future behavior over a finite horizon. This makes it particularly effective for complex tasks like path tracking and obstacle avoidance. However, the computational burden of solving optimization problems in real time remains a challenge, particularly for high-speed vehicle dynamics. MPC offers unparalleled flexibility and robustness, but its high computational requirements can hinder real-time implementation. Efficient solvers and approximations are essential to make MPC viable for real-world applications.

2.4 | Reinforcement Learning (RL)

In recent years, traditional control techniques have increasingly been complemented by data-driven approaches. This paradigm shift is driven by advancements in computational power, the availability of large datasets, and the development of sophisticated algorithms. As a result, control systems are now leveraging data to enhance performance, improve adaptability, and provide more robust solutions in various applications.

The goal of a control system is to determine the correct inputs into a system that will generate the desired system behavior. In feedback control systems, the controller uses state observations to improve performance and correct for random disturbances and errors. Engineers utilize this feedback, along with a model of the plant and environment, to design the controller to meet the system requirements. This concept, although simple in theory, can become complex when the system is difficult to model, highly nonlinear, or has large state and action spaces. For instance, developing a control system for a walking robot involves commanding numerous motors

for the joints, processing state observations from various sensors, and satisfying multiple requirements such as balance, obstacle avoidance, and disturbance rejection. Traditionally, this problem is approached by breaking it into smaller sections, each solved independently. However, the interactions between control loops and the complexity of system requirements make design and tuning challenging. Recently, data-driven techniques like Reinforcement Learning (RL) have begun to complement traditional methods [50] [51] [52] [53]. This approach is gaining traction due to its ability to learn and adapt from interactions with the environment, making it highly effective for complex and dynamic systems. RL enhances traditional techniques by providing a robust framework for decision-making and optimization in uncertain conditions. It enhances the ability to handle large state and action spaces and adapt to uncertainties, making it a valuable addition to the control design toolkit.

RL works with data from a dynamic environment. The goal is to find the best sequence of actions that will generate the optimal outcome. RL solves this problem allowing a piece of software called an agent to explore, interact with, and learn from the environment.

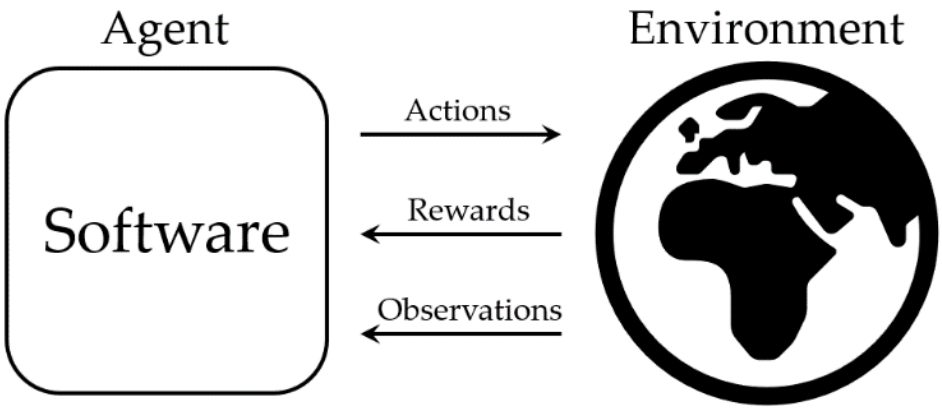


Figure 2.7 | Schematic representation of the interaction between agent and environment.

The agent can observe the current state of the environment. Based on this observed state, it decides which action to take. The environment then changes state and provides a reward for that action, both of which the agent receives. Using this new information, the agent can determine whether the action was beneficial and should be repeated or if it was not and should be avoided. This observation-action-reward cycle continues until learning is complete.

An internal function of the agent converts state observations (the inputs) to actions (the outputs). This is the single function that takes the place of all of the individual subcomponents of the control system. This function is referred to as the policy in the RL terminology. The policy selects the course of action based on a set of observations.

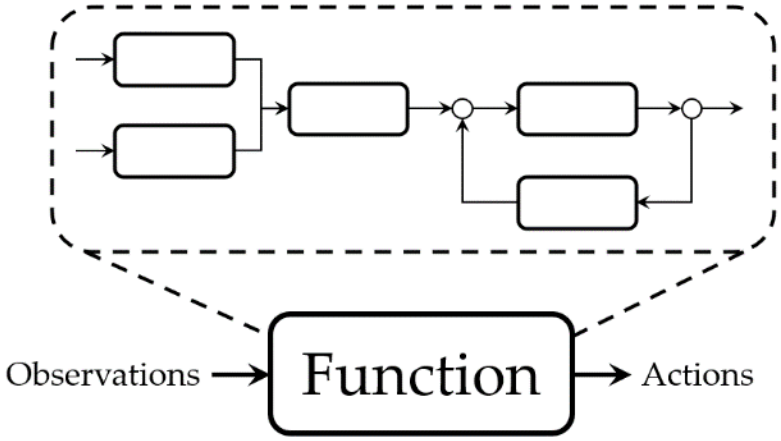


Figure 2.8 | Replacement of subcomponents of a control system with the single function of the RL.

The control problem would be solved by designing a perfect policy that accurately commands the right actuators for every observed state. However, achieving this is challenging in most situations, and even a perfect policy may become suboptimal as the environment changes over time. This is where RL algorithms come into play. RL algorithms adapt the policy based on the actions taken, observations from the environment, and

the rewards collected. The goal of the agent using RL algorithms is to learn the optimal policy through interaction with the environment, ensuring that it always takes the most rewarding action in the long run, regardless of the state.

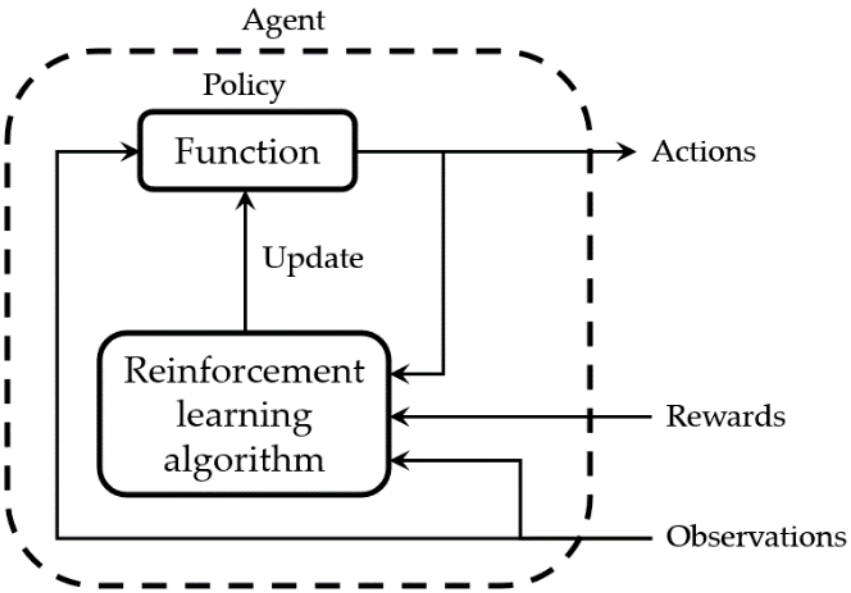


Figure 2.9 | Exploded representation of an agent with Function and RL algorithm.

The goal of RL is the same of the traditional control problems; it represents the same concepts using different terms. Both methods aim to determine the correct inputs into a system to achieve the desired behavior. In both approaches, the objective is to design the policy (or controller) that maps the observed state of the environment (or plant) to the optimal actions (actuator commands). The state feedback signal corresponds to observations from the environment, while the reference signal is incorporated into both the reward function and the environment observations.

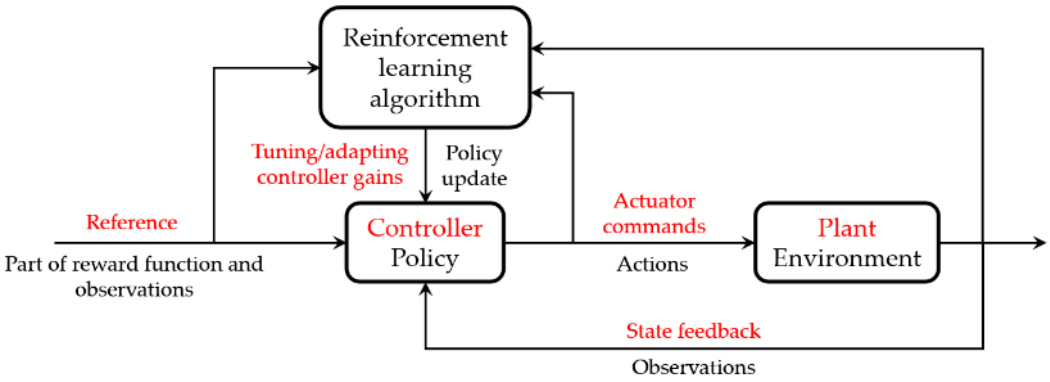


Figure 2.10 | RL algorithm vs Traditional control system.

Five different areas need to be addressed with RL:

- Environment.
- Reward.
- Policy.
- Training.
- Deployment.

2.4.1 | Environment

In RL, the environment encompasses everything outside of the agent. It is where the agent sends actions and from which rewards and observations are generated.

This definition can be confusing from a control perspective, where the environment is often viewed as disturbances impacting the system being controlled.

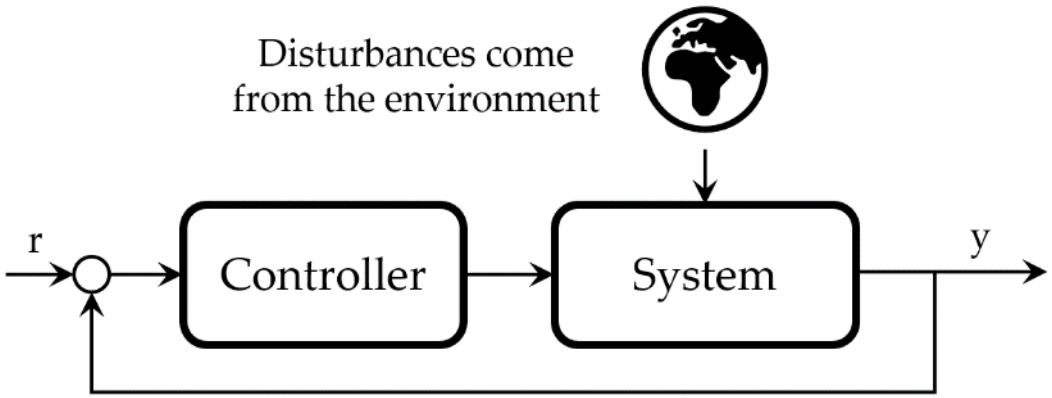


Figure 2.11 | Environment in the perspective of a traditional control system.

However, in RL terminology, the environment includes all aspects except the agent, encompassing the system dynamics. Thus, most of the system is considered part of the environment, while the agent is the software responsible for generating actions and updating the policy through learning.

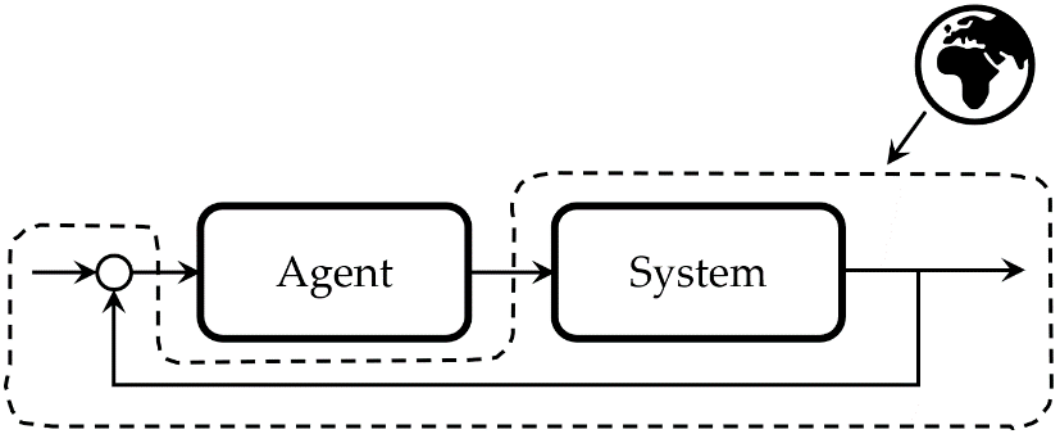


Figure 2.12 | Environment in the perspective of a traditional control system.

One reason RL is so powerful is that the agent does not need any prior knowledge about the environment to learn how to interact with it. For instance, the agent does not need to understand the dynamics or kinematics of a walking robot; it can still determine how to maximize rewards without knowing the specifics of joint movements or appendage lengths. An RL-equipped agent can be placed into any system and learn the optimal policy,

provided it has access to observations, rewards, actions, and sufficient internal states.

Since the agent learns through interaction with the environment, it is essential to provide a way for the agent to interact, either with a real physical environment or a simulation. The choice between real and simulated environments depends on various factors. A real environment offers unmatched accuracy and simplicity, as it requires no model creation or validation, and it may be necessary if the environment is constantly changing or difficult to model accurately. However, simulations provide significant advantages in terms of speed, safety, and control. Simulations can run faster than in real-time and be parallelized, thus accelerating the learning process. They also allow modeling of conditions that are difficult to test and eliminate the risk of hardware damage. For instance, training an agent with a physical pendulum setup might be feasible due to its simplicity and low risk, but training a walking robot in the real world could result in damage and inefficiencies. Simulated environments are commonly used to train agents, especially when existing system models are available from traditional control design. Utilizing simulations, one can replace existing controllers with RL agents, add reward functions, and commence the learning process. Since learning requires numerous samples, simulations running faster than real-time and in parallel can significantly speed up this process. Additionally, simulations offer greater control over conditions, such as simulating low-friction surfaces like ice, which aids in preparing the robot for various real-world surfaces. Therefore, simulations can create more effective training environments for RL agents.

2.4.2 | Reward

Once the environment is established, the next critical step is defining what actions the agent should take and how it will be rewarded for doing so. This involves crafting a reward function that helps the learning algorithm discern when the policy is improving and ultimately converges on the desired outcome [54]. A reward function generates a scalar value representing the goodness of an agent's state and action.

$$reward = function(state, action) \tag{2.14}$$

This concept mirrors the cost function in Linear Quadratic Regulator (LQR) theory, which penalizes poor performance and excessive actuator effort. The key distinction is that while LQR aims to minimize the cost, a reward function in RL seeks to maximize the reward, with rewards essentially acting as the negative of costs.

$$J = \int_0^\infty (x^T Q x + u^T R u) dt \tag{2.15}$$

Unlike the quadratic cost functions in LQR, RL allows for a more flexible reward structure, including sparse rewards, time-step-based rewards, or rewards given only at the end of an episode. This flexibility means rewards can be derived from complex, nonlinear functions or numerous parameters, tailored to the needs of effective training. Sparse rewards, where the agent only receives feedback after a lengthy sequence of actions, can lead to inefficiencies as the agent may spend considerable time exploring without receiving feedback, thus making it unlikely to discover the optimal action sequence by chance. To address this, reward shaping is used to provide intermediate rewards that guide the agent toward the desired behavior. Yet, reward shaping can introduce its own challenges, as shortcuts in reward functions may lead to suboptimal solutions. For instance, a poorly designed

reward function for a walking robot might cause the agent to adopt unintended behaviors, such as crawling instead of walking, which may appear optimal based on the rewards but is not aligned with the overall goal. Reward shaping isn't always used to solve the problems of sparse rewards. Reward shaping is also employed to incorporate domain-specific knowledge, helping to guide the agent more effectively. For example, if one wants the robot to walk rather than crawl along the ground, one can reward the agent for keeping the robot's trunk at a walking height. Additionally, rewards could be given for low actuator effort, maintaining an upright position for longer periods, and staying on the intended path. Despite its potential, crafting an effective reward function is complex and often requires extensive experimentation to avoid poorly designed rewards that fail to achieve the desired results.

Furthermore, reinforcement learning involves balancing exploration and exploitation. Since RL operates online, the agent's actions influence the data it receives, creating a tradeoff between exploiting known rewarding actions and exploring unknown areas of the state space [55].

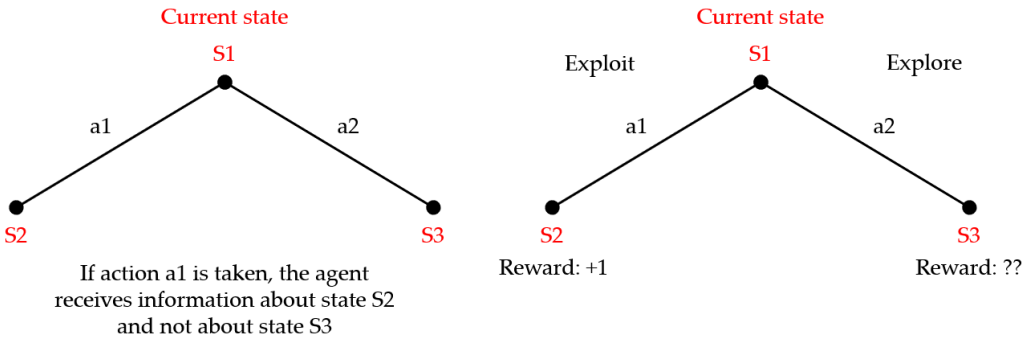


Figure 2.13 | Graphic representation of the difference between exploitation and exploration: Example 1.

For example, if an agent knows that going left yields a reward while going right yields a penalty, it might prefer to always go left. However, if

the agent does not explore right, it may miss out on potentially better rewards elsewhere.

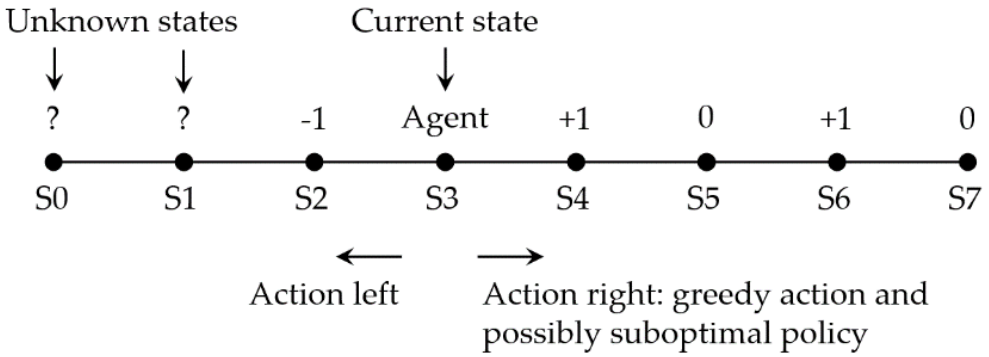


Figure 2.14 | Graphic representation of the difference between exploitation and exploration: Example 2.

Thus, a balance must be struck between exploiting known rewards and exploring new possibilities, similar to how students might explore various career options before settling on a path. While pure exploration can help find global solutions, it is inefficient and potentially risky in physical environments. For example, when driving down a highway, the potential harm that could arise from an autonomous vehicle exploring arbitrary steering wheel inputs should be considered. Simulated environments offer a controlled setting where exploration can be more extensive without risk. As the agent learns, it usually explores more at the beginning and shifts towards exploitation as it gathers more information.

Another crucial aspect of RL is the concept of value, which assesses the long-term rewards of a state or action rather than just the immediate reward. The value of a state represents the total rewards expected from that state into the future. For instance, if an agent can choose between actions yielding short-term rewards, it may ultimately achieve a higher total reward by choosing actions with greater long-term value.

	Going left has higher value		← Agent →	Going right is the better option	
Reward	+5	-1	0	+1	-1
Value	+4			+1	
State	S0	S1	S2	S3	S4

Figure 2.15 | Difference between Reward and Value.

However, predicting future rewards can be uncertain, and immediate rewards might be more valuable than distant ones.

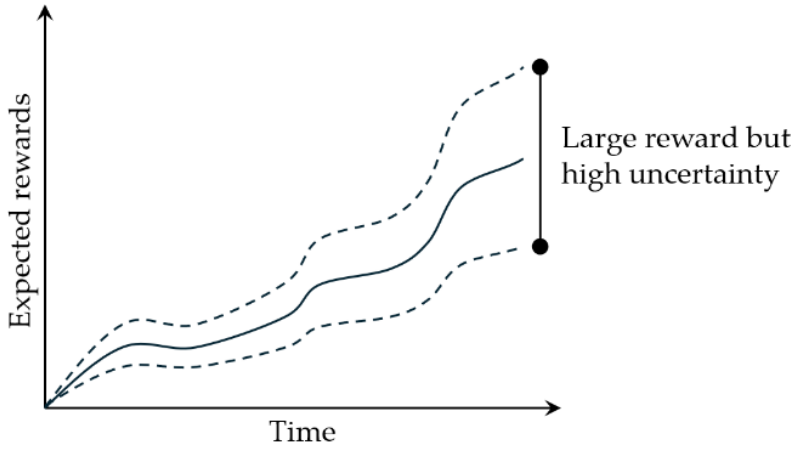


Figure 2.16 | Graphical representation of the increase in uncertainty in predicting reward as time increases.

To account for this, RL uses a discount factor (gamma) to adjust how future rewards are valued, balancing between immediate and long-term benefits. This discount factor, ranging between 0 and 1, helps manage the tradeoff between immediate gratification and future gains, ensuring a more balanced approach to learning and decision-making.

$$Total\ discounted\ reward = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \cdots = \sum_{i=1}^{T} \gamma^{i-1} r_i \tag{2.16}$$

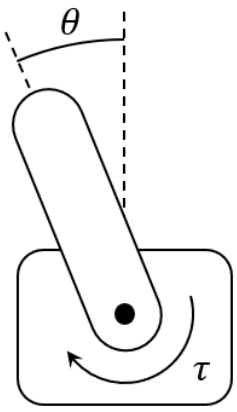
2.4.3 | Policy

In RL, the agent is crucial for determining how to interact with the environment, and it consists of two main components: the policy and the learning algorithm. The policy is a function that maps observations from the environment to actions taken by the agent, while the learning algorithm optimizes this policy to achieve the best possible performance.

$$Policy \rightarrow actions = function(state\ observations) \tag{2.17}$$

Policies can be structured in two main ways: directly or indirectly. A direct policy has a specific mapping from state observations to actions, whereas an indirect policy relies on other metrics, such as value estimates, to infer the optimal actions.

For environments with discrete and manageable state and action spaces, a simple table, like a Q-table, can represent the policy by mapping state-action pairs to their respective values. The policy then chooses the action with the highest value for any given state. However, this table-based approach becomes impractical when dealing with large or continuous state and action spaces, a challenge known as the curse of dimensionality. For instance, controlling an inverted pendulum requires handling a continuous range of states (angles and angular rates) and actions (motor torques), making a table infeasible. To address this, continuous functions can be used to represent the policy, and NNs offer a robust solution.



Angle rate	Angle	Torque: -0.01	-0.001	0.02	0.021
-0.3	0.124	Value	Value	Value	Value
0.17	0.137	Value	Value	Value	Value
0.175	0.139	Value	Value	Value	Value
0.223	0.204	Value	Value	Value	Value

$value = -(\dot{\theta}^2 + \theta^2) + \tau$? $value = \dot{\theta}\sin\theta + \tau$?

Figure 2.17 | Policy examples: Q-table and continuous function.

NNs, with their nodes and layers, serve as universal function approximators capable of learning complex, nonlinear mappings between states and actions. Instead of manually designing intricate functions, NNs can be trained to approximate the desired function through systematic adjustments of their parameters. This adaptability and capability to generalize across various environments make NNs a powerful tool in reinforcement learning, despite the complexity involved in setting up and training these networks.

2.4.3.1 | Policy function-based, value function-based, and actor-critic

In RL, the agent's policy, represented by NNs, is intricately linked with the chosen RL algorithm. Three primary approaches to RL, policy function-based, value function-based, and actor-critic, demonstrate the differences in policy structures [56] [57].

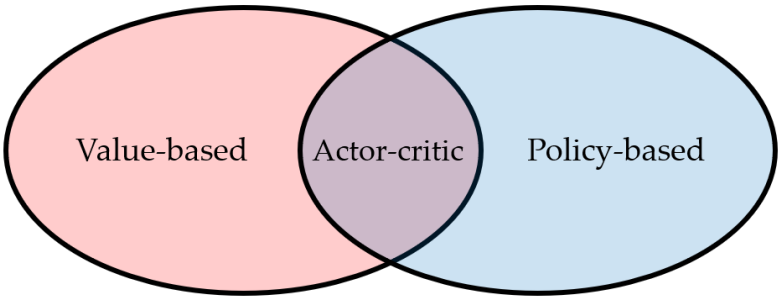


Figure 2.18 | Actor-critic as the intersection of value-based and policy-based approaches.

Policy function-based algorithms train an NN, known as the actor, to map state observations directly to actions.

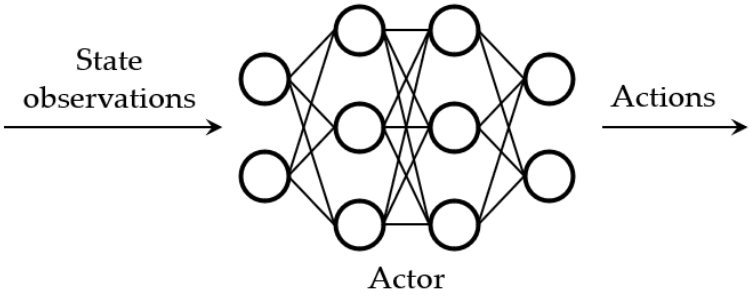


Figure 2.19 | Representation of the actor, policy-based function, by NN.

These methods often employ policy gradients, which adjust action probabilities based on received rewards to balance exploration and exploitation. A critical aspect of this balance is the exploration-exploitation tradeoff: the agent must decide between exploiting the environment by choosing actions that yield known rewards or exploring new actions that might lead to even greater rewards. A stochastic policy addresses this tradeoff by incorporating exploration into the action probabilities. When learning, the agent updates these probabilities, nudging them towards actions that yield higher rewards. Over time, the probabilities of the most advantageous actions increase, leading the agent to consistently choose those actions.

The agent determines the effectiveness of actions by executing the current policy, collecting rewards along the way, and then updating the network to increase the probabilities of actions that lead to higher rewards. This adjustment involves taking the derivative of each weight and bias in the network concerning reward and modifying them to increase positive rewards. This process is known as gradient ascent, which explains the term "policy gradient."

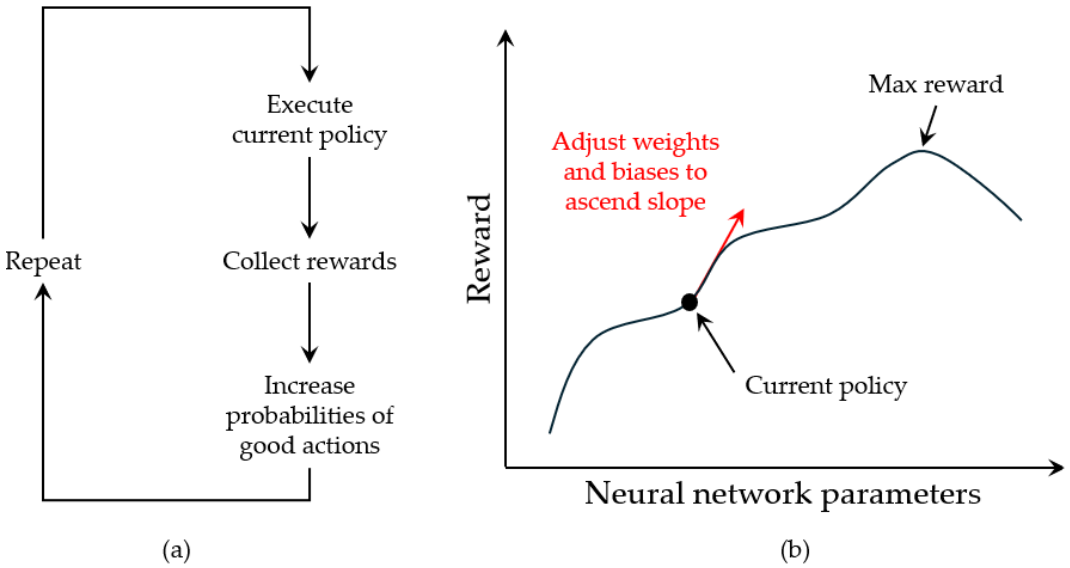


Figure 2.20 | Schematic (a) and graphical (b) representation of the gradient ascent process in the training of a policy-based function.

Despite their utility, policy gradient methods can suffer from slow convergence and sensitivity to noisy rewards. The naive approach of following the direction of the steepest ascent may lead to convergence on a local maximum rather than a global one. Additionally, the high variance in cumulative rewards can slow down convergence.

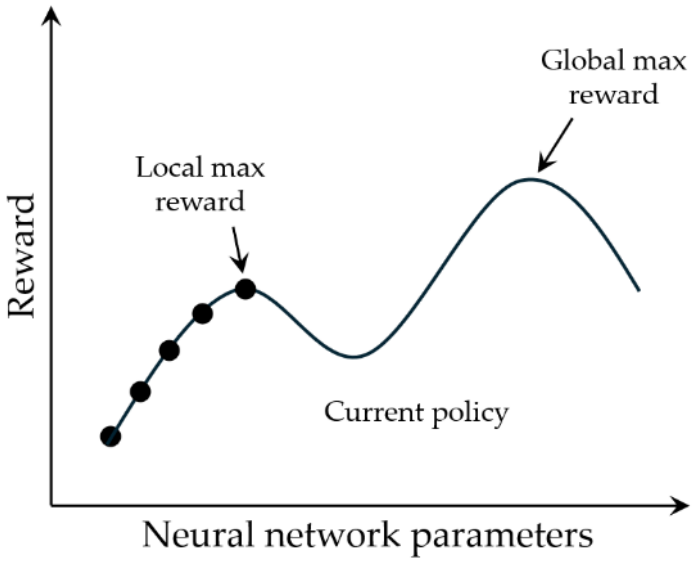


Figure 2.21 | Problem of convergence to a local minimum during gradient ascent.

Conversely, value function-based methods use a critic to evaluate the value of each action from a given state, guiding the policy to select the highest value actions.

$$value = function(state\ observations, actions) \tag{2.18}$$

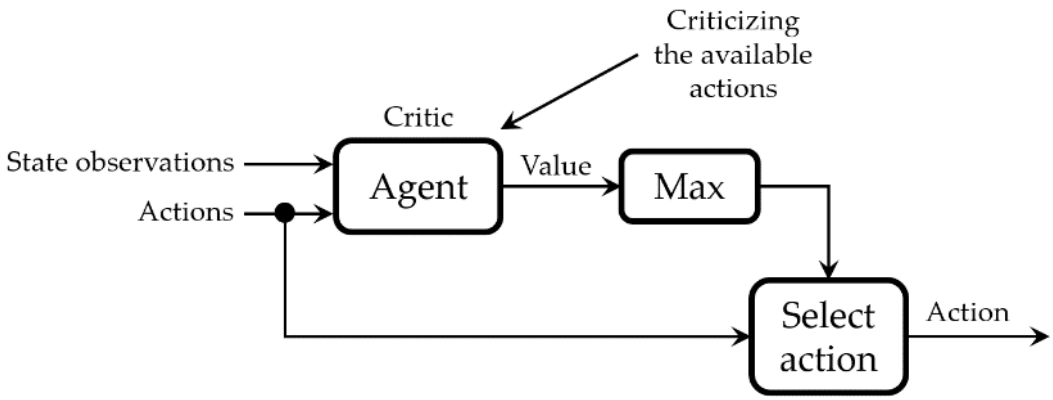


Figure 2.22 | Schematic representation of the value-based method.

A classic example is Q-learning in Grid World, where the agent updates a Q-table with state-action values through iterative learning, utilizing the Bellman equation to refine its strategy.

$$Q(S,a) = Q(S,a) + \alpha[R(S,a) + \gamma \cdot max \cdot Q'(S',a') - Q(S,a)] \tag{2.19}$$

Where:

- $R(S, a)$: Reward for taking action a from state S.
- $max \cdot Q'(S', a')$: Maximum expected value from state S.
- γ: Discount future reward.
- $Q(S, a)$: Previous estimate of value.
- α: Learning rate.

The value function can also be represented by a NN. The idea is similar to using a table: state observations and an action are input, and the NN returns the value of that state/action pair. The policy consists of choosing the action with the highest value. Over time, the network would gradually converge on a function that outputs the true value for every action throughout the continuous state space.

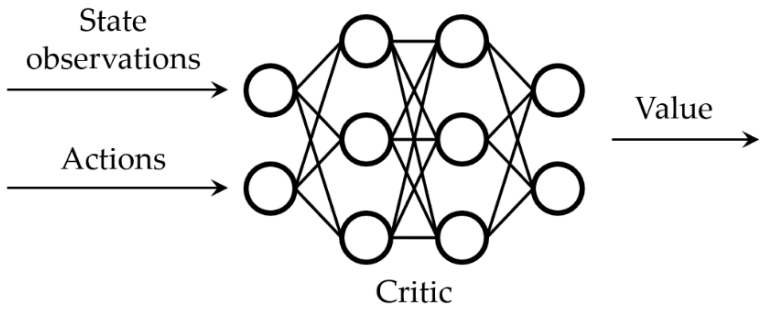


Figure 2.23 | Representation of the critic, policy-based function, by NN.

Value function–based policies are not effective for continuous action spaces. The reason is that it is impossible to compute the value for each action one at a time within an infinite action space to determine the maximum value. Even in a large, but finite, action space, this process becomes computationally prohibitive. This limitation is particularly problematic in control problems, where continuous action spaces are common, such as applying a continuous range of torques in an inverted pendulum problem.

Actor-critic methods merge these two approaches, employing an actor-network to propose actions and a critic network to evaluate them, making them suitable for continuous action spaces and environments with high reward variance. This approach is effective for continuous action spaces because the critic only needs to evaluate the single action taken by the actor, rather than attempting to find the optimal action by assessing all possible actions.

The actor selects an action in a manner similar to a policy function algorithm and applies it to the environment. The critic then predicts the value of that action for the current state-action pair. Using the reward from the environment, the critic assesses the accuracy of its value prediction. The error is calculated as the difference between the new estimated value of the previous state and the old value from the critic network. This new estimated value is derived from the received reward and the discounted value of the current state. The error indicates to the critic whether the outcome was better or worse than expected. The critic updates itself based on this error, improving its future predictions for the same state. The actor also adjusts itself based on the critic's feedback, modifying its probabilities of selecting that action in the future. In this manner, the policy improves by following the reward gradient as recommended by the critic, rather than relying directly on the rewards.

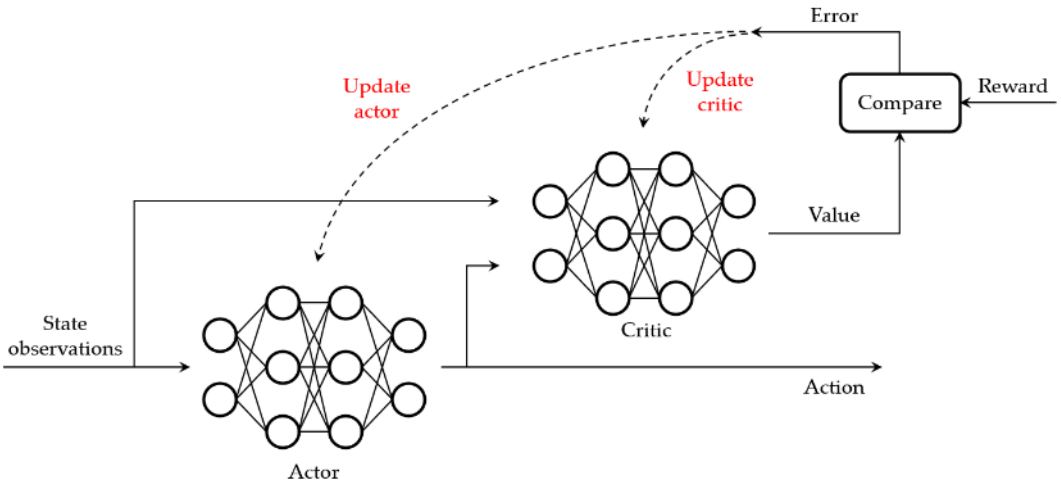


Figure 2.24 | Schematic representation of the actor-critic method.

This synergy accelerates learning and improves performance in complex environments, demonstrating why multiple NNs may be necessary for creating an effective RL agent.

2.4.3.2 | Deep Deterministic Policy Gradient (DDPG)

A DDPG agent is a type of actor-critic RL agent that aims to find an optimal policy maximizing the expected cumulative long-term reward. DDPG agents employ the following NN-based components:

- *Critic*: Q-value function critic $Q(S, A)$.
- *Actor*: Deterministic policy actor $\pi(S)$.

During the training process, a DDPG agent:

- Updates the properties of the actor and critic at each time step during the learning.
- Maintains a circular experience buffer to store past experiences. The agent updates the actor and critic by using a mini-batch of experiences randomly sampled from this buffer.
- Introduces perturbations to the action chosen by the policy using a stochastic noise model at each training step.

The addition of noise during learning improves the generalization of the model and makes it more robust [58]. There are three main reasons for its use:

- *Exploration*: During learning, an agent must make decisions to maximize a reward. The addition of noise allows the agent to explore new actions and strategies that might not have been considered otherwise.
- *Regularisation*: The addition of noise can act as a form of regularisation, helping to prevent overfitting. A model that learns to handle variability in its environment is more likely to generalize well to new data rather than overfitting to training data.
- *Numerical stability*: In some situations, adding noise can help maintain numerical stability during optimization. For example, it may prevent the model from getting tripped up by local minima.

The noise used is of the Ornstein-Uhlenbeck (OU) type [59]. OU noise is characterized by significant temporal correlation, which means that its perturbations over time are correlated. This can be advantageous in situations where it is important to maintain temporal consistency in the agent's actions. For example, in continuous control problems, OU noise can help to produce smoother actions over time. Unlike Gaussian noise, OU noise can be used to guide exploration in a more focused manner. Temporal correlation can help the agent explore more coherent sequences of actions rather than simply moving randomly through the action space.

The noise value $v(k)$ is updated at each sample time step k using the following equation:

$$v(k+1) = v(k) + \hat{\mu} \cdot (\mu - v(k)) \cdot T_S + \sigma(k) \cdot N \cdot \sqrt{T_S} \qquad (2.20)$$

The standard deviation decays at each sample time step according to the following equations:

$$\tilde{\sigma} = \sigma(k) \cdot (1 - d\sigma)$$
$$\sigma(k+1) = \max(\tilde{\sigma}, \sigma_{min}) \tag{2.21}$$

Where:

- μ: Mean noise value.
- $\hat{\mu}$: Constant specifying how quickly the noise model output is attracted to the mean.
- σ: Initial value of noise standard deviation.
- $d\sigma$: Decay rate of the standard deviation.
- $\tilde{\sigma}$: Decayed standard deviation.
- σ_{min}: Minimum value of σ.
- N: Random number drawn from a normal Gaussian distribution.
- T_S: Agent sample time.
- $v(1)$: Initial noise.

A DDPG agent keeps track of four function approximators to estimate the policy and value function:

- Actor $\pi(S;\theta)$: Using parameters θ, the actor receives observation S and outputs the action that maximizes the long-term reward.
- *Target actor* $\pi_t(S;\theta_t)$: The agent continuously modifies the target actor parameters θ_t using the most recent actor parameter values to increase the stability of the optimization.
- *Critic* $Q(S,A;\phi)$: Given observation S and action A as inputs, the critic, with parameters ϕ, yields the associated expectation of the long-term reward.

- *Target critic* $Q_t(S, A; \phi_t)$: The agent continuously modifies the target critic parameters ϕ_t using the most recent values of the critic parameters to increase the stability of the optimization.

The structure and parameterization of $Q(S, A; \phi)$ and $Q_t(S, A; \phi_t)$ are the same, as are the structures and parameterization of $\pi(S; \theta)$ and $\pi_t(S; \phi_t)$.

DDPG agent updates its actor and critic models at each time step using the following training procedure:

- Set random parameter values ϕ for the critic $Q(S, A; \phi)$ and set the same values for the target critic parameters ϕ_t, so that $\phi_t = \phi$.
- Set random parameter values θ for the actor $\pi(S; \theta)$, and set the same values for the target actor parameters θ_t, so that $\theta_t = \theta$.
- For each training time step:
    1. Choose action $A = \pi(S; \theta) + N$ for the current observation S, where N is the noise model.
    2. Apply action A. Note the reward R and subsequent observation S'.
    3. Store the experience (S, A, R, S') in the experience buffer.
    4. Draw a random mini-batch of M experiences (S_i, A_i, R_i, S'_i) from the experience buffer.
    5. Change the value function target y_i to R_i if S' is a terminal condition. If not, assign

$$y_i = R_i + \gamma Q_t(S'_i, \pi_t(S'_i; \theta_t); \phi_t) \tag{2.22}$$

The experience reward R_i plus the discounted future reward composes the value function target.

The agent first calculates the next action by providing the next observation S'_i from the sampled experience to the target

actor. Then the agent calculates the cumulative reward by providing the next action to the target critic.

6. Update the critic parameters by minimizing the loss L across all sampled experiences M.

$$L = \frac{1}{2M}\sum_{i=1}^{M}(y_i - Q(S_i, A_i; \phi))^2 \tag{2.23}$$

7. To maximize the expected discounted reward, update the actor parameters according to the sampled policy gradient.

$$\begin{gathered}\nabla_\theta J \approx \frac{1}{M}\sum_{i=1}^{M} G_{ai}\, G_{\pi i} \\ G_{ai} = \nabla_A Q(S', A; \phi) \text{ where } A = \pi(S_i; \theta) \\ G_{\pi i} = \nabla_\theta \pi(S_i; \theta)\end{gathered} \tag{2.24}$$

G_{ai} is the gradient of the critic output concerning the action computed by the actor-network, and $G_{\pi i}$ is the gradient of the actor output concerning the actor parameters. Both gradients are calculated for observation S_i.

8. Update the target actor and critic parameters using smoothing factor $\tau(< 1)$.

$$\begin{aligned}\phi_t &= \tau\phi + (1-\tau)\phi_t \text{(critic parameters)} \\ \theta_t &= \tau\theta + (1-\tau)\theta_t \text{ (actor parameters)}\end{aligned} \tag{2.25}$$

2.4.4 | Deployment

The final step in the reinforcement learning workflow is the deployment of the policy. If the learning process occurs with the physical agent in the real environment, the learned policy can be directly applied and exploited. However, in scenarios where the majority of learning is conducted offline through interaction with a simulated environment, the policy, once sufficiently optimized, can be transferred to physical hardware. This static

policy can then be deployed across multiple targets, akin to traditional control laws. Nonetheless, continuous learning with the real hardware may still be necessary after deployment. This is because real environments may be difficult to model precisely, leading to discrepancies between the model's optimal policy and the real-world performance. Additionally, as environments evolve over time, ongoing learning allows the agent to adapt to these changes. Therefore, both the static policy and the learning algorithms are deployed to the target. This setup enables the choice between executing the static policy with learning turned off or continuing to refine the policy with learning enabled. This approach leverages the complementary nature of simulated and real-world learning, where simulation provides a safe and efficient means to achieve a near-optimal policy, and online learning with physical hardware fine-tunes the policy to ensure its effectiveness in the actual environment.

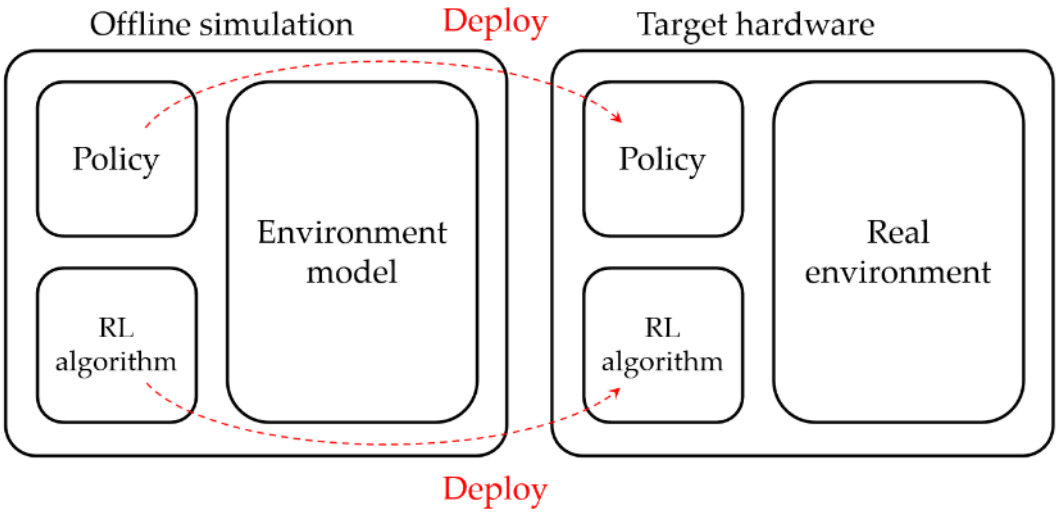


Figure 2.25 | Deployment of policy and RL algorithm in the target hardware.

2.4.5 | Strengths and Weaknesses

RL has emerged as a promising tool for autonomous vehicle control, capable of learning complex strategies from interaction with the

environment. Its ability to handle high-dimensional and dynamic systems makes it a strong candidate for advanced control tasks. However, RL's training process is data- and time-intensive, and the resulting policies may lack interpretability, raising concerns for safety-critical systems. RL provides adaptability and the potential to handle intricate control problems, but its limitations in interpretability and training efficiency pose significant challenges. Combining RL with model-based approaches can provide a structured framework while retaining its adaptability.

3 | OVERVIEW OF KEY ROAD VEHICLE DYNAMICS TOPICS

This section provides a comprehensive look at essential concepts in vehicle dynamics, focusing on four critical aspects. First, it examines tire-road interaction forces (3.1), crucial for understanding vehicle behavior under various driving conditions. Then, it discusses the sideslip angle (3.2), which affects vehicle stability and control. The third subsection covers unsprung masses (3.3), which influence ride comfort and handling. Lastly, the section addresses roll and pitch (3.4), key factors in determining vehicle balance and dynamic response during maneuvers.

3.1 | Tire-road interaction forces

Tire-road interaction forces are crucial components in vehicle dynamics, significantly affecting traction, handling, and safety. These forces are generated at the contact patch between the tire and the road surface, and they determine how effectively a vehicle can accelerate, decelerate, and navigate turns. Understanding and optimizing tire-road interaction forces is essential for enhancing vehicle performance and ensuring safe driving conditions.

3.1.1 | Definition and types of tire-road interaction forces

Tire-road interaction forces refer to the forces exerted by the tire on the road surface and vice versa. These forces are responsible for transmitting the vehicle's power to the road, enabling movement and control. The

primary forces involved in tire-road interaction include longitudinal, lateral, and vertical forces.

- *Longitudinal forces*: These forces are parallel to the direction of travel and are responsible for acceleration and braking. Longitudinal forces arise from the friction between the tire and the road surface as the vehicle speeds up or slows down.
- *Lateral forces*: These forces act perpendicular to the direction of travel and are crucial for cornering. Lateral forces are generated when the vehicle changes direction, allowing it to navigate turns and maintain stability.
- *Vertical forces*: Also known as normal forces, these are the forces acting perpendicular to the road surface. Vertical forces are primarily due to the vehicle's weight and are essential for maintaining tire contact with the road.

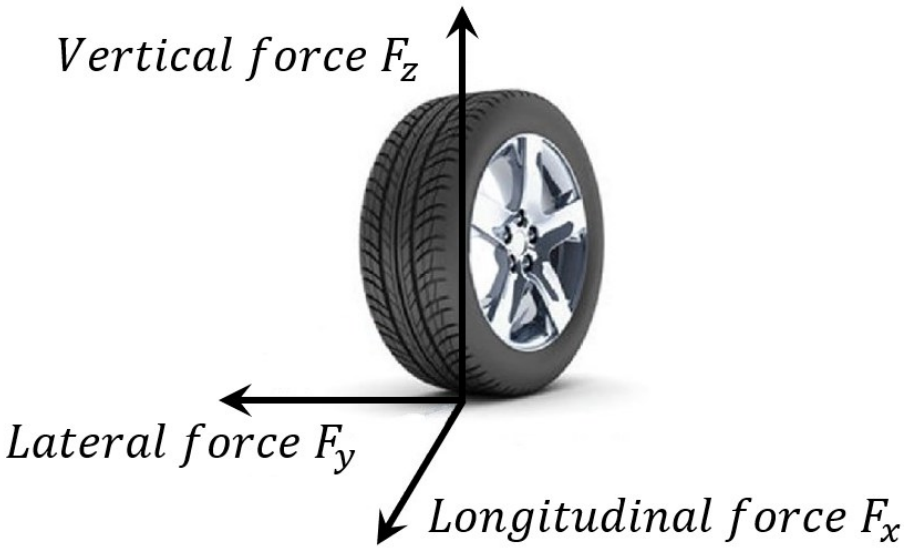


Figure 3.1 | Schematic representation of the tire-road interaction forces.

3.1.2 | Measurement of tire-road interaction forces

Measuring tire-road interaction forces directly involves sophisticated equipment and techniques. Common methods include:

- *Load cells*: These devices measure the forces transmitted through the tire by converting mechanical force into electrical signals. Load cells are often integrated into the vehicle's suspension system.
- *Force transducers*: Mounted on the tire or wheel assembly, force transducers measure the forces acting on the tire in multiple directions, providing detailed data on tire-road interactions.
- *Tire pressure sensors*: These sensors measure changes in tire pressure caused by deformation and contact with the road, indirectly providing information on interaction forces.

3.1.3 | Impact on traction and stability

Tire-road interaction forces are vital for maintaining traction and stability. Adequate longitudinal forces ensure efficient acceleration and braking, while sufficient lateral forces enable safe cornering. Vertical forces keep the tires in contact with the road, preventing loss of control.

Handling performance is directly influenced by tire-road interaction forces. Properly managed forces allow for precise steering response and control, enhancing the driver's ability to navigate various driving conditions. In high-performance vehicles, optimizing these forces is critical for achieving superior handling characteristics.

3.1.4 | Tire design and material

The design and material of tires significantly impact tire-road interaction forces. Key factors include:

- *Tread pattern*: The tread pattern affects how the tire interacts with the road surface, influencing traction, water dispersion, and noise generation.

- *Rubber compound*: The composition of the rubber affects the tire's grip and wear characteristics. Harder compounds last longer but offer less grip, while softer compounds wear out more quickly but offer higher traction.
- *Tire structure*: The internal structure, including the belt and carcass materials, determines the tire's stiffness and ability to deform under load, affecting interaction forces.

3.1.5 | Case studies and practical applications

- *Integration with Advanced Driver Assistance Systems (ADAS)*: Integrating tire-road interaction data with ADAS can enhance vehicle performance and safety. ADAS systems, such as anti-lock braking systems (ABS) and electronic stability control (ESC), can benefit from accurate data on interaction forces to optimize their functions.
- *Autonomous vehicles*: As autonomous vehicles become more prevalent, managing tire-road interaction forces will be crucial for ensuring smooth and safe operation. Advanced sensors and control algorithms will be essential for maintaining optimal traction and stability in a wide range of driving conditions.
- *Motorsport applications*: In motorsport, optimizing tire-road interaction forces is critical for maximizing performance. Advanced tire technologies and precise tuning allow racing teams to maintain optimal traction and grip, enhancing speed and handling. Case studies of successful racing teams highlight the importance of managing tire-road interaction forces in competitive settings.
- *Commercial vehicles*: For commercial vehicles, such as trucks and buses, managing tire-road interaction forces improves both safety

and efficiency. By optimizing these forces, commercial vehicles can offer better fuel economy, reduce tire wear, and enhance overall performance.

- *Off-road vehicles*: Off-road vehicles face extreme conditions that challenge tire-road interaction. Advanced tire designs and technologies ensure that off-road vehicles maintain traction and stability on rough terrain. Practical applications in off-road environments demonstrate the effectiveness of these technologies in managing tire-road interaction forces.

3.2 | Sideslip angle

The sideslip angle, often referred to as β (beta), is a fundamental parameter in vehicle dynamics that significantly influences the handling and stability of a vehicle. It is defined as the angle between the vehicle's longitudinal axis and the direction of travel of its center of mass. Understanding and controlling the sideslip angle is crucial for ensuring safe and efficient vehicle operation, particularly in high-performance and autonomous driving contexts.

3.2.1 | Definition and significance of the sideslip angle

Mathematically, it can be expressed as:

$$\beta = \tan^{-1}\left(\frac{v_y}{v_x}\right) \tag{3.1}$$

where v_y is the lateral velocity and v_x is the longitudinal velocity of the vehicle. A non-zero sideslip angle indicates that the vehicle is not aligned with its trajectory, which is particularly important during cornering or when the vehicle is subjected to lateral forces.

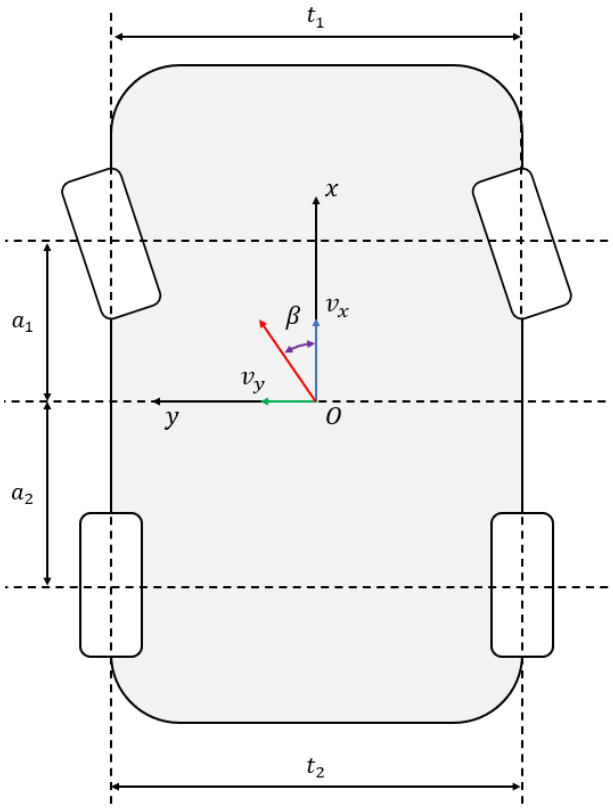


Figure 3.2 | Schematic representation of the sideslip angle of a vehicle.

The sideslip angle is a critical parameter for several reasons:

- *Vehicle stability*: Excessive sideslip angles can lead to instability and potential loss of control. For instance, a high sideslip angle can precede phenomena such as understeer or oversteer, where the vehicle fails to follow the intended path.

- *Tire wear*: High sideslip angles can result in increased tire wear due to the greater lateral forces acting on the tires. Efficient management of the sideslip angle can thus enhance tire lifespan.

- *Performance*: In motorsport, minimizing and accurately controlling the sideslip angle can lead to better handling and faster lap times. Precision in this aspect allows drivers to maintain optimal traction and speed through corners.

- *Safety*: For everyday vehicles, controlling the sideslip angle contributes to safer driving, particularly in adverse conditions such as wet or icy roads.

3.2.2 | Measurement of sideslip angle

Direct measurement of the sideslip angle is challenging due to the complexity of accurately determining the direction of travel of the vehicle's center of mass. Advanced technologies used for this purpose include:

- *GPS systems*: High-precision GPS can provide accurate measurements of a vehicle's trajectory, which can be used to infer the sideslip angle.
- *Inertial Measurement Units (IMUs)*: These devices measure acceleration and angular velocity, which can be processed to estimate the vehicle's sideslip angle.
- *Optical sensors*: Cameras and other optical devices can track the motion of the vehicle relative to the road to provide an estimate of the sideslip angle.

3.2.3 | Role of sideslip angle in vehicle dynamics

The sideslip angle has a profound effect on vehicle handling. It influences how the vehicle responds to steering inputs and how it maintains its path during maneuvers. A well-controlled sideslip angle ensures that the vehicle can follow the driver's intended path with minimal deviation, thus enhancing overall handling and stability.

To manage the sideslip angle effectively, modern vehicles employ various control strategies:

- *Electronic Stability Control (ESC)*: This system automatically adjusts brake pressure on individual wheels and modulates engine power to correct deviations in the sideslip angle, thereby preventing skidding and maintaining stability.

- *Active steering systems*: These systems adjust the steering angle automatically to compensate for changes in the sideslip angle, providing more precise control and improving cornering performance.

- *Torque vectoring*: By varying the distribution of power between the wheels, torque vectoring systems can influence the sideslip angle to enhance cornering grip and stability.

For autonomous vehicles, precise control of the sideslip angle is essential for reliable and safe operation. Autonomous driving algorithms must constantly monitor and adjust the sideslip angle to ensure the vehicle remains on its intended path, especially during complex maneuvers or when reacting to unexpected obstacles.

3.3 | Unsprung masses

The vertical displacement of unsprung masses is a critical aspect of vehicle dynamics, influencing both ride comfort and handling performance. Unsprung masses include components such as wheels, tires, brakes, and parts of the suspension system that are not supported by the vehicle's springs. Understanding and managing the vertical displacement of these masses is essential for optimizing vehicle performance, ensuring passenger comfort, and maintaining stability.

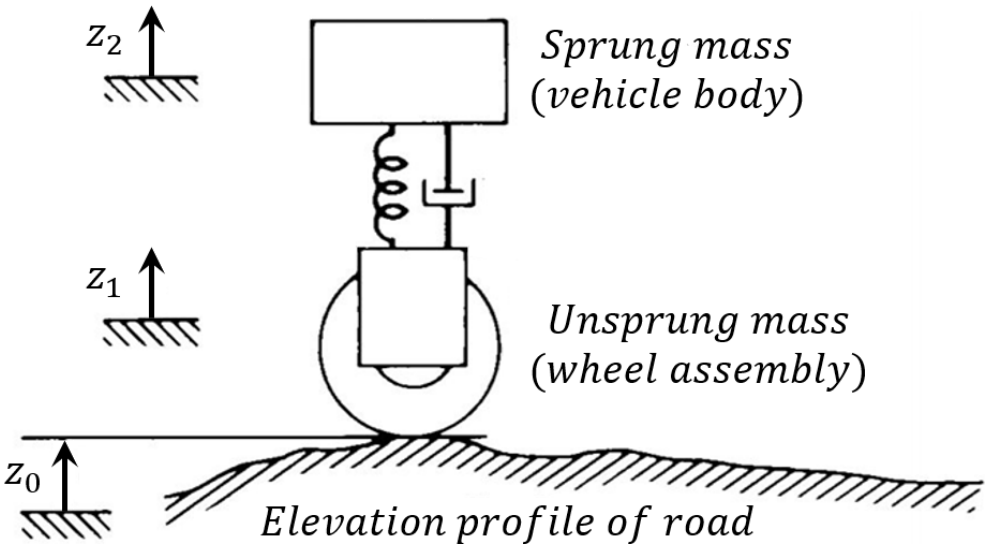


Figure 3.3 | Schematic representation of sprung and unsprung masses

3.3.1 | Definition and significance of unsprung mass vertical displacement

Vertical displacement of unsprung masses refers to the upward and downward movement of these components. This movement is primarily induced by road irregularities, such as bumps and potholes, which cause the wheels and other unsprung components to move vertically.

The vertical displacement of unsprung masses is a key factor in several areas:

- *Ride comfort*: Excessive vertical movement can transmit harsh vibrations and shocks to the vehicle's body, reducing passenger comfort. Effective management of this displacement is crucial for a smooth and comfortable ride.
- *Handling performance*: The behavior of unsprung masses affects the tire-road contact, which is essential for traction and stability. Proper control of vertical displacement ensures optimal tire grip and improves handling performance.
- *Vehicle safety*: Stability during braking, cornering, and acceleration can be compromised by uncontrolled vertical displacement of

unsprung masses. Ensuring minimal and controlled movement enhances overall vehicle safety.

- *Tire wear*: Excessive vertical displacement can lead to uneven tire wear, reducing tire lifespan and increasing maintenance costs.

3.3.2 | Measurement of vertical displacement

Direct measurement of vertical displacement involves using sensors and instruments to track the movement of unsprung masses. Techniques include:

- *Accelerometers*: Mounted on wheels and other unsprung components, accelerometers measure the acceleration due to vertical movements, which can be integrated to determine displacement.
- *Laser displacement sensors*: These sensors measure the distance between the unsprung components and a reference point on the vehicle body, providing precise displacement data.
- *Strain gauges*: Installed on suspension components, strain gauges measure the deformation caused by vertical loads, which can be used to estimate displacement.

3.3.3 | Role of vertical displacement in vehicle dynamics

Vertical displacement of unsprung masses directly affects the comfort of passengers. When a vehicle encounters a road irregularity, the unsprung masses move vertically, generating vibrations and shocks. Effective suspension design aims to minimize these impacts, isolating the vehicle body from road disturbances and enhancing ride comfort.

The ability of a vehicle to maintain traction and stability during maneuvers is heavily influenced by the vertical displacement of its unsprung masses. Excessive vertical movement can cause the tires to lose

contact with the road, reducing grip and compromising handling. Suspension systems are designed to control this displacement, ensuring that the tires remain in optimal contact with the road surface.

Modern vehicles incorporate advanced technologies to better manage vertical displacement:

- *Active suspension systems*: These systems use actuators to adjust the suspension characteristics in real-time, providing optimal control of vertical displacement under varying conditions.
- *Adaptive damping*: This technology adjusts the damping force based on road conditions and driving style, enhancing both ride comfort and handling.
- *Air suspension*: Air springs can be adjusted for different loads and driving conditions, providing a versatile solution for controlling vertical displacement.

3.3.4 | Suspension system design

The design of a vehicle's suspension system plays a crucial role in managing the vertical displacement of unsprung masses. Key components and considerations include:

- *Springs*: The stiffness of the springs affects how much the unsprung masses move in response to road irregularities. Softer springs provide better ride comfort by allowing more movement, while stiffer springs improve handling by limiting movement.
- *Dampers*: Also known as shock absorbers, dampers control the rate of vertical displacement by dissipating the energy from the movement. Proper damping is essential for balancing ride comfort and handling performance.

- *Anti-roll bars*: These components reduce body roll during cornering, influencing the vertical displacement of unsprung masses and improving stability.

3.4 | Roll and pitch

Roll and pitch are fundamental quantities of vehicle dynamics that significantly impact handling, stability, and passenger comfort. Roll refers to the tilting motion of a vehicle from side to side, while pitch describes the tilting motion of a vehicle from front to back. Understanding and controlling roll and pitch are essential for optimizing vehicle performance and ensuring safe and comfortable driving experiences.

3.4.1 | Definition and significance of roll and pitch

Roll is the rotational movement of a vehicle around its longitudinal axis, which runs from the front to the back of the vehicle. It is typically induced by lateral forces, such as those encountered during cornering. The roll angle, denoted as φ (phi), describes the extent of this rotation.

Pitch is the rotational movement of a vehicle around its lateral axis, which runs from one side of the vehicle to the other. It is primarily influenced by longitudinal forces, such as acceleration and braking. The pitch angle, denoted as ϑ (theta), indicates the degree of this rotation.

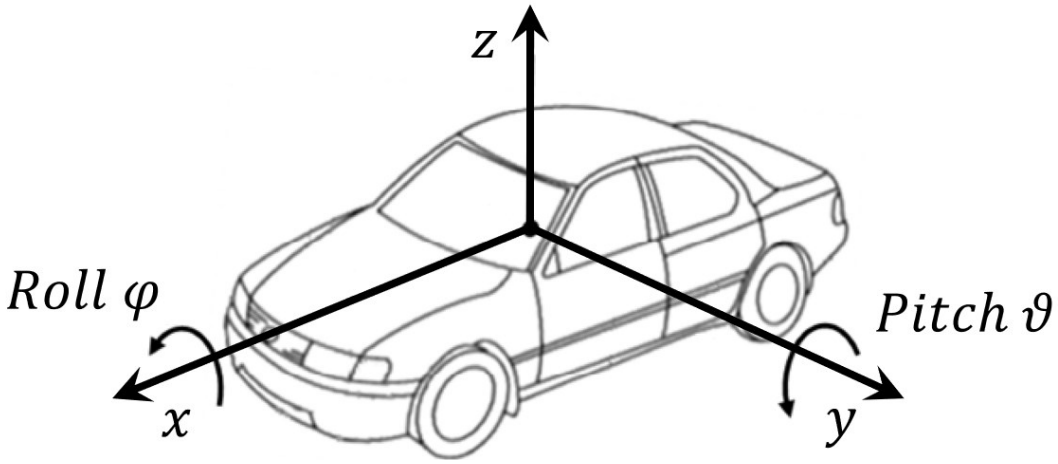


Figure 3.4 | Schematic representation of roll and pitch angles of a vehicle.

3.4.2 | Measurement of roll and pitch

Direct measurement of roll and pitch involves using sensors and instruments to monitor the rotational movements of the vehicle. Techniques include:

- *Inertial Measurement Units (IMUs)*: IMUs combine accelerometers and gyroscopes to measure angular velocities and accelerations, providing accurate data on roll and pitch angles.
- *Gyroscopes*: These sensors measure the rate of rotation around the vehicle's axes, allowing for the calculation of roll and pitch angles.
- *Laser and optical sensors*: Mounted on the vehicle's body, these sensors measure the displacement and angle changes relative to a reference point, providing data on roll and pitch dynamics.

3.4.3 | Role of roll and pitch in vehicle dynamics

Roll and pitch significantly affect a vehicle's handling and stability. Properly managed roll dynamics ensure that the vehicle remains level and stable during cornering, preventing excessive tilting that can compromise tire grip and lead to skidding or rollover. Similarly, controlled pitch

dynamics ensure that the vehicle's weight distribution remains balanced during acceleration and braking, maintaining traction and stability.

3.4.4 | Influence on suspension design

The design of a vehicle's suspension system plays a crucial role in managing roll and pitch motions. Key components and considerations include:

- *Anti-roll bars*: These components, also known as sway bars, connect the left and right wheels and resist roll motion by distributing lateral forces more evenly across the vehicle. This reduces body roll during cornering and enhances stability.
- *Suspension geometry*: The arrangement of suspension components affects how the vehicle responds to roll and pitch. Optimized suspension geometry can minimize unwanted tilting motions and improve overall handling.
- *Dampers*: Shock absorbers control the rate of roll and pitch by dissipating energy from the suspension movement. Proper damping is essential for balancing ride comfort and handling performance.
- *Springs*: The stiffness of the springs influences the vehicle's resistance to roll and pitch. Stiffer springs reduce roll and pitch angles but may compromise ride comfort, while softer springs provide better comfort but can lead to increased tilting.

4 | APPLICATIONS IN THE PROPOSED VEHICLE DYNAMICS TOPICS

For each of the vehicle dynamics topics proposed in Chapter 3, the techniques presented in Chapters 1 and 2 were applied. The proposed solutions are based on physical models, AI, or a combination of the two approaches.

In the following, each application is indicated:

- *Tire-road interaction forces*:
    - *AI-based estimation (Section 4.1)*: A method for predicting tire-road interaction forces using only longitudinal and lateral acceleration measurements, bypassing the need for expensive sensors, was proposed. By employing a multi-output NN, the study accurately predicts the interaction forces on the rear wheels, with experimental validation highlighting its effectiveness and offering valuable insights for vehicle dynamics and control systems [60].
    - *Model and AI combined estimation (Sections 4.2 and 4.3)*: Two solutions have been proposed. The first consists of a technique for estimating tire-road interaction forces using a Central Difference Kalman filter applied to a Double Track Model. The approach integrates Pacejka's magic formulas and NNs to account for nonlinearities, combined slips, camber angle, load transfers, and aerodynamic forces, using measurements like longitudinal velocity, yaw rate, and steering angles [61]. The second introduces an

enhanced method for estimating tire-road interaction forces under combined slip conditions by combining Pacejka's formula with NNs. The NNs, trained on various driving scenarios, effectively correct the estimation errors of Pacejka's formula, significantly improving traction force estimation, especially in non-pure slip conditions like curves [62].

- *Sideslip angle*:
  - *AI-based estimation (Section 4.3)*: Two NNs to estimate the sideslip angle, a critical factor in vehicle stability and safety. Using inputs like lateral acceleration, steering wheel angle, and yaw rate, the networks accurately estimate the sideslip angle without the need for expensive sensors [63].
- *Unsprung masses*:
  - *Model-based estimation (Section 4.5)*: A virtual sensor to measure the vertical displacement of a vehicle's unsprung masses was designed. It employes a 7-degree-of-freedom vehicle model with inputs from longitudinal and lateral accelerations.
  - *AI-based estimation (Section 4.6)*: A virtual sensor for estimating the vertical displacement of a vehicle's wheels has been presented. Using a multi-output NN with inputs from an IMU and active suspension data, the virtual sensor accurately predicts displacements for all four wheels [64].
  - *Model and AI combined estimation (Section 4.7)*: A multi-output NN with a model-based approach to improve the accuracy of wheel displacement estimation for vehicle

suspension systems has been presented. By compensating for errors in traditional models using only vertical acceleration measurements, the methodology significantly enhances estimation accuracy [65].

- *Roll and pitch*:
    - *AI-based estimation (Section 4.8)*: A multi-output estimator that uses longitudinal and lateral accelerations to estimate a vehicle's roll and pitch angles simultaneously [66].
    - *AI-based control (Section 4.9)*: A hybrid control strategy for active anti-roll bars, combining a proportional controller with a Reinforcement Learning (RL) based controller to reduce vehicle body roll during cornering and on rough terrain was introduced.

4.1 | Multi-output physically analyzed neural network for the prediction of tire–road interaction forces

This study presents a novel method for predicting tire-road interaction forces using only longitudinal and lateral acceleration measurements. Given the high cost and difficulty of implementing sensors that directly measure these forces in a vehicle, this approach addresses a critical need by utilizing commonly available sensor data. The study employs a multi-output NN architecture to predict the longitudinal, lateral, and vertical forces exerted by the rear wheels, particularly those related to traction. Experimental validation confirms the effectiveness of this method in accurately forecasting tire-road interaction forces. Furthermore, an in-depth analysis of the input-output relationships sheds light on the complex dynamics of tire-road interactions. This research highlights the potential of

NN models to improve predictive capabilities in vehicle dynamics, offering valuable insights for various automotive engineering and control system applications.

To define the physical quantities for both input and output, it is essential to establish reference systems.

The vehicle reference system, SyV, is defined according to ISO 8855:2011 [67] as follows:

- *Longitudinal axis* (x): This axis runs from the rear to the front of the vehicle, aligned with the direction of movement.
- *Lateral axis* (y): This axis extends from the right side to the left side of the vehicle, perpendicular to the longitudinal axis.
- *Vertical axis* (z): This axis points from the bottom to the top of the vehicle, perpendicular to the plane on which the vehicle is moving.

The origin of the SyV reference system is the vehicle's center of gravity, denoted as point G (Figure 4.1).

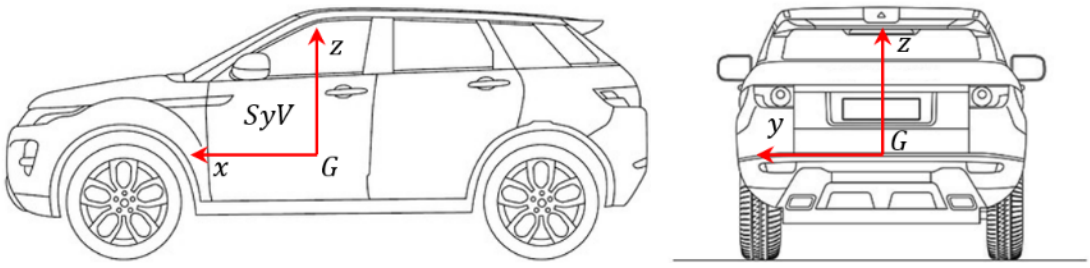


Figure 4.1 | SyV vehicle reference system.

The wheel reference system, SyW, is also defined according to ISO 8855:2011 [67] as follows:

- *Longitudinal axis* (x)**:** This axis runs from the rear to the front of the tire, aligned with its rolling direction when the vehicle is moving forward.

- *Lateral axis (y)*: This axis extends from the right side to the left side of the tire, parallel to the plane of the road.
- *Vertical axis (z)*: This axis points upward, perpendicular to the road surface. The z-axis is orthogonal to the plane formed by the x- and y-axes.

The origin of the SyW reference system is at point P, located at the tire–road contact patch (Figure 4.2).

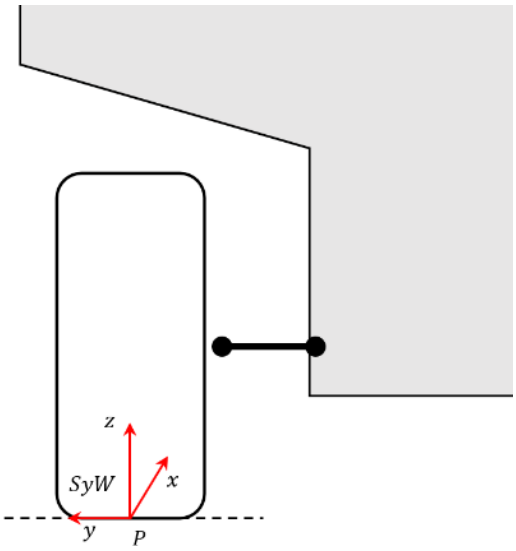


Figure 4.2 | ***SyW*** wheel reference system.

4.1.1 | Choice of inputs and outputs

Two quantities, defined in the SyV reference system, have been selected as inputs to the NNs, based on their physical significance:

- *Longitudinal acceleration (a_x)*: When a vehicle accelerates, the interaction forces between the tires and the road are crucial in transferring power to the wheels, generating the traction needed to propel the vehicle forward or applying braking forces to decelerate it. Therefore, the vehicle's longitudinal acceleration depends on the tires' ability to generate traction and braking forces on the road surface.

- *Lateral acceleration (a_y)*: When a vehicle turns or changes direction, lateral acceleration becomes significant. The interaction forces between the tires and the road provide the lateral traction necessary to keep the vehicle on its intended path.

The outputs of the NN, defined in the SyW reference system, include:

- *Longitudinal force ($F_{x_{RL}}$)*: Traction and braking force of the rear-left tire.
- *Lateral force ($F_{y_{RL}}$)*: Lateral steering force of the rear-left tire.
- *Vertical force ($F_{z_{RL}}$)*: Load force of the rear-left tire.
- *Longitudinal force ($F_{x_{RR}}$)*: Traction and braking force of the rear-right tire.
- *Lateral force ($F_{y_{RR}}$)*: Lateral steering force of the rear-right tire.
- *Vertical force ($F_{z_{RR}}$)*: Load force of the rear-right tire.

Including vertical forces as an output variable in the NN model is justified by their significant impact on vehicle performance and safety. Vertical forces on the tires, known as tire vertical loads, are crucial for various aspects of vehicle behavior, such as handling, stability, and ride comfort. These forces directly affect tire-road interaction, influencing traction, grip, and the vehicle's ability to maintain control during maneuvers like braking, cornering, and acceleration [68].

The vertical acceleration at the center of gravity, although measurable, was not used because it didn't provide additional information beyond what was already captured by the horizontal accelerations (a_x and a_y). The downward force on the wheels from the ground consists of a constant part—the vehicle's weight on each wheel—and a variable part caused by shifts in the vehicle's weight distribution during maneuvers. These shifts are already reflected in the horizontal accelerations, as the sensor is inertial.

Therefore, the horizontal acceleration data contains all the necessary information for the NN to estimate vertical contact forces. Additionally, vertical acceleration alone cannot indicate how vertical forces are distributed between the two traction tires. By excluding vertical acceleration as an input, the NN becomes more computationally efficient, needing to process only two inputs instead of three. Experimental results demonstrated that omitting vertical acceleration still allowed for accurate estimation of the vertical forces.

4.1.2 | Data collection

Data were collected as part of experimental tests in the context of the ITEAM project by Flanders Make. Table 4.1 provides details about the sensors used, including their locations, the quantities they measure, their measurement range, and the units of measurement.

Sensor	Location	Measured quantity	Measurement range	Measurement unit
IMU SBG IG500A [69]	Center of gravity of the vehicle	a_x, a_y	$\pm 5g$	$\left(\frac{m}{s^2}\right)$
Kistler RoaDyn s635 [70]	Rear wheels	$F_{x_{RL}}, F_{y_{RL}}, F_{z_{RL}}$, $F_{x_{RR}}, F_{y_{RR}}, F_{z_{RR}}$	$\pm 35\ kN\ (F_x, F_z)$ $\pm 20\ kN\ (F_x, F_z)$	(N)

Table 4.1 | Sensors used, locations, measured quantities, and measurement units.

The accelerometer is mounted at the vehicle's center of gravity (COG) using an aluminum bracket, which is lightweight and does not significantly impact the accuracy of the acceleration measurements due to its negligible weight relative to the vehicle. The IMU has a sampling rate of 100 Hz, while the force sensor samples at 125 Hz. To ensure consistency between the data, the force measurements used for training were resampled to 100 Hz, matching the lower of the two sampling rates. This means the NN, which

uses acceleration as input, can estimate forces at a rate of 100 Hz, providing new predicted force values every 0.01 s.

The NN model was trained offline using a system with an AMD Ryzen 5 4500U processor and 16 GB of DDR4-2666 dual-channel memory. The training process took approximately 3 hours and 56 minutes to reach convergence, highlighting the efficiency of modern ML algorithms and optimization techniques, even on modest computational hardware. This demonstrates the practical feasibility of the approach.

Using the same hardware setup, the NN was able to estimate all outputs for the experimental data (comprising 752909 instances) in 4873 s. This corresponds to producing all six outputs approximately every $6.5 \cdot 10^{-5}$ s, or at a frequency of around 15400 Hz, which is notably higher than the input rate of 100 Hz. This rapid output generation on a low-end computer indicates the potential for real-time application of the proposed solution.

Table 4.2 shows the distance a vehicle travels before a new force output is generated at various longitudinal speeds, calculated by multiplying the vehicle speed by the sample time.

Velocity $\left(\frac{km}{h}\right)$	Velocity $\left(\frac{m}{s}\right)$	Vehicle travel distance (m)
20	5.555	0.05555
40	11.11	0.1111
60	16.67	0.1667
80	22.22	0.2222
100	27.77	0.2777
120	33.33	0.3333

Table 4.2 | Vehicle travel distance at different velocities between two successive outputs.

No filters were applied to the measured acceleration and wheel forces. NNs inherently perform signal filtering during both training and data

processing. This is achieved through algebraic operations that connect inputs, including inputs from previous time steps, to outputs. This process effectively acts as a filter, reducing noise in the outputs. Additionally, NNs can detect dynamic relationships between inputs and outputs, even without an explicit physical model. Since noise in the input does not have a meaningful relationship with output noise, the NN naturally filters out noise during its operation, ensuring cleaner output signals.

Maneuvers conducted with a Range Rover Evoque vehicle (Figure 4.3) were utilized to generate the training, validation, and test datasets. Table 4.3 outlines the vehicle's specifications:

Symbol	Name	Value	Measurement unit
M	Sprung mass of the vehicle	2012.47	(kg)
m_{FL}	Front-left unsprung mass of the vehicle	54.47	(kg)
m_{FR}	Front-right unsprung mass of the vehicle	54.47	(kg)
m_{RL}	Rear-left unsprung mass of the vehicle	42.43	(kg)
m_{RR}	Rear- right unsprung mass of the vehicle	42.43	(kg)
h	Height of the CoG	0.631	(m)
l	Wheel base	2.662	(m)
a_1	Distance CoG to front axle	1.148	(m)
a_2	Distance CoG to rear axle	1.514	(m)
t_1	Front-wheel track	1.706	(m)
t_2	Rear-wheel track	1.712	(m)
J_{zx}	Products of inertia zx	-48.67	$(kg \cdot m^2)$
J_z	Moment of inertia with respect to z	4101	$(kg \cdot m^2)$

Table 4.3 | Vehicle characteristics.

Figure 4.3 | Experimental vehicle during maneuvers.

The maneuvers were performed at the Ford Lommel Proving Ground (Figure 4.4).

Figure 4.4 | Ford Lommel Proving Ground.

To capture the vehicle's behavior in different conditions, 88 maneuvers were performed, covering 12 distinct types:

Number	Name	Description
$1-14$	Constant radius	Radius ranging from 30 m to 100 m. Maneuver performed both clockwise and anticlockwise.
$15-19$	Step steer	Velocity of 80 $\frac{km}{h}$ and steering wheel angle ranging from 40° to 120°. Maneuver performed both clockwise and anticlockwise.
$20-23$	Brake in turn	Initial velocity of 80 $\frac{km}{h}$ and 100 $\frac{km}{h}$. Maneuver performed both clockwise and anticlockwise.
$24-29$	Sine with dwell	Velocity of 60 $\frac{km}{h}$ and steering wheel angle ranging from 40° to 80°.
$30-38$	Slalom	Equispaced cones of 17 and velocity ranging from 30 $\frac{km}{h}$ to 60 $\frac{km}{h}$.
$39-46$	Track 7, Inner durability road	Laps on track 7 with variable velocities up to 100 $\frac{km}{h}$.
$47-49$	Hill section	Variable slopes of up to $\pm$30%.
$50-54$	Gradients	Uphill slopes followed by downhill slopes. Gradients of $\pm$25%.
$55-60$	Track 5, High-speed oval	High-speed track laps with velocities of up to 120 $\frac{km}{h}$.
$61-70$	Acceleration test	Accelerations with traction torques ranging from 1500 $\frac{Nm}{s}$ to 4000 $\frac{Nm}{s}$.
$71-78$	Brake test	Braking with pedal travel from 20% to 60%.
$79-88$	ISO double lane change	Velocities ranging from 30 $\frac{km}{h}$ to 55 $\frac{km}{h}$.

Table 4.4 | List of the performed maneuvers with description.

A total of 752909 multi-input-multi-output pairs were gathered for the training and validation datasets, representing 2 hours and 5 minutes of driving time.

Figure 4.5 shows the distribution of samples collected for the training and validation datasets based on longitudinal acceleration a_x and lateral acceleration a_y.

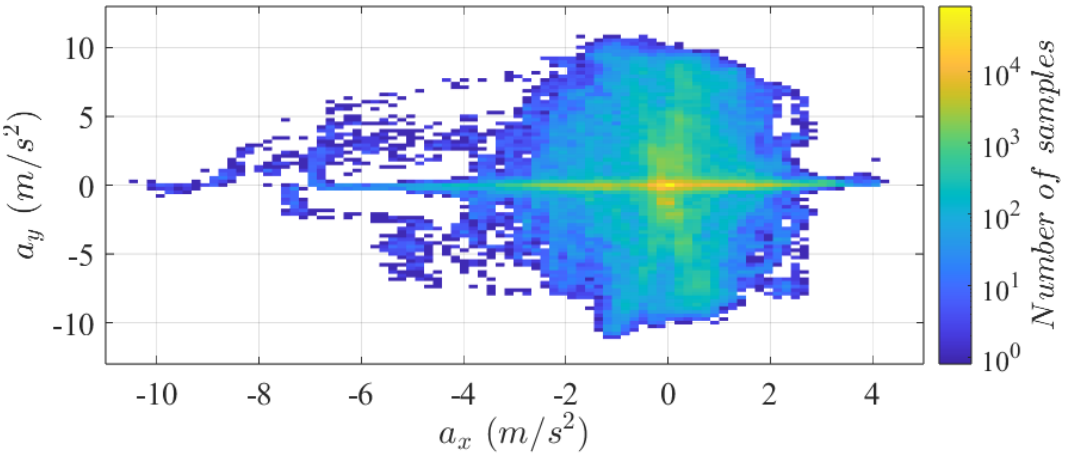


Figure 4.5 | Distribution of acquired samples with respect to inputs $\boldsymbol{a_x}$ and $\boldsymbol{a_y}$.

Figures 4.6 and 4.7 illustrate the distribution of samples collected for the training and validation datasets in relation to the input accelerations and the corresponding output forces.

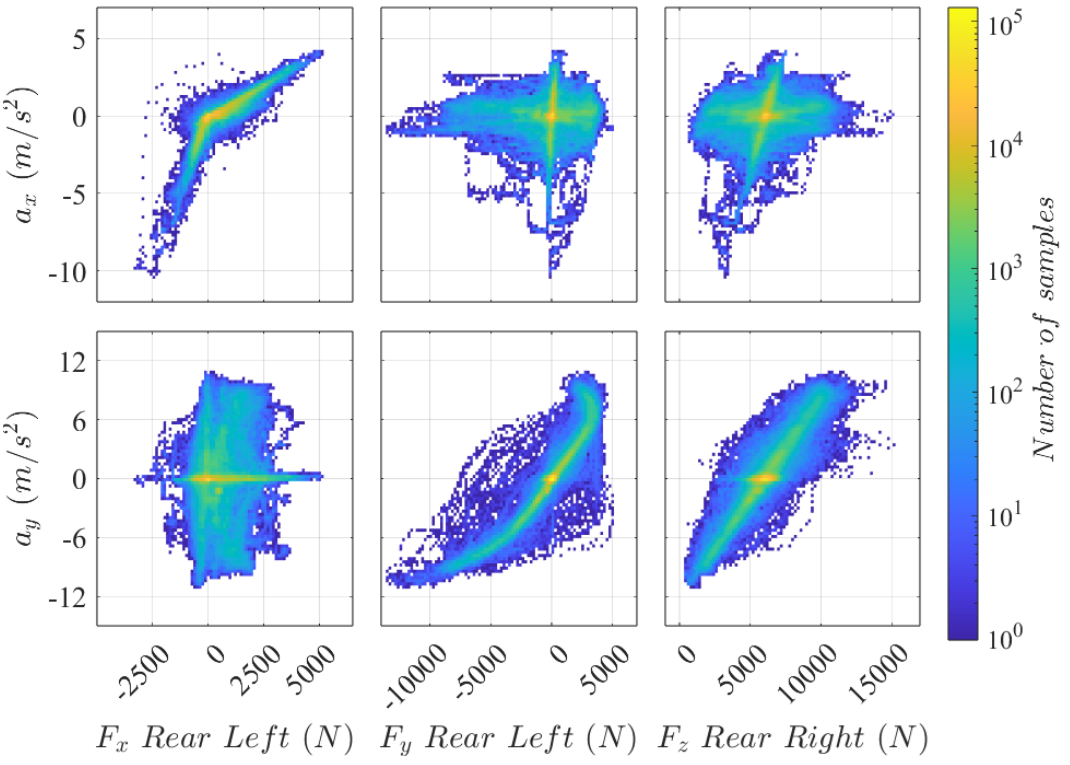


Figure 4.6 | Distribution of acquired sample with respect to accelerations and forces exchanged with the road by the rear-left tire.

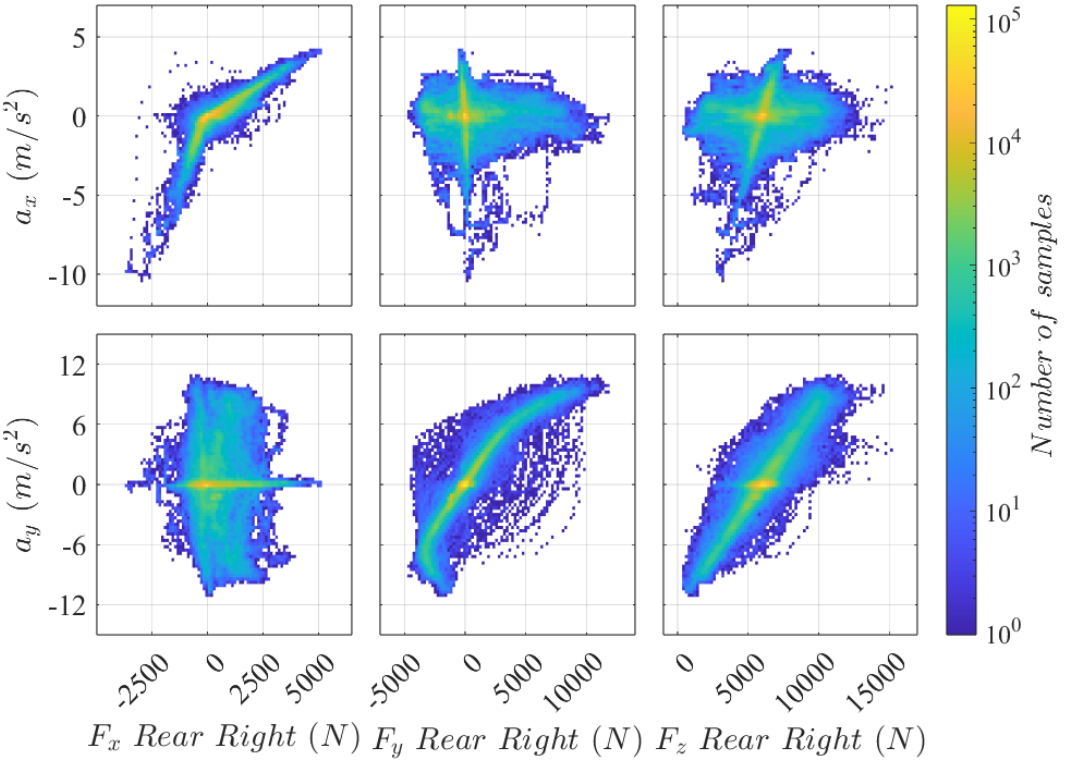


Figure 4.7 | Distribution of acquired samples with respect to accelerations and forces exchanged with the road by the rear-right tire.

It is clear that the relationship between the two inputs, a_x and a_y, and the six outputs—$F_{x_{RL}}$, $F_{y_{RL}}$, $F_{z_{RL}}$, $F_{x_{RR}}$, $F_{y_{RR}}$ and $F_{z_{RR}}$— is nonlinear and complex, making it difficult to describe with simple mathematical equations.

The maneuvers performed to test the NN, along with the corresponding results, are discussed in detail in a specific section.

4.1.3 | Neural network development

The NN utilized in this study has the following structure:

- Input layer.
- First hidden layer with a ReLU activation function.
- Second hidden layer with a ReLU activation function.
- Output layer.

There are several methods for determining the number of hidden layers in a NN [71] [72]. However, no single method can universally determine the optimal number of layers for every NN. Since most NNs in the literature employ two hidden layers, this approach was also adopted for the present study. Figure 4.8 illustrates the NN's structure.

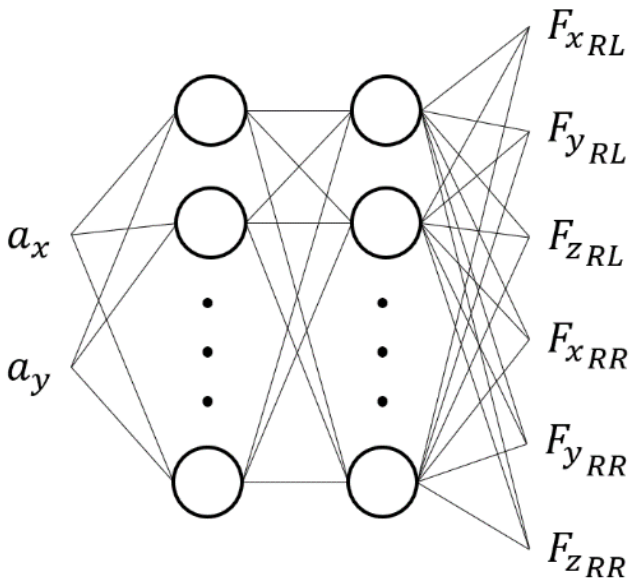


Figure 4.8 | NN architecture with evidence of inputs $(\boldsymbol{a_x}, \boldsymbol{a_y})$ and outputs $(\boldsymbol{F_{x_{RL}}}, \boldsymbol{F_{y_{RL}}}, \boldsymbol{F_{z_{RL}}}, \boldsymbol{F_{x_{RR}}}, \boldsymbol{F_{y_{RR}}}, \boldsymbol{F_{z_{RR}}})$.

To enhance convergence to the optimal set of model parameters, the data collected for the training and validation datasets—used as inputs for the NN—were normalized to have a zero mean and a unit standard deviation [73].

Figure 4.9 displays histograms of the non-normalized inputs, clearly showing the differences in standard deviations. Table 4.5 provides the mean and standard deviation of the non-normalized inputs.

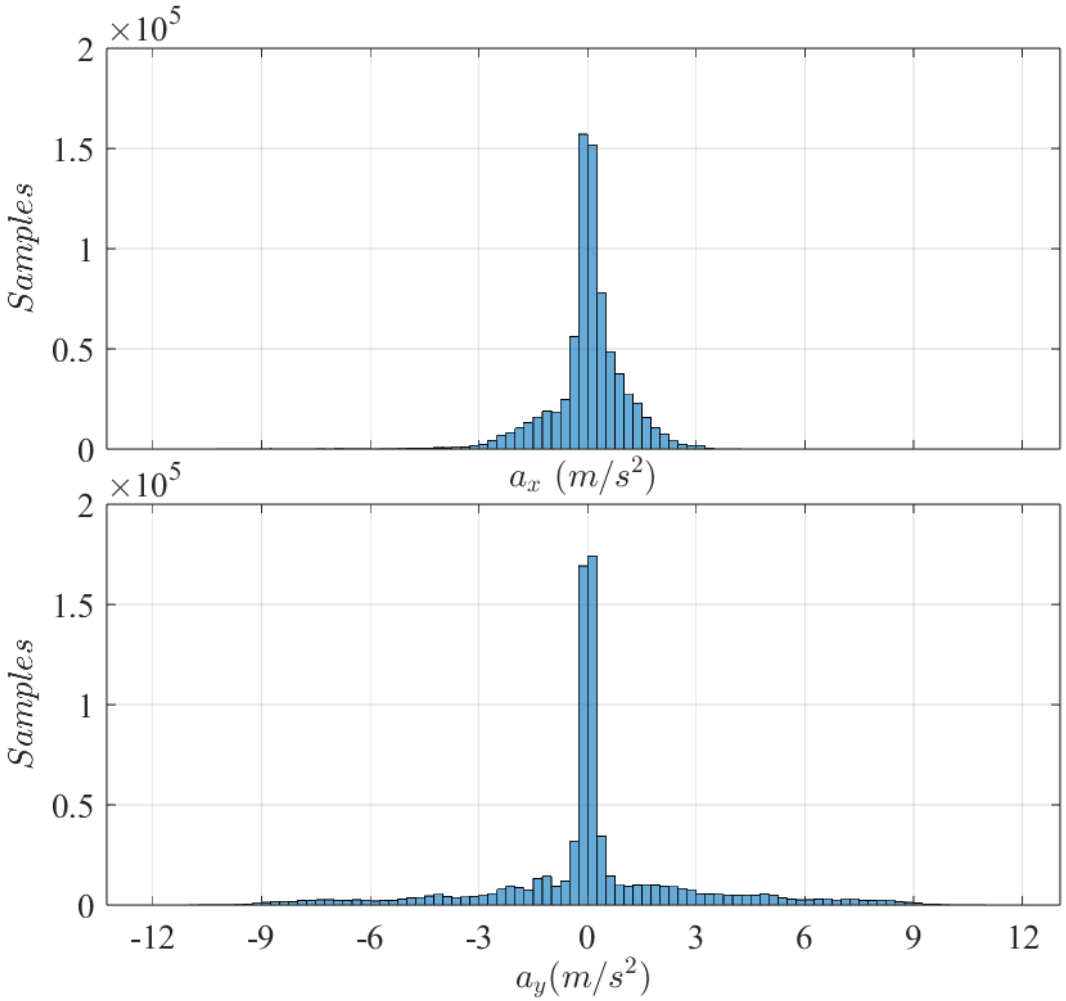


Figure 4.9 | Histograms of the non-normalized inputs.

	a_x	a_y
mean	0.0291	0.1127
standard deviation	0.9968	2.7555

Table 4.5 | Mean and standard deviation of non-normalized inputs.

When dealing with multiple outputs that have different value ranges, MSE training tends to prioritize accuracy for the output with the larger range over the one with the smaller range. To address this, the outputs of the training and validation datasets were normalized to have a zero mean and unit standard deviation [10]. Figure 4.10 presents histograms of the non-normalized outputs, clearly indicating the differences in means and

standard deviations. Table 4.6 lists the mean and standard deviation of the non-normalized outputs.

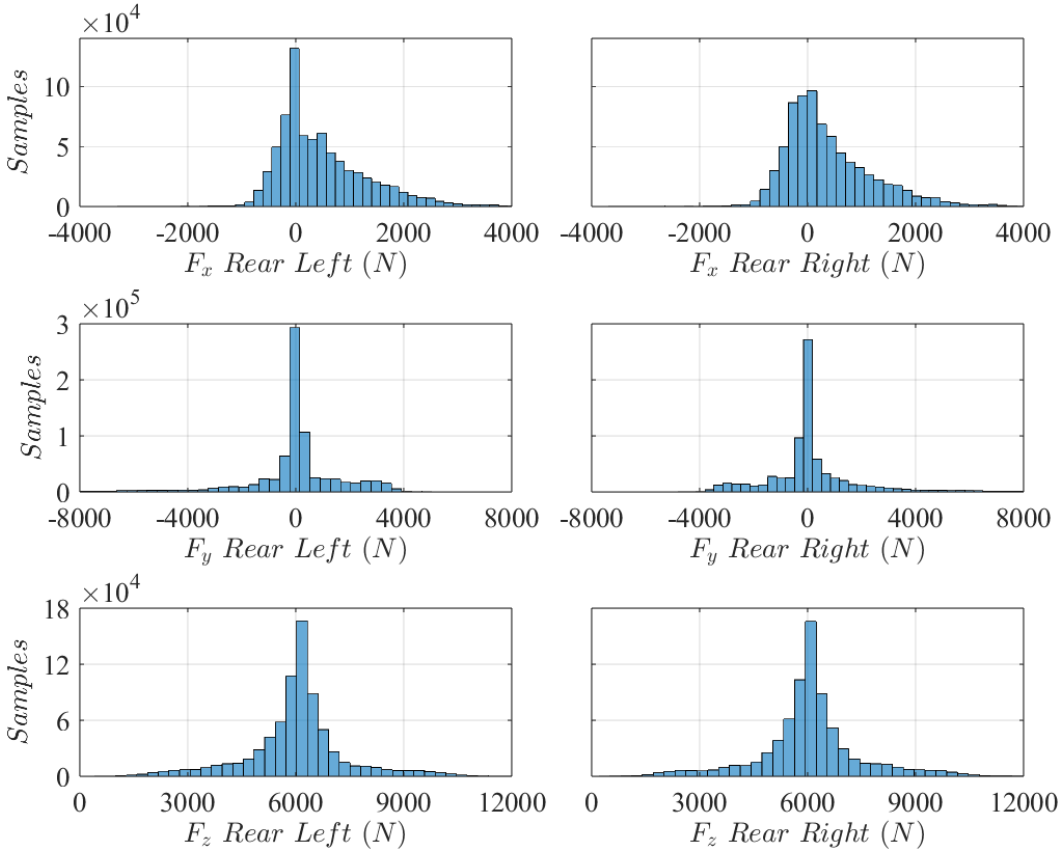


Figure 4.10 | Histograms of the non-normalized outputs.

	$F_{x_{RL}}(N)$	$F_{y_{RL}}(N)$	$F_{z_{RL}}(N)$	$F_{x_{RR}}(N)$	$F_{y_{RR}}(N)$	$F_{z_{RR}}(N)$
mean	472.14	−0.68	6021.99	364.29	113.65	6045.80
standard deviation	827.40	1694.94	1358.65	802.85	1677.35	1397.31

Table 4.6 | Mean and standard deviation of non-normalized outputs.

To enhance generalization, all input-output samples were shuffled [10]. After normalization and shuffling, the multi-input-multi-output pairs were split so that 80% of the pairs, corresponding to 100.4 minutes of driving time, were used for the training dataset, while the remaining 20%, corresponding to 25.1 minutes of driving time, were allocated to the validation set.

To predict the road-tire interaction forces at time $k+1$, inputs from not only time k but also earlier time instants were utilized. This approach allows the NN to leverage additional information for prediction, even if the measurements are the same. Figure 4.11 illustrates this concept. In each pair (i,j), element i represents the physical quantity, and element j denotes the time instant. m is the number of samples of each input quantity used for output prediction and serves as the hyperparameter to be optimized. n is the number of input quantities, which in this case is 2.

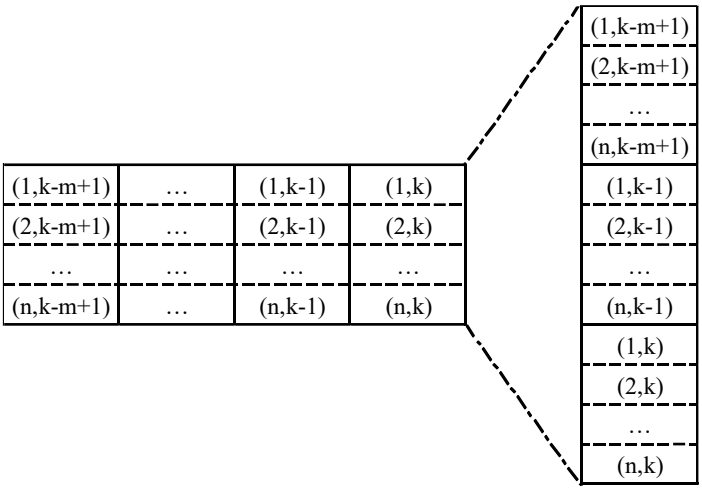


Figure 4.11 | Incorporation of $m-1$ samples preceding the current one k.

Bayesian optimization [74] was employed to determine the set of hyperparameters that minimize the Root Mean Square Error (RMSE) on the validation dataset, thereby identifying the model parameters that minimize the cost function. The ranges of the hyperparameters optimized hyperparameters resulting from this process are presented in Table 4.7.

Name	Range	Type	**Value**
Initial learn rate	$[10^{-2}, 10^{-1}]$	Real	0.0128
Learn rate drop factor	$[0.1, 0.3]$	Real	0.2466
Learn rate drop period	$[1, 2]$	Integer	1
Hidden units	$[500, 1500]$	Integer	1159

L2 regularization coefficient	$[10^{-6}, 10^{-2}]$	Real	0.0012
Dropout probability	$[0.1, 0.3]$	Real	0.1651
Mini batch size	$[128, 256, 512, 1024]$	Integer	1024
Sample number	$[1, 10]$	Integer	10

Table 4.7 | Ranges of variation and identified optimal hyperparameters.

The NN training progress was monitored by measuring the MSE on each mini-batch at every iteration. The training was conducted over six epochs in total. Figure 4.12 shows the reduction in training loss across iterations on a logarithmic scale, alongside the final MSE on the last mini-batch (Final Training Loss, FTL) and the final MSE on the validation dataset at the end of training (Final Validation Loss, FVL). The black vertical lines indicate the end of each epoch.

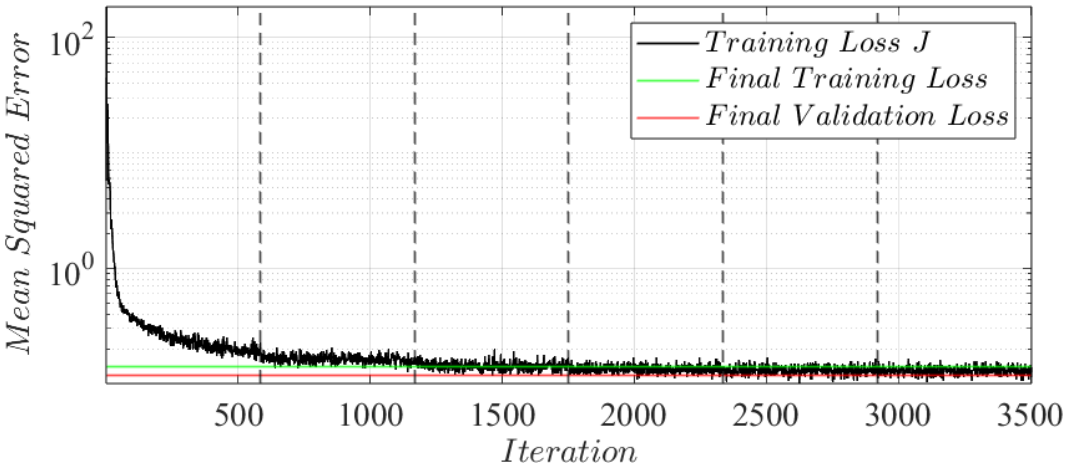


Figure 4.12 | Decrease in training loss at successive iterations.

FTL and FVL are equal to 0.13901 and 0.11718, respectively. Since the values assumed by these metrics exhibit the same order of magnitude, it can be concluded that the NN does not overfit the training dataset.

Tables 4.8 presents the RMSE between the non-normalized predictions and non-normalized references for the training and validation datasets [75].

Dataset	$F_{x_{RL}}\ (N)$	$F_{y_{RL}}\ (N)$	$F_{z_{RL}}\ (N)$	$F_{x_{RR}}\ (N)$	$F_{y_{RR}}\ (N)$	$F_{z_{RR}}\ (N)$
Training	176.11	323.35	383.90	179.95	327.83	399.78
Validation	176.80	322.45	382.58	180.06	329.97	398.80

Table 4.8 | RMSE between predicted and reference forces related to training and validation datasets.

4.2 | Estimation of the tire-road interaction forces by using Pacejka's formulas

This study presents a novel approach for estimating tire-road interaction forces by integrating models and measurements. A Central Difference Kalman Filter (CDKF) was applied to a Double Track Model, chosen to handle the system's nonlinearity. The tire-road interaction was modeled using Pacejka's magic formulas, which account for the combined effects of longitudinal and lateral slips, as well as the camber angle. This approach allowed for the execution of complex and realistic maneuvers. The developed estimator also considers the influence of lateral and longitudinal load transfers and aerodynamic forces in all three spatial directions. The camber angle in this observer was estimated using NNs. The measurements used include longitudinal velocity, yaw rate, longitudinal slip, and wheel steering angles.

4.2.1 | Estimation-oriented vehicle model

The reference systems considered are:

- Inertial reference system (SyO).
- Vehicle reference system (SyV).
- Wheel reference system (SyW).

The inertial reference system (SyO) is the earth-fixed reference system. It is defined as follows:

- (O) is the origin.

- (O, x, y) is the horizontal driving plane (road).
- (z) points upwards.

A vehicle model is made of three separate sets of equations:

- Congruence equations.
- Equilibrium equations.
- Constitutive (tire) equations.

The estimator was developed based on a rear-wheel-drive sports vehicle that has been experimentally validated. Simulations were conducted using CarMaker software [81] [82].

4.2.1.1 | Congruence equations

Congruence equations are a link between kinematic quantities. They relate the kinematics of the tires with the motion of the vehicle. The Slip Angles, shown in Figure 4.13 for the ij-wheel, are calculated using the estimates of longitudinal velocity u, lateral velocity v and yaw rate r expressed in the vehicle reference system (SyV). These quantities are shown in Figure 4.14.

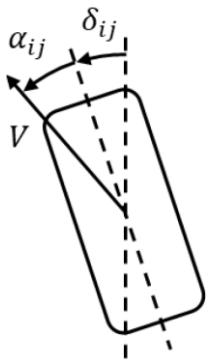


Figure 4.13 | Slip angle of the wheel.

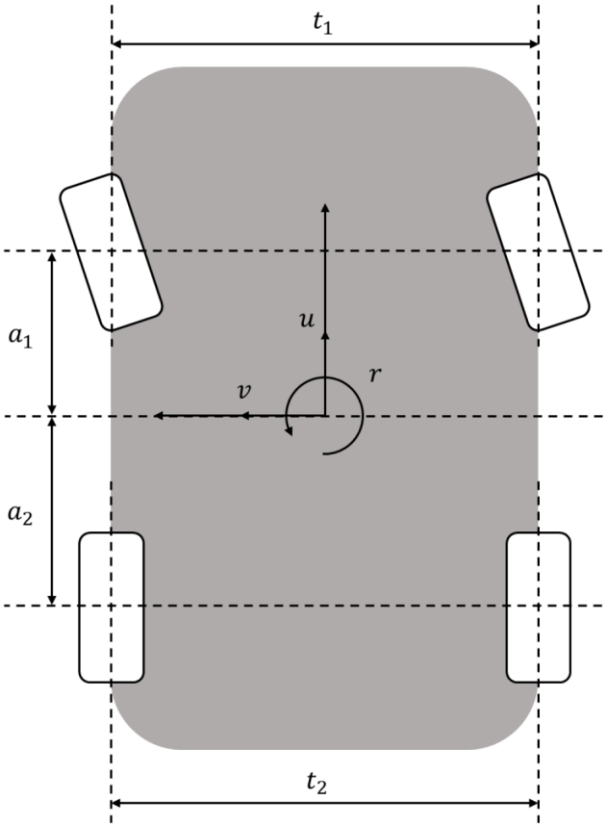


Figure 4.14 | Longitudinal velocity $\boldsymbol{u}$, lateral velocity $\boldsymbol{v}$ and yaw rate $\boldsymbol{r}$.

$$\begin{aligned}
\alpha_{11} &= \operatorname{atan}\left(\frac{v + ra_1}{u - r\frac{t_1}{2}}\right) - \delta_{11} \\
\alpha_{12} &= \operatorname{atan}\left(\frac{v + ra_1}{u + r\frac{t_1}{2}}\right) - \delta_{12} \\
\alpha_{21} &= \operatorname{atan}\left(\frac{v - ra_2}{u - r\frac{t_2}{2}}\right) \\
\alpha_{22} &= \operatorname{atan}\left(\frac{v - ra_2}{u + r\frac{t_2}{2}}\right)
\end{aligned} \tag{4.1}$$

The Longitudinal slips are provided as model inputs. The camber angles were obtained using NNs. A supervised training procedure has been employed for training NNs to fit the following nonlinear functions:

$$\begin{aligned} \gamma_{11} &= f_{11}(r, \alpha_{11}, \delta_{11}) \\ \gamma_{12} &= f_{12}(r, \alpha_{12}, \delta_{12}) \\ \gamma_{21} &= f_{21}(r, \alpha_{21}) \\ \gamma_{22} &= f_{22}(r, \alpha_{22}) \end{aligned} \tag{4.2}$$

Where steering angles δ_{ij}, slip angles α_{ij} and the yaw rate r are the inputs, while camber angles γ_{ij} are the outputs of NNs. Specifically, δ_{11} and δ_{12} are the steering angles of the front-left and front-right wheels. The inputs of the NN were chosen as they undergo a variation during the curve of the vehicle, as well as the output. The NNs made it possible to identify their dependence. The camber angles constitute inputs for Pacejka's Magic Formula.

4.2.1.2 | Constitutive equations

In each vehicle model, equations are set to relate the movement of the vehicle to the grip forces that each tire exchanges with the road. In this case, Pacejka's Magic Formulas were used. Magic-Formula tire models are considered state-of-the-art for modeling tire-road interaction forces in vehicle dynamics applications. Since 1987, Pacejka has published several versions of this type of tire model [76]. In this study, the latest development was used.

Since the movements of a vehicle primarily depend on the road forces on the tires, accurate knowledge of tire-road interaction forces is inevitable. The MF tire models consider the tire to behave as a parallel linear spring and linear damper in the radial direction with one point of contact with the road surface. The contact point is found by treating the tire and wheel as a rigid disc. The slip between the tire patch elements and the road and the inclination of the center plane of the wheel with respect to the road strongly influence the contact forces in the longitudinal and lateral directions. The inputs for the Magic Formula are the wheel load (F_z), the longitudinal (κ)

and lateral slip (α), and the inclination angle with the road or camber angle (γ).

There are three conditions of slip:

- *Pure cornering slip*: cornering with a free rolling tire.
- *Pure longitudinal slip*: braking or driving the tire without cornering.
- *Combined slip*: cornering and braking/driving simultaneously.

For pure slip conditions, the lateral force F_y as a function of the lateral slip α and the longitudinal force F_x as a function of longitudinal slip κ have a similar shape which can be described by the following formula:

$$y(x) = Dcos\{C\arctan[Bx - E(Bx - \arctan(Bx))]\}$$
$$Y(X) = y(x) + S_V \tag{4.3}$$
$$x = X + S_H$$

Where $Y(X)$ being F_x or F_y and X respectively κ or α. S_H is the horizontal and S_V is the vertical shift. The peak factor D determines the peak of the characteristic. The shape factor C mainly influences the shape of the curve. The stiffness factor B stretches the curve. The curvature factor E can modify the characteristic around the peak of the curve.

Figure 4.15 illustrates the meaning of the factors.

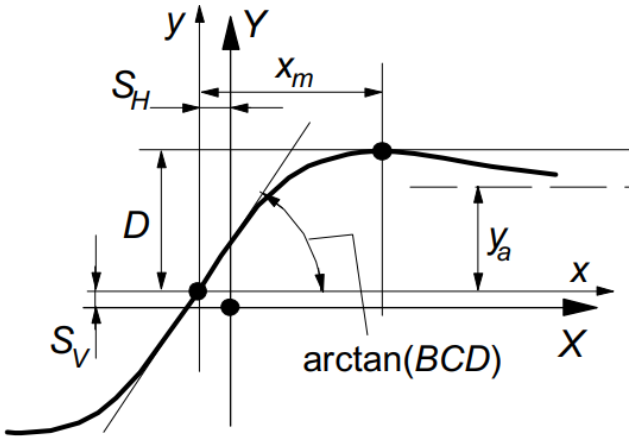


Figure 4.15 | Road-tire interaction force characteristic.

However, the pure slip condition represents an ideal condition. In real conditions, the longitudinal and lateral forces between the tire and the road depend on a combination of longitudinal and lateral slips. The combined slip version of the Magic Formula corrects previous formulas using weighting functions.

$$
\begin{aligned}
F_x &= F_{x0} G_{x\alpha}(\alpha, \kappa, F_z) \\
F_y &= F_{y0} G_{y\kappa}(\alpha, \kappa, F_z) + S_{Vy\kappa} \\
G_{x\alpha} &= \frac{\cos\{C_{x\alpha}\arctan[B_{x\alpha}\alpha_s - E_{x\alpha}(B_{x\alpha}\alpha_s - \arctan(B_{x\alpha}\alpha_s))]\}}{\cos\{C_{x\alpha}\arctan[B_{x\alpha}S_{Hx\alpha} - E_{x\alpha}(B_{x\alpha}S_{Hx\alpha} - \arctan(B_{x\alpha}S_{Hx\alpha}))]\}} \\
G_{y\kappa} &= \frac{\cos\{C_{x\kappa}\arctan[B_{y\kappa}\kappa_s - E_{y\kappa}(B_{y\kappa}\kappa_s - \arctan(B_{y\kappa}\kappa_s))]\}}{\cos\{C_{y\kappa}\arctan[B_{y\kappa}S_{Hy\kappa} - E_{y\kappa}(B_{y\kappa}S_{Hy\kappa} - \arctan(B_{y\kappa}S_{Hy\kappa}))]\}}
\end{aligned}
\tag{4.4}
$$

with $G_{x\alpha}$ the weighting function of the longitudinal force for pure slip and $G_{y\kappa}$ the weighting function for the lateral force at pure slip. $S_{Vy\kappa}$ is the κ-induced side force.

$$
\begin{aligned}
F_{z0ij} &= F_{z0ij}\lambda_{Fz0ij} \\
df_{zij} &= \frac{F_{zij} - F_{z0}}{F_{z0}} \\
S_{Hxij} &= (p_{Hx1} + p_{Hx2} df_{zij})\lambda_{Hxij} \\
S_{Vxij} &= F_{zij}(p_{Vx1} + p_{Vx2} df_{zij})\lambda_{Vxij}\lambda_{\mu xij} \\
\kappa_{xij} &= \kappa_{ij} + S_{Hxij} \\
C_{xij} &= p_{Cx1}\lambda_{Cxij} \\
D_{xij} &= \mu_x F_{zij} \\
E_{xij} &= \begin{pmatrix} p_{Ex1} + p_{Ex2} df_{zij} \\ + p_{Ex3} df_{zij}{}^2 \end{pmatrix} * (1 - p_{Ex4} sign(\kappa_x))\lambda_{Exij} \\
K_{xij} &= F_{zij}(p_{Kx1} + p_{Kx2} df_{zij}) e^{p_{Kx3} df_{zij}} \lambda_{Kxij} \\
B_{xij} &= \frac{K_{xij}}{C_{xij} D_{xij}} \\
F_{x0ij} &= D_{xij} sin\left\{ C_{xij} \arctan\left[\begin{matrix} B_{xij}\kappa_{xij} \\ -E_{xij}\left(\begin{matrix} B_{xij}\kappa_{xij} \\ -\arctan(B_{xij}\kappa_{xij}) \end{matrix} \right) \end{matrix} \right] \right\} + S_{Vxij}
\end{aligned}
\tag{4.5}
$$

$$\gamma_{yij} = \gamma_{ij}\lambda_{\gamma y_{ij}}$$

$$S_{Hyij} = (p_{Hy1} + p_{Hy2}df_{zij})\lambda_{Hyij} + p_{Hy3}\gamma_{yij}$$

$$S_{Vyij} = F_{zij}\begin{pmatrix}(p_{Vy1} + p_{Vy2}df_{zij})\lambda_{Vyij}\\ +(p_{Vy3} + p_{Vy4}df_{zij})\gamma_{yij}\end{pmatrix}\lambda_{\mu yij}$$

$$\alpha_{yij} = \alpha_{ij} + S_{Hyij}$$

$$C_{yij} = p_{Cy1ij} + \lambda_{Cyij}$$

$$D_{ijy} = \mu_y F_{zij}$$

$$E_{yij} = (p_{Ey1} + p_{Ey2}df_{zij})\begin{pmatrix}1 - (p_{Ey3}\\ +p_{Ey4}\gamma_{yij})sign(\alpha_{yij})\end{pmatrix}\lambda_{Eyij}$$

$$K_{y0ij} = p_{Ky1}F_{z0ij}\sin\left(2\arctan\left(\frac{F_{zij}}{p_{Ky2}F_{z0}\lambda_{Fz0ij}}\right)\right)\lambda_{Fz0ij}\lambda_{Kyij}$$

$$K_{yij} = K_{y0ij}(1 - p_{Ky3}|\gamma_{yij}|)$$

$$B_{yij} = \frac{K_{yij}}{C_{yij}D_{yij}}$$

$$F_{y0ij} = D_{yij}sin\left\{C_{yij}\arctan\left[\begin{matrix}B_{yij}\alpha_{yij}\\ -E_{yij}\begin{pmatrix}B_{yij}\alpha_{yij}\\ -\arctan(B_{yij}\alpha_{yij})\end{pmatrix}\end{matrix}\right]\right\} + S_{Vyij}$$

$$S_{Hx\alpha ij} = r_{Hx1}$$

$$B_{x\alpha ij} = r_{Bx1}\cos(\arctan(r_{Bx2}\kappa_{ij}))\,\lambda_{x\alpha ij}$$

$$C_{x\alpha ij} = r_{Cx1}$$

$$E_{x\alpha ij} = r_{Ex1} + r_{Ex2}df_{zij}$$

$$\alpha_{sij} = \alpha_{ij} + S_{Hx\alpha ij}$$

$$S_{Hy\kappa ij} = r_{Hy1} + r_{Hy2}df_{zij}$$

$$B_{y\kappa ij} = r_{By1}\cos(\arctan(r_{By2}(\alpha_{ij} - r_{By3})))\,\lambda_{y\kappa ij}$$

$$C_{y\kappa ij} = r_{Cy1}$$

$$E_{y\kappa ij} = r_{Ey1} + r_{Ey2}df_{zij}$$

$$\kappa_{sij} = \kappa_{ij} + S_{Hy\kappa ij}$$

Symbol	Name
p_{Cx1}	Shape factor for longitudinal force
p_{Ex1}	Longitudinal curvature at F_{znom}
p_{Ex2}	Variation of curvature with load
p_{Ex3}	Variation of curvature with load squared
p_{Ex4}	Factor in curvature while driving
p_{Kx1}	Longitudinal slip stiffness at F_{znom}
p_{Kx2}	Variation of slip stiffness with load
p_{Kx3}	Exponent in slip stiffness with load
p_{Hx1}	Horizontal shift F_{znom}
p_{Hx2}	Variation of shift with load
p_{Vx1}	Vertical shift at F_{znom}
p_{Vx2}	Variation of shift with load

Table 4.9 | Longitudinal force coefficients at pure slip.

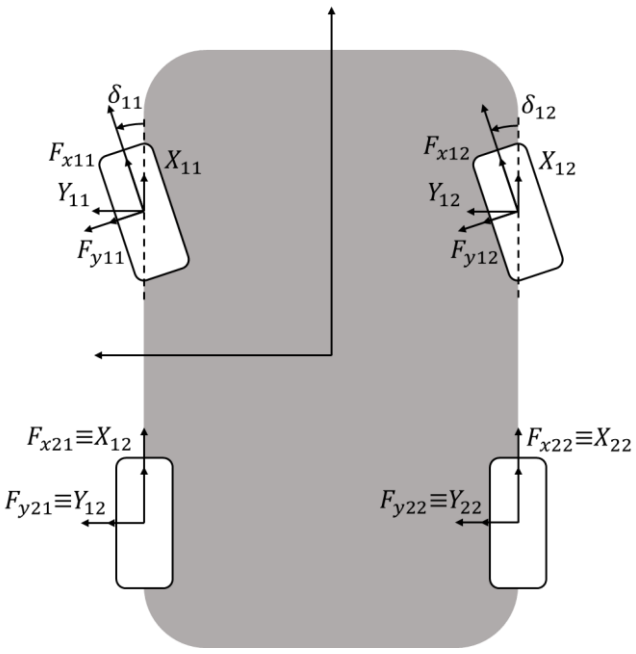


Figure 4.16 | Road-tire interaction forces in $\boldsymbol{FrW}$ and $\boldsymbol{Fr1}$ reference systems.

Name	Symbol
p_{Cy1}	Shape factor for lateral forces
p_{Ey1}	Longitudinal curvature at F_{znom}
p_{Ey2}	Variation of curvature with load

p_{Ey3}	Variation of curvature with load squared
p_{Ey4}	Factor in curvature while driving
p_{Ky1}	Longitudinal slip stiffness at F_{znom}
p_{Ky2}	Variation of slip stiffness with load
p_{Ky3}	Exponent in slip stiffness with load
p_{Hy1}	Horizontal shift F_{znom}
p_{Hy2}	Variation of horizontal shift with load
p_{Hy3}	Variation of horizontal shift with inclination
p_{Vy1}	Vertical shift at F_{znom}
p_{Vy2}	Variation of vertical shift with load
p_{Vy3}	Variation of vertical shift with inclination
p_{Vy4}	Variation of vertical shift with inclination and load

Table 4.10 | Lateral force coefficients at pure slip.

Symbol	Name
λ_{Fz0}	Scale factor of nominal (rated) load
λ_{Cx}	Scale factor of F_x shape factor
$\lambda_{\mu x}$	Scale factor of F_x peak friction coefficient
λ_{Ex}	Scale factor of F_x curvature factor
λ_{Kx}	Scale factor of F_x slip stiffness
λ_{Hx}	Scale factor of F_x horizontal shift
λ_{Vx}	Scale factor of F_x vertical shift
λ_{Cy}	Scale factor of F_y shape factor
$\lambda_{\mu y}$	Scale factor of F_y peak friction coefficient
λ_{Ey}	Scale factor of F_y curvature factor
λ_{Ky}	Scale factor of F_y cornering stiffness
λ_{Hy}	Scale factor of F_y horizontal shift
λ_{Vy}	Scale factor of F_y vertical shift
$\lambda_{x\alpha}$	Scale factor of α influence on F_x
$\lambda_{y\kappa}$	Scale factor of κ influence on F_y
$\lambda_{Vy\kappa}$	Scale factor of κ induced 'ply-steer' F_y

Table 4.11 | Scaling factor coefficients for pure slip.

The parameters of the Magic Formula and the friction coefficients correspond to those of the simulation model in IPG CarMaker. The scaling factors, used to correct the parameters of the formula, were tuned by making the vehicle perform a lap on the Nürburgring circuit at a maximum speed of 150 $\frac{km}{h}$ and comparing the outputs, longitudinal and lateral forces, of the Magic Formula (obtained using as input the values taken from the simulation vehicle) with the outputs of the IPG CarMaker model.

Pacejka's formulas return the forces in the FrW reference system. It is necessary to convert them to the $Fr1$ reference system for vehicle modeling purposes.

$$\begin{aligned} X_{1j} &= F_{x1j}\,cos(\delta_{1j}) - F_{y1j}sin(\delta_{1j}), \qquad j = 1,2 \\ Y_{1j} &= F_{x1j}\,sin(\delta_{1j}) + F_{y1j}cos(\delta_{1j}), \qquad j = 1,2 \\ X_{2j} &= F_{x2j}, \qquad j = 1,2 \\ Y_{2j} &= F_{y2j}, \qquad j = 1,2 \end{aligned} \tag{4.6}$$

4.2.1.3 | Equilibrium equations

During acceleration, due to the inertia forces, the vertical tire-road forces of interactions related to the rear axle increase, and those related to the front axle decrease. This phenomenon is called longitudinal load transfer. Similarly, during a curve, the vertical tire-road interaction forces related to the external tires increase, while the internal ones decrease. This phenomenon is called lateral load transfer.

In a motionless vehicle, the vertical loads must balance only the vehicle weight. The static loads on each axle are:

$$\begin{aligned} Z_1^0 &= \frac{mga_2}{l} \\ Z_2^0 &= \frac{mga_1}{l} \end{aligned} \tag{4.7}$$

The longitudinal load transfer is expressed by the following relationship:

$$\begin{gathered} Z_1 = Z_1^0 + \Delta Z^x \\ Z_2 = Z_2^0 - \Delta Z^x \\ \Delta Z^x = -\frac{m a_x h}{l} + \frac{J_{zx} r^2}{l} \approx -\frac{m a_x h}{l} \end{gathered} \tag{4.8}$$

a_x is the longitudinal acceleration. The second term considers the longitudinal load transfer due to a rotation of the vehicle around the vertical axis and can be neglected with respect to the first.

Suspension geometry and compliances directly influence lateral load transfer. In the first-order vehicle analysis shown in [76], the lateral load transfers are linear functions of Y_1 and Y_2, where Y_1 is the total tire-road interaction force relative to the front axle in the y direction of the $Fr1$ reference system and Y_2 is the total tire-road interaction force relative to the rear axle in the y direction of the $Fr1$ reference system.

$$\begin{gathered} \Delta Z_1^y = \xi_{11} Y_1 + \xi_{12} Y_2 \\ \Delta Z_2^y = \xi_{21} Y_1 + \xi_{22} Y_2 \end{gathered} \tag{4.9}$$

These coefficients were identified through a parametric identification procedure.

$$\begin{gathered} \xi_{11} = 0.2344 \\ \xi_{12} = 0.1193 \\ \xi_{21} = 0.1132 \\ \xi_{22} = 0.2243 \end{gathered} \tag{4.10}$$

Modelling of the aerodynamic forces has been included in the vehicle model to make it more complete. The aerodynamic forces acting on the vehicle are the Drag Force F_D, the Side Force F_s and the Lift Force F_L. The three forces act respectively in the x, y and z directions. The aerodynamic force has the same direction and opposite sign with respect to the vehicle-air relative velocity. The $* -$ subscript indicates the relative velocity. If the air velocity is zero, these relative velocities coincide with the absolute velocities u and v in the x and y directions.

$$\begin{aligned} \mathrm{F}_D &= \frac{1}{2}\rho C_D A u_*^2 \\ \mathrm{F}_s &= \frac{1}{2}\rho C_S A v_*^2 \\ \mathrm{F}_L &= \frac{1}{2}\rho C_L A u_*^2 \end{aligned} \tag{4.11}$$

Where:

- C_D: drag force coefficient.
- C_S: side force coefficient.
- C_L: lift force coefficient.

The three coefficients were identified through a parametric identification procedure.

$$\begin{aligned} C_D &= 0.32 \\ C_S &= 0.4 \end{aligned} \tag{4.12}$$

The vehicle was modeled as a double track with three degrees of freedom: longitudinal speed u, lateral speed v, and yaw rate r. The equations that characterize the dynamics of the system were obtained by applying the balance of forces in the x and y directions and around the z direction in the reference system $Fr1$.

The state of the system is:

$$\{x\} = \begin{Bmatrix} u \\ v \\ r \end{Bmatrix} \tag{4.13}$$

Since only the longitudinal speed and the yaw rate are measured, the output of the system is:

$$\{y\} = \begin{Bmatrix} u \\ r \end{Bmatrix} \tag{4.14}$$

In continuous time:

$$\begin{aligned} \dot{u} &= \frac{1}{m}(X_{11} + X_{12} + X_{21} + X_{22} - \mathrm{F}_D) + rv \\ \dot{v} &= \frac{1}{m}(Y_{11} + Y_{12} + Y_{21} + Y_{22} - \mathrm{F}_S) - ru \end{aligned} \tag{4.15}$$

$$\dot{r} = \frac{1}{J_z}\begin{pmatrix} a_1(Y_{11} + Y_{12}) - a_2(Y_{21} + Y_{22}) \\ -\frac{1}{2}t_1(X_{12} - X_{11}) + \frac{1}{2}t_2(X_{22} - X_{21}) \end{pmatrix}$$

In discrete time:

$$\begin{aligned} u_{k+1} &= u_k + \left(\frac{1}{m}\begin{pmatrix} X_{11k} + X_{12k} + X_{21k} \\ +X_{22k} - F_{Dk} \end{pmatrix} + r_k v_k\right) dt \\ v_{k+1} &= v_k + \left(\frac{1}{m}\begin{pmatrix} Y_{11k} + Y_{12k} + Y_{21k} \\ +Y_{22k} - \mathrm{F}_{Sk} \end{pmatrix} - r_k u_k\right) dt \end{aligned} \tag{4.16}$$

After modeling the system, state estimation was carried out through the use of a CDKF.

The overall estimator is schematically shown in Figure 4.17.

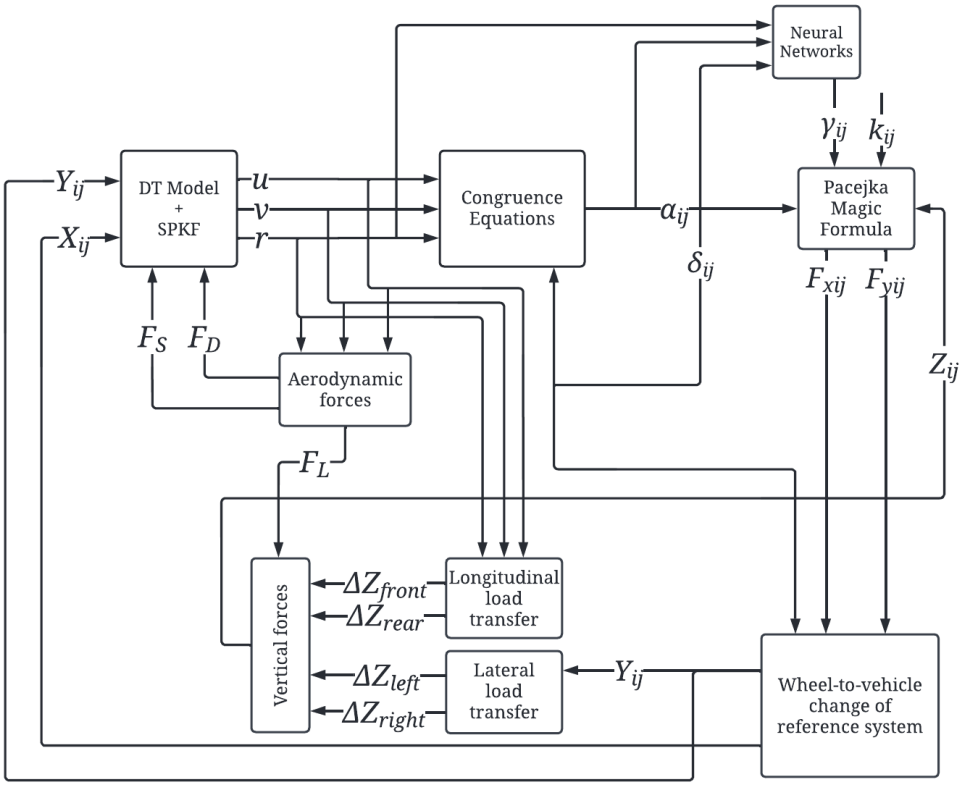


Figure 4.17 | Road-tire interaction forces estimator flow chart.

4.3 | Improvement of traction force estimation in cornering through neural network

Accurate traction force estimation is crucial for the advancement of control systems, particularly in the realm of autonomous driving. This study introduces an innovative approach to enhance the estimation of tire-road interaction forces under combined slip conditions, leveraging a combination of empirical models and NNs. Initially, the well-known Pacejka's Formula, or Magic Formula, was employed to estimate tire-road interaction forces under pure longitudinal slip conditions. However, it was found that the formula produced unsatisfactory results under non-pure slip conditions, such as during cornering. To address this issue, an NN architecture was developed to predict the estimation error associated with Pacejka's Formula. Two distinct NNs were developed for this purpose. The first network used as inputs the longitudinal slip ratios and slip angles of the traction wheels. The second network utilized the longitudinal slip ratios of the traction wheels along with the vehicle's longitudinal and lateral accelerations. Both networks were designed as multi-output models, capable of simultaneously estimating the longitudinal force errors for both traction wheels. The NNs were trained using data from straight-line accelerations, circuit maneuvers, and sinusoidal steering maneuvers. The performance of the estimator was tested by completing two laps on the Hockenheim circuit in the reverse direction. The initial Root Mean Square Error (RMSE) was significantly reduced through the use of these corrective NNs. These results confirm the effectiveness of the NN-based approach in improving traction force estimation under combined slip conditions, addressing the limitations of Pacejka's Formula in cases of non-pure slip,

and opening up new possibilities for the development of more advanced and safer vehicle control systems.

To define the physical quantities within this estimator, it is essential to establish reference systems: SyV vehicle reference system and SyW wheel reference system.

This estimator is designed to predict the forces exerted by the road on the tires of a rear-wheel-drive vehicle, specifically the rear-left traction force F_{rl} and the rear-right traction force F_{rr}. These forces are aligned with the x-axis of the SyW reference system and are illustrated in Figure 4.18.

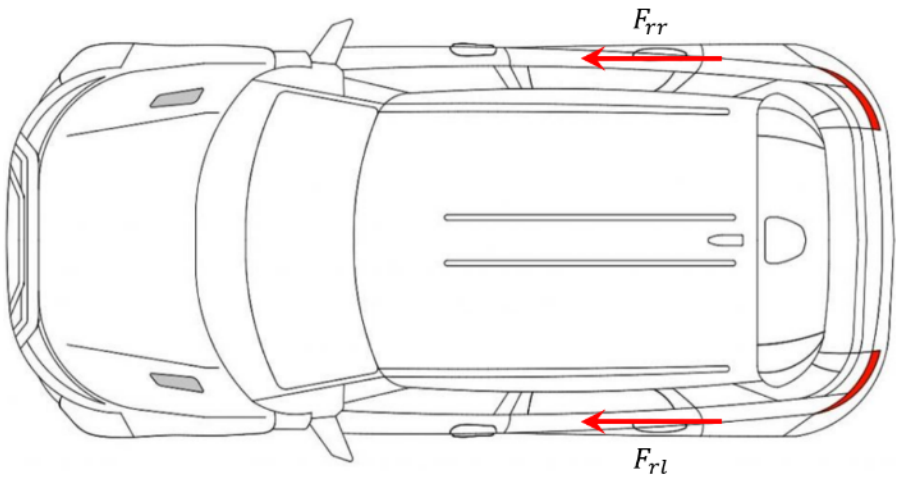


Figure 4.18 | Traction forces $\boldsymbol{F_{rl}}$ and $\boldsymbol{F_{rr}}$.

4.3.1 | Pediction using the Pacejka's formula

In the context of longitudinal forces under pure longitudinal slip conditions, Pacejka's formula can be simplified and expressed as follows:

$$\tilde{F}_{rj} = D \cdot \sin\left(C \cdot \arctan\left(B \cdot \kappa_{rj} - E \cdot \left(B \cdot \kappa_{rj} - \arctan\left(B \cdot \kappa_{rj}\right)\right)\right)\right) \qquad (4.17)$$

Equation (4.17) represents the longitudinal force $\tilde{F}_{rj}$ generated by a tire mounted on wheel j (left or right) of the rear axle r, as a function of the longitudinal slip κ_{rj}.

The coefficients in the equation have the following meanings:

- C *(Shape factor)*: This parameter affects the steepness of the force function relative to slip (in this case, the longitudinal slip ratio). It controls how quickly the force increases or decreases in response to changes in slip. A higher value of C makes the curve sharper around zero slip, resulting in a more sensitive response to slip variations.
- D *(Peak factor)*: This scaling parameter influences the maximum force generated by the tire. It represents the tire's maximum potential force output. Higher values of D lead to greater overall longitudinal forces.
- B *(Stiffness factor)*: The stiffness parameter B affects the shape of the force function. It determines how rapidly the force changes with slip variations. Higher B values produce a quicker response, while lower values yield a more gradual response.
- E *(Curvature factor)*: This parameter is used to adjust the function's shape around zero slip. It introduces a correction in the term arctan $(B \cdot \kappa_{rj})$, modeling additional curvature in the force response based on longitudinal slip. A positive E value tends to smooth the behavior around zero, while a negative value tends to enhance it.

These coefficients were determined by conducting straight-line acceleration maneuvers and fitting curves to the resulting data derived from Pacejka's formula. Four acceleration maneuvers with maximum longitudinal accelerations of 0.5, 1, 1.5, and 2 $\frac{m}{s^2}$ were performed starting from zero velocity and reaching a velocity of 180 $\frac{km}{h}$. The traction force and longitudinal slip ratio values of both driving wheels were recorded. The data obtained for all simulations were then aggregated and used for the curve fitting procedure.

The curve-fitting procedure consists of the following steps:

1. *Definition of the quadratic cost function*:

$$J = \frac{1}{2} \sum_{j=L,R} \sum_{k=1}^{n} (\tilde{F}_{rj_k}(B, C, D, E) - F_{rj_k})^2 \tag{4.18}$$

   - $\tilde{F}_{rj_k}$ is the estimated traction force with Pacejka's formula, parameterized with respect to B, C, D, E, related to tire rj, obtained at the k-th time instant.
   - F_{rj_k} is the reference traction force related to tire rj, obtained at the k-th time instant.
   - n is the number of samples.
2. *Random selection from a normal Gaussian distribution of the initial values of* B, C, D, E.
3. *Calculation of the gradient of the cost function*. The gradient of the cost function with respect to the parameters B, C, D, E, denoted as $\nabla J(B, C, D, E)$, is a vector containing the partial derivatives of the cost function with respect to each parameter. This vector indicates the direction and magnitude of the maximum decrease in the cost function with respect to each parameter.
4. *Update Parameters via Gradient Descent* [77].
5. *Iteration. Steps 3 and 4 were repeated until the cost function converged to a minimum value*.

Convergence verification to a global minimum. Steps 3, 4, and 5 were repeated 100 times with different initial values to find the global minimum among all minima.

Since the tire mounted on the rear-left wheel is identical to that on the rear-right wheel, the same coefficients were obtained for both tires. Figure

4.19 displays the collected data, the fitted curve, and the identified coefficients.

At low values of the longitudinal slip ratio, there is a phase where an increase in longitudinal slip results in a linear increase in traction force. In this range, the traction force is directly proportional to the slip. As the slip continues to increase, a maximum grip point is eventually reached, where the longitudinal force reaches its peak. Beyond this maximum grip point, any further increase in longitudinal slip leads to a decrease in longitudinal force, indicating that the tire is entering a saturation phase and losing grip.

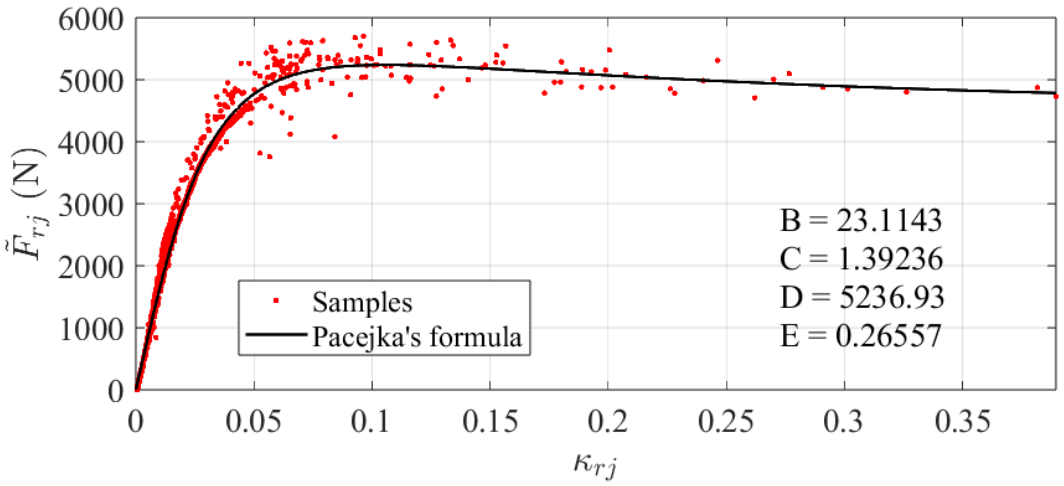


Figure 4.19 | $\tilde{F}_{rj}$ Pacejka's traction force vs κ_{rj} longitudinal slip ratio: comparison between acquired data and interpolating curve.

4.3.2 | Corrective neural networks

The estimation of tire forces using the simplified Pacejka's formula can be improved by incorporating additional coefficients that account for specific tire behaviors. However, implementing such formulations is complex due to the difficulties in identifying these coefficients, which require a significant blend of theoretical knowledge and practical experience from the operator conducting the tests [78].

To improve the accuracy of longitudinal force estimation, a parallel NN has been developed that uses traction wheel slip angles or vehicle accelerations, in addition to longitudinal slip ratios. This NN is designed to quickly assess the discrepancy, denoted as ΔF_{rj} $(j = l, r)$, between the estimation derived solely from Pacejka's formula and the reference force.

The corrections are calculated as the difference between the forces obtained from the executed maneuvers and the forces computed using Pacejka's formula with the identified coefficients:

$$\Delta F_{rj} = F_{rj} - \tilde{F}_{rj} \tag{4.19}$$

Figure 4.20 illustrates the overall estimator.

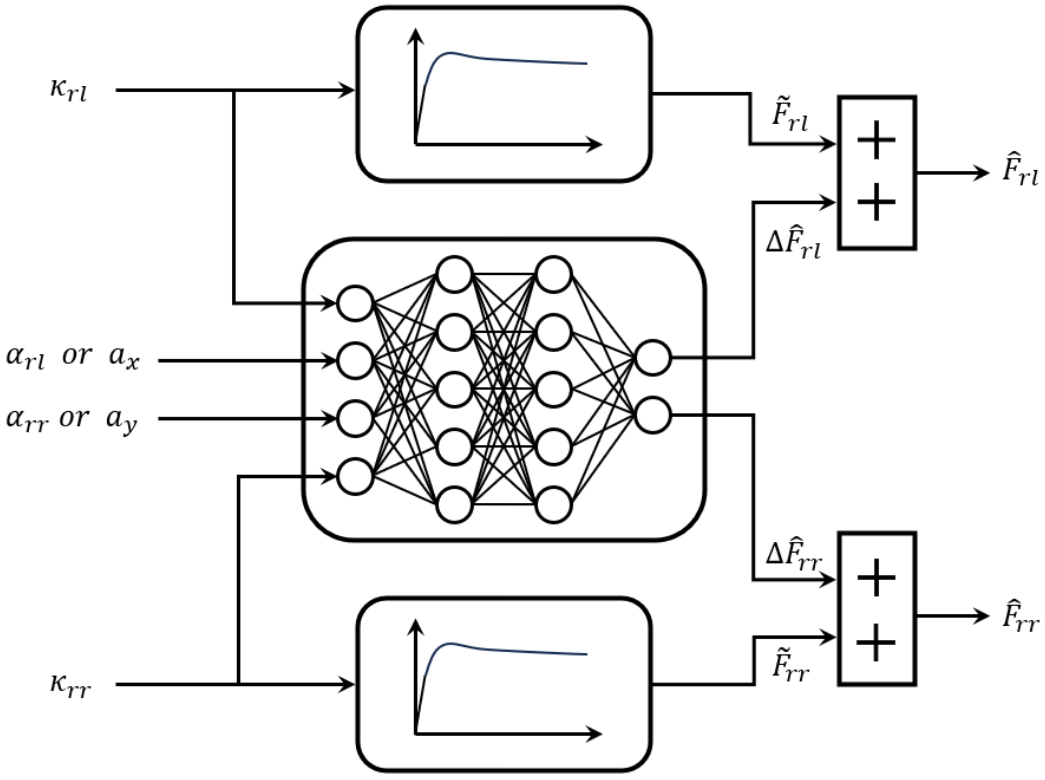


Figure 4.20 | Schematic representation of the estimator used for the prediction of longitudinal tensile forces with evidence of the combination of Pacejka's model and corrective NN.

4.3.3 | Choice of inputs

The two NNs presented in this study utilize six distinct input variables, of which four for each NN:

- *Longitudinal slip ratios of traction tires* κ_{rl}, κ_{rr}: It is well-established in the literature [78] that a vehicle's traction forces depend on the longitudinal slip between the tire and the road surface. These values are expressed in the SyW reference system.
- *Slip angles of traction tires* α_{rl}, α_{rr}: Research indicates [78] a connection between cornering traction force and the vehicle's slip angle. These values are also expressed in the SyW reference system.
- *Longitudinal acceleration* a_x: The traction forces experienced by a vehicle are directly related to its longitudinal acceleration, as any object subjected to force will accelerate proportionally in the same direction [79] [80]. These values are expressed in the SyV reference system.
- *Lateral acceleration* a_y: The traction forces on a vehicle change during cornering. Increased longitudinal acceleration leads to a greater deviation from pure longitudinal slip, resulting in larger variations in longitudinal force at a constant slip ratio.

The primary goal of this study is to investigate longitudinal interaction forces specifically under traction conditions. Consequently, from all the samples collected during the maneuvers, only those in which the longitudinal slip ratios of both the left and right tires are greater than zero were selected for analysis.

4.3.4 | Data collection

The estimator was developed based on a rear-wheel-drive sports vehicle that has been experimentally validated. Simulations were conducted using CarMaker software [81] [82]. The maneuvers carried out to gather the necessary data for the training and validation sets include the following:

- *Fourteen laps on the Nürburgring track*, with seven laps completed in each direction, as shown in Figure 4.21. The driver was instructed to navigate the track while performing various accelerations, ensuring a diverse range of data points. During the seven laps in each direction, the driver achieved maximum longitudinal cornering accelerations ranging from 0.3 to 2.1 $\frac{m}{s^2}$, increasing in increments of 0.3 $\frac{m}{s^2}$. Higher longitudinal cornering accelerations were not attempted, as the vehicle was unable to maintain a grip on the road.

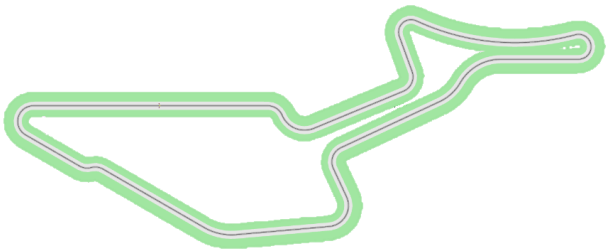


Figure 4.21 | Top view of the Nürburgring track used for training NNs.

- *Twelve sinusoidal steering maneuvers*, each consisting of five periods lasting 2 s each. The amplitude of these maneuvers varied from 30° to 180° in increments of 30°. The velocity increased across six maneuvers, starting at 10 $\frac{km}{h}$ for six cases and 60 $\frac{km}{h}$ for the remaining six, until the lateral stability limit was reached, beyond which the vehicle lost grip. This approach emulates challenging conditions for the vehicle's cornering acceleration.

In addition to these maneuvers, data from acceleration maneuvers used to identify the coefficients of Pacejka's formula were also included. This addition was strategically implemented to enhance the NNs' understanding of the vehicle's operational scenarios.

Two multi-output NNs, referred to as NN1 and NN2, were developed, each with four inputs and two outputs. Both networks use the longitudinal slip ratios of both tires as inputs. The first network also includes the slip angles of the two traction tires. However, measuring or calculating tire slip angles is challenging, as they depend on the longitudinal and lateral velocities of the vehicle, which are difficult to measure accurately [83]. Furthermore, calculating tire slip angles requires knowledge of the vehicle's geometric characteristics, such as wheelbase, front and rear track widths, and suspension type [84]. In this study, the slip angle of each tire is calculated using the following equation [82]:

$$\alpha = \frac{v_y^{(P)}}{\left| v_x^{(P)} \right|} \tag{4.20}$$

where $v_x^{(P)}$ and $v_y^{(P)}$ represent the longitudinal and lateral velocities at the tire-road contact point, expressed in the SyW reference system.

To address this challenge, the second network incorporates the longitudinal and lateral accelerations of the vehicle as additional inputs, which can be measured using an IMU. Acceleration measurements are easier to obtain than measuring or estimating tire slip angles, and their use allows the estimator to be unaffected by vehicle specifications. Differences in wheelbase, front and rear track widths, and suspension types can lead to variations in the accelerations experienced by the vehicle, as measured by the IMU. However, IMU measurements occur downstream of these

variations, making the corrective NN independent of vehicle characteristics.

Figures 4.22 and 4.23 show, by means of density matrices, the orders of magnitude of the acquired data and how they are related to each other.

The physical relationship between the longitudinal slip ratio and slip angle of the tire is not easy to interpret. However, it is easy to see that when a tire has a positive (negative) slip angle, which occurs when the vehicle turns right (left), the inner tire, right (left) when the vehicle turns right (left), experiences an increase in the longitudinal slip ratio. It is also easy to see that the slip angle of a tire changes linearly with respect to the slip angle of the opposite tire.

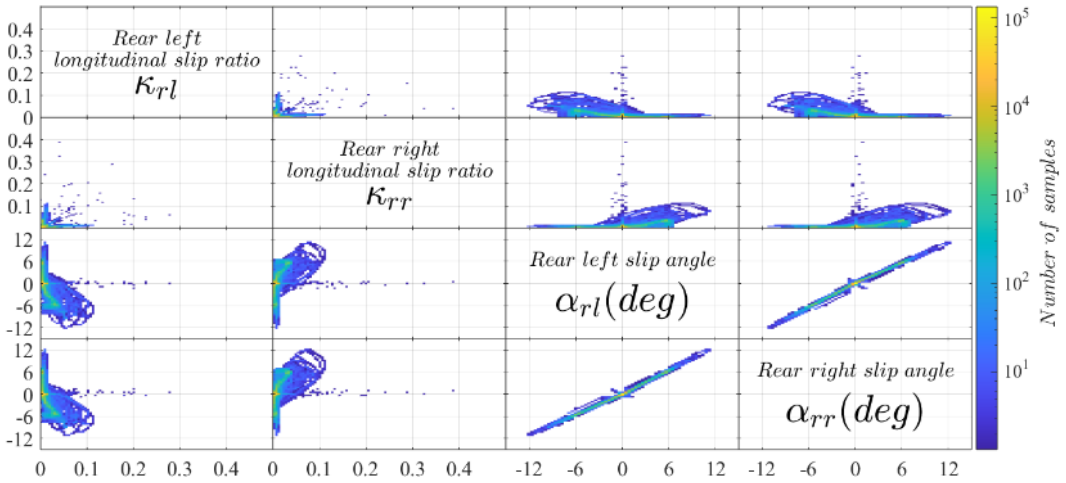


Figure 4.22 | NN1 input (longitudinal slip ratios κ_{rl}, κ_{rr} and slip angles α_{rl}, α_{rr}) density matrix.

Similarly, in the case of the relationship between the longitudinal slip ratio and acceleration, it can be seen that when the lateral acceleration is positive (negative), which occurs when the vehicle turns left (right), the inner tire, left (right) when the vehicle turns left (right), experiences an increase in the longitudinal slip ratio.

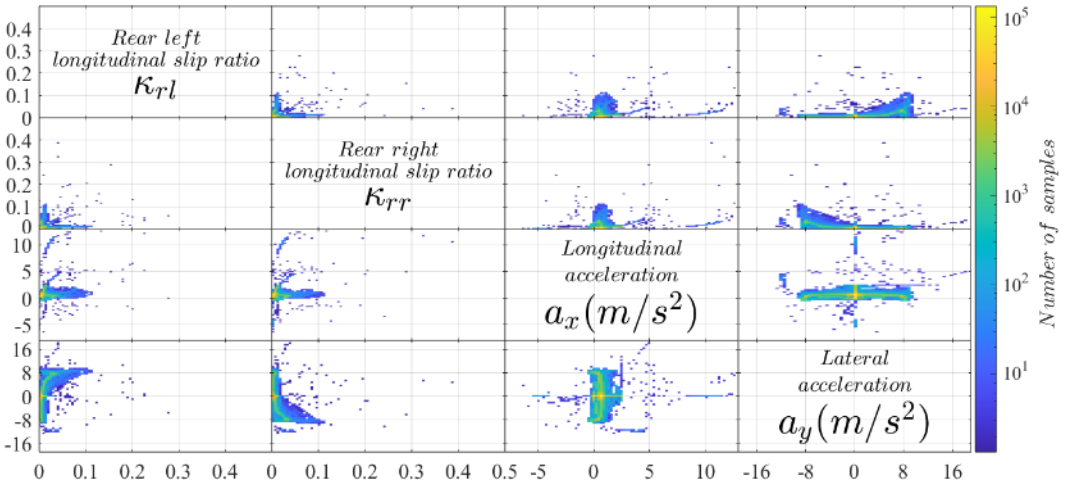


Figure 4.23 | NN2 input (longitudinal slip ratios $\boldsymbol{\kappa_{rl}, \kappa_{rr}}$ and accelerations $\boldsymbol{a_x, a_y}$) density matrix.

Figures 4.24-4.29 show how each input is related to the two outputs. On the left are scatter plots, and on the right are density plots.

When the vehicle experiences a curve, the traction force undergoes a non-linear decrease with respect to the longitudinal slip ratio of the tire. This decrease shows saturation as the longitudinal slip increases. Physically, this means that a tire's cornering behavior is more distant from its straight-line behavior the greater the vehicle's traction in cornering. The relationship between the longitudinal slip ratio of one tire and the traction force of the opposite tire is complex, but there is still a decrease compared to pure longitudinal slip.

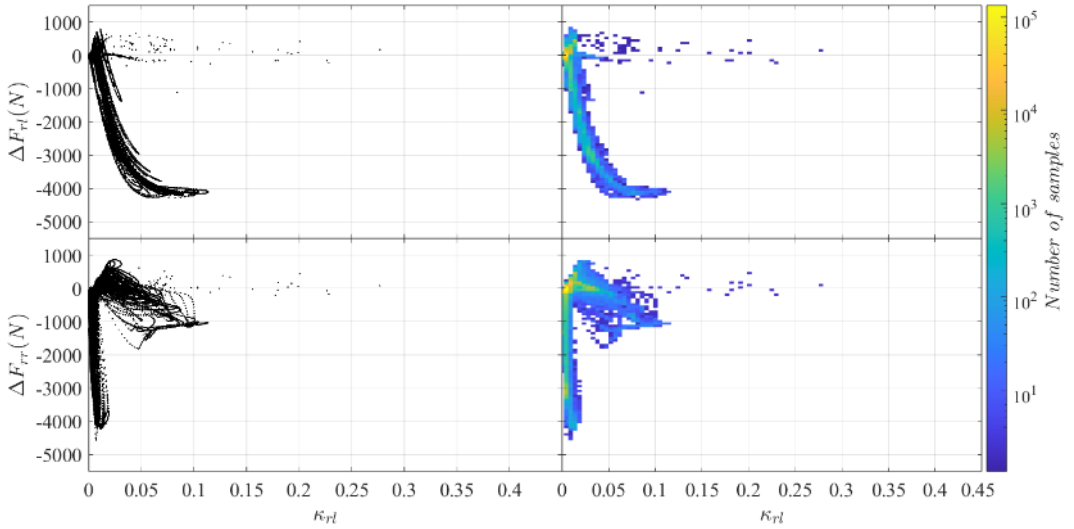


Figure 4.24 | Scatter plot (left) and density plot (right) of the traction force correction $\boldsymbol{\Delta F_{rj}}$ vs rear-left longitudinal slip ratio $\boldsymbol{\kappa_{rl}}$.

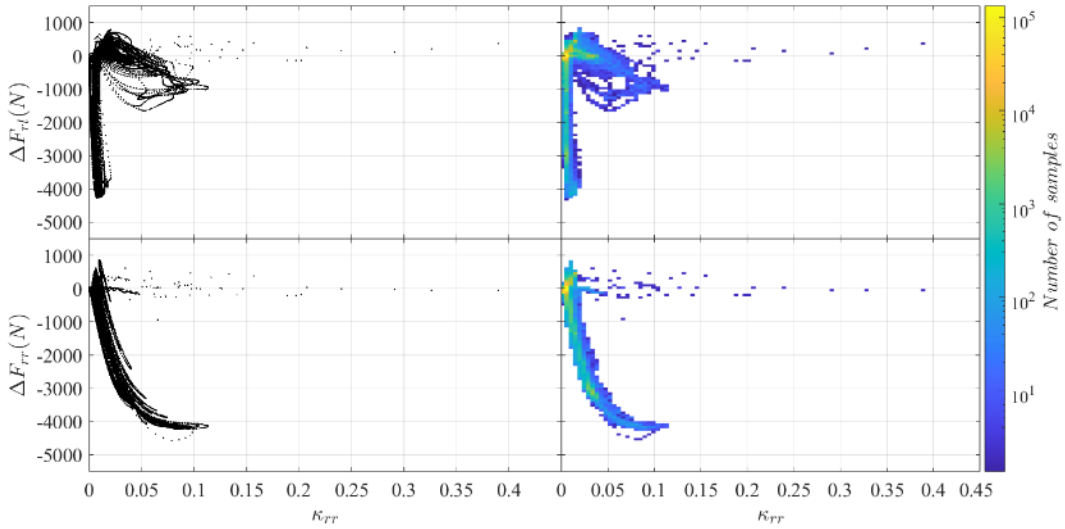


Figure 4.25 | Scatter plot (left) and density plot (right) of the traction force correction $\boldsymbol{\Delta F_{rj}}$ vs rear-right longitudinal slip ratio $\boldsymbol{\kappa_{rr}}$.

When the sideslip angle of a tire is positive (negative), i.e., when the vehicle is driving through a right (left) turn, the inner tire, right (left) in the case of a right (left) turn, experiences a decrease in traction force compared to that in pure longitudinal slip conditions. The relationship between the slip angle of one tire and the traction force correction of the opposite tire is symmetrical to that of the same tire.

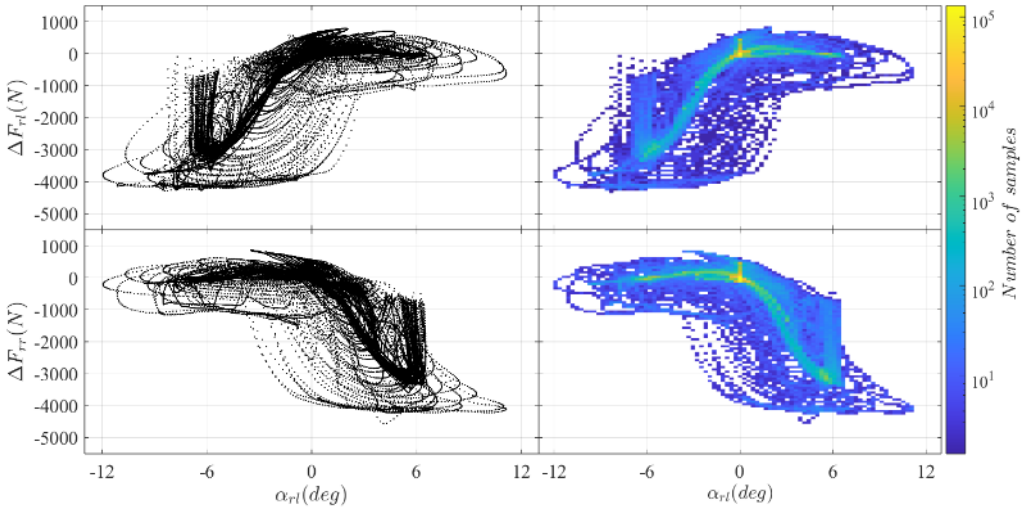


Figure 4.26 | Scatter plot (left) and density plot (right) of the traction force correction $\boldsymbol{\Delta F_{rj}}$ vs rear-left slip angle $\boldsymbol{\alpha_{rl}}$.

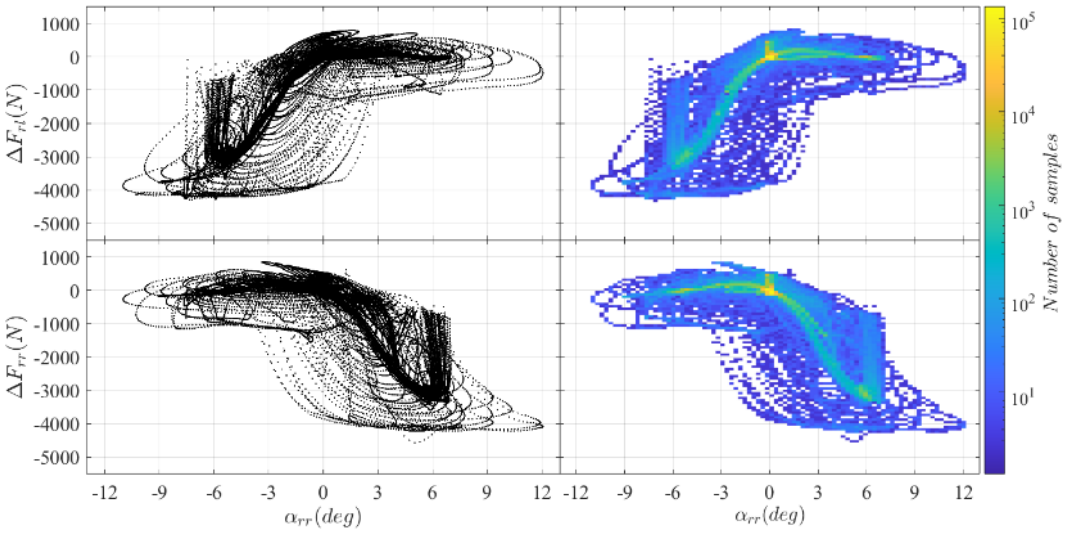


Figure 4.27 | Scatter plot (left) and density plot (right) of the traction force correction $\boldsymbol{\Delta F_{rj}}$ vs rear-right slip angle $\boldsymbol{\alpha_{rr}}$.

The traction force experiences a general decrease in cornering compared to that calculated in pure longitudinal slip conditions. The vehicle cannot easily negotiate curves with accelerations greater than $2\frac{m}{s^2}$, and values outside the $0 \div 2\frac{m}{s^2}$ range represent outliers.

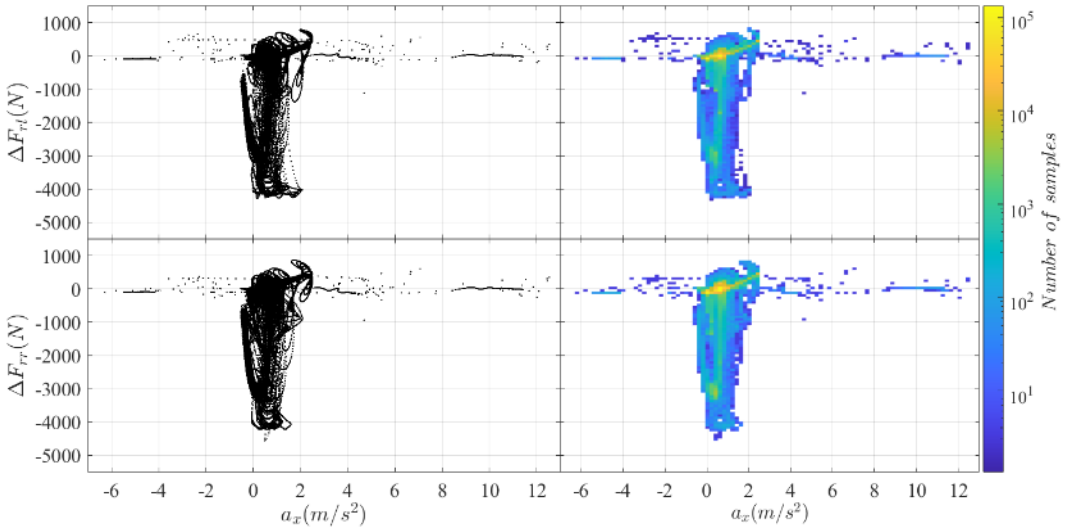


Figure 4.28 | Scatter plot (left) and density plot (right) of the traction force correction $\boldsymbol{\Delta F_{rj}}$ vs longitudinal acceleration $\boldsymbol{a_x}$.

When the lateral acceleration is negative (positive), i.e., when the vehicle is driving through a right (left) turn, the inner tire, right (left) in the case of a right (left) turn, experiences a decrease in traction force compared to that in pure longitudinal slip conditions.

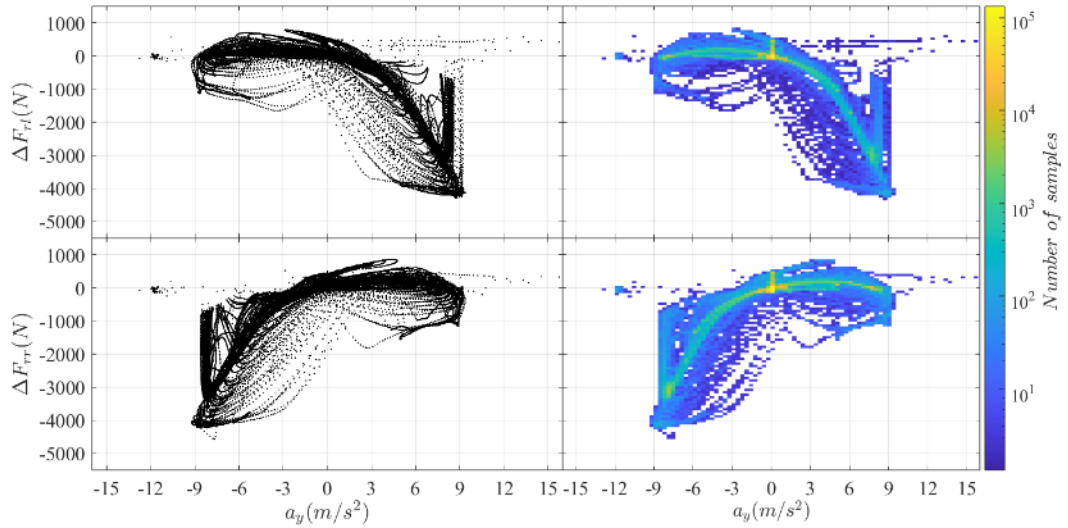


Figure 4.29 | Scatter plot (left) and density plot (right) of the traction force correction $\boldsymbol{\Delta F_{rj}}$ vs lateral acceleration $\boldsymbol{a_y}$.

The longitudinal and lateral slips of one tire are also related to the force of the other tire. This demonstrates the advantage of using a multi-output

network rather than two single-output networks for each tire. In this way, more data can be obtained for output prediction simply by making more efficient use of the available data.

All input-output samples were then shuffled to improve generality [85]. The multi-input-multi-output sample pairs were divided into training and validation datasets with an $80-20$ proportion.

4.3.5 | Neural networks development

The NNs are structured as follows:

- Input layer.
- First hidden layer with Rectified Linear Unit (ReLU) activation function.
- Second hidden layer with Rectified Linear Unit (ReLU) activation function.
- Output layer.

The Bayesian optimization method [74] was used to train the two NNs. The capacity of Bayesian optimization to effectively explore the hyperparameter space and make well-informed decisions about which configurations to assess next makes it especially well-suited for this goal. Unlike traditional grid or random search methods [86] [87], Bayesian optimization leverages a probabilistic model to guide the search process, allowing for a more targeted and effective optimization process by iteratively selecting the most promising configurations. For each trial, the Adam optimizer was utilized [88].

With Bayesian optimization, it was possible to determine the set of hyperparameters that minimized the RMSE of the validation dataset and, as a result, the model parameters that minimized the loss function. The

range of variation, the optimal hyperparameters, and the RMSEs found during optimization for the two NNs are shown in Table 4.12.

Name	Range	Type	Value NN1	Value NN2
Initial learn rate	$[1^{-2} \div 1^{-1}]$	Real	0.0307	0.0101
Hidden units	$[100 \div 1000]$	Integer	843	841
L2 regularization coefficient	$[1^{-6} \div 1^{-2}]$	Real	$4.5 \cdot 10^{-6}$	$7.5 \cdot 10^{-3}$
Dropout probability	$[0.1 \div 0.3]$	Real	0.1867	0.1005
Mini batch size	$[512, 1024, 2048, 4096]$	Integer	1024	1024
MSE of the validation dataset			1649.5 N	1440.8 N

Table 4.12 | Ranges of variation of hyperparameters, optimal hyperparameters, and RMSE of the validation dataset of NN1 and NN2.

Figures 4.30 and 4.31 show the decrease in the Training Loss during the training of the two NNs and the relative Validation Loss. The Validation Loss is calculated every four epochs for both networks. Training lasts for 20 epochs each, and each epoch contains approximately 229 mini-batches.

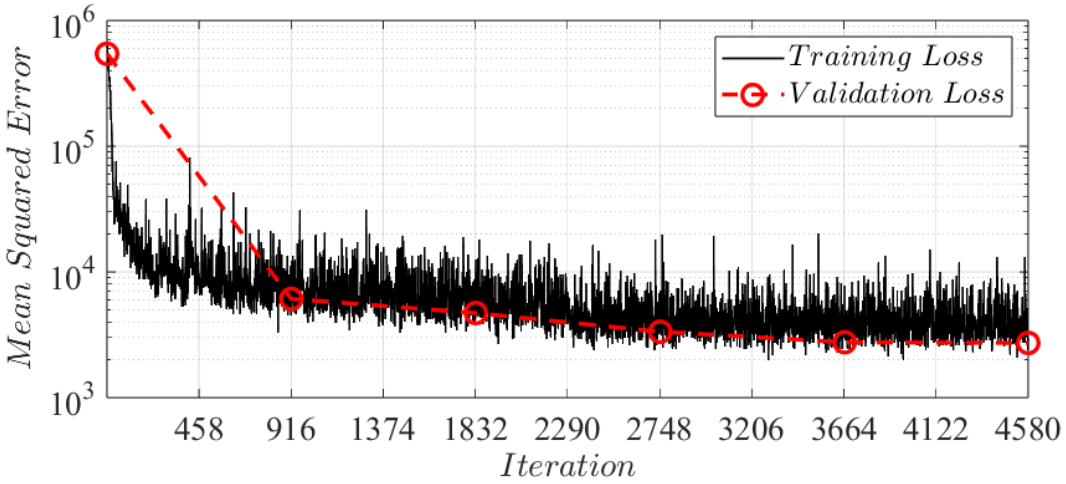


Figure 4.30 | Decreasing Training and Validation Loss on a logarithmic scale during NN1 training.

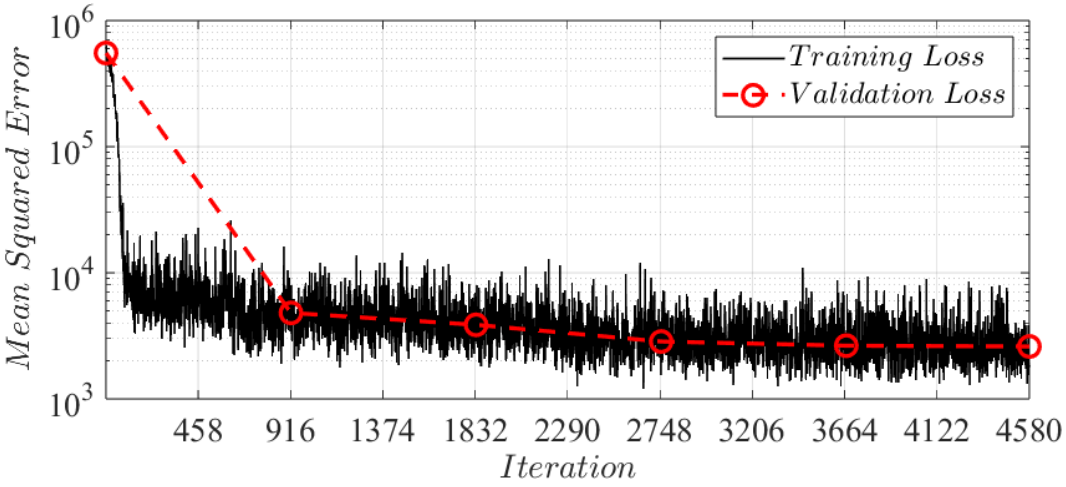


Figure 4.31 | Decreasing Training and Validation Loss on a logarithmic scale during NN2 training.

The NN1 and NN2 show a comparable MSE of the validation dataset, demonstrating that the use of accelerations, simpler to measure or estimate than the slip angle, still allows an accurate estimate of the correction on the error of the Pacejka's formula and consequently an accurate estimation of traction force.

4.4 | Prediction of the sideslip angle using dynamic neural networks

With the increasing interest in autonomous vehicles, safety in vehicle operation is becoming an increasingly critical focus. The sideslip angle is a key parameter for modern control systems designed to enhance passenger safety, as it directly impacts the lateral motion and stability of a vehicle. A large sideslip angle can lead to oversteer or understeer, resulting in a loss of control and potentially causing an accident. Therefore, it's crucial to continuously monitor this parameter while driving to take corrective actions if needed. However, sensors that directly measure the sideslip angle are costly and difficult to implement. This study proposes two NNs for estimating the sideslip angle. The factors that most significantly influence

the vehicle's sideslip angle were identified and the NNs are designed to leverage data from previous time instants for estimation. Specifically, the first network uses lateral acceleration and steering wheel angle as inputs, while the second uses longitudinal acceleration, lateral acceleration, and yaw rate. Experimental tests conducted during maneuvers that provoke sideslip have demonstrated that, despite using a limited number of measurements, the estimators can accurately predict the sideslip angle.

4.4.1 | Choice of inputs and output

The selection of inputs was based on physical observations. Initially, fifteen potential inputs were identified:

- a_x, a_y: The longitudinal and lateral accelerations at the vehicle's center of gravity. These are related to the velocities v_x and v_y, as shown in Equation (4.21). Referring to the well-known bicycle model [89]:

$$\begin{aligned} \dot{v}_x &= a_x + \dot{\psi} \cdot v_y \\ \dot{v}_y &= a_y - \dot{\psi} \cdot v_x \end{aligned} \tag{4.21}$$

- $\dot{\vartheta}, \dot{\psi}, \delta$: The roll rate, yaw rate, and steering wheel angle of the vehicle. The sideslip angle becomes relevant during cornering maneuvers, much like these three variables. Additionally, the yaw rate is explicitly featured in Equation (4.21), which is used to calculate the sideslip angle.
- $\omega_{fl}, \omega_{fr}, \omega_{rl}, \omega_{rr}$: The rotational speeds of the four wheels. Under pure rolling and longitudinal motion conditions, the rotational speeds of the four wheels are given by:

$$\omega_{ij} = R \cdot v_x \tag{4.22}$$

Where R is the wheel radius. In real-world conditions, this equation does not hold strictly due to longitudinal slip. Moreover, the wheel radius at a

standstill differs from the actual radius when the wheel is in motion. During cornering, the four speeds also vary due to the differential in the driving wheels and the fact that the outer wheels trace a larger radius than the non-driving wheels. However, there remains a physical correlation between the vehicle's longitudinal velocity v_x and the rotational speed of the wheels.

$bp_{fl}, bp_{fr}, bp_{rl}, bp_{rr}, trq_{rl}, trq_{rr}$: brake pressures and motor torques. Brake pressure induces deceleration, reducing the longitudinal velocity v_x. Conversely, motor torque causes acceleration, increasing the longitudinal velocity v_x.

The output of the NN in this study is the sideslip angle. According to Equation (4.21), the sideslip angle depends on the longitudinal and lateral velocities at the vehicle's center of gravity. However, direct measurement of the lateral velocity at the center of gravity is not possible, as the sensor used—a Corrsys-Datron optical sensor—cannot be positioned there. Instead, the sensor was mounted at the front of the vehicle, as depicted in Figure 4.32.

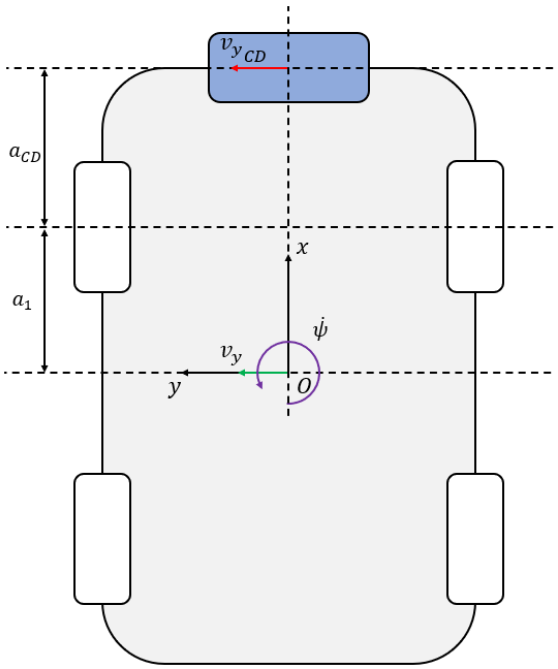


Figure 4.32 | Placement of the Corrsys-Datron sensor.

The longitudinal velocity measured by the Corrsys-Datron sensor corresponds to the longitudinal velocity at the center of gravity. The lateral velocity at the center of gravity can be calculated using geometric relationships and the vehicle's yaw rate:

$$v_y = v_{y_{CD}} - (a_1 + a_{CD}) \cdot \dot{\psi} \tag{4.23}$$

Where:

- $v_{y_{CD}}$ is the lateral velocity at the Corrsys-Datron sensor.
- a_1 is the longitudinal distance between the vehicle's center of gravity and the front axle.
- a_{CD} is the longitudinal distance between the front axle and the Corrsys-Datron sensor.
- $\dot{\psi}$ is the vehicle's yaw rate.

At the start and end of the maneuvers, the sideslip angle is close to 0°, and v_x is near $0\,\frac{km}{h}$. Since v_x appears in the denominator in Equation (3.1), this could lead to unrealistically large sideslip angles approaching $\pm 90°$, which could corrupt the network training. To prevent this, initial and final samples from each maneuver that meet the condition $v_x < 10\,\frac{km}{h}$ were discarded and are not included in the training or validation datasets.

4.4.2 | Data collection

Data were collected as part of experimental tests in the context of the ITEAM project by Flanders Make. The training, validation, and test datasets were created by performing various maneuvers with a Range Rover Evoque vehicle.

The maneuvers were performed at the Ford Lommel Proving Ground, with the track layout shown in Figure 4.33.

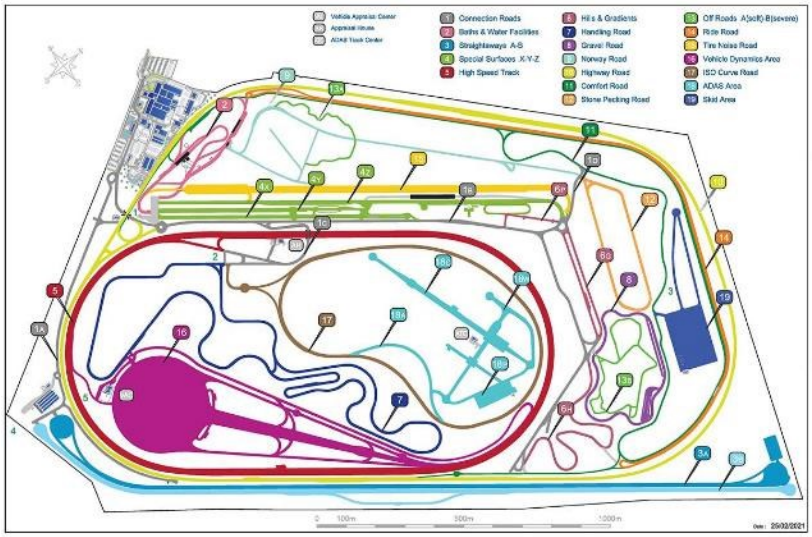


Figure 4.33 | Tracks of the Ford Lommel Proving Ground [90].

The maneuvers used to generate the training and validation datasets include:

- Constant radius.
- Step steer.
- Brake in turn.
- Sine with dwell.
- Slalom.
- Inner durability road.
- Gradients.
- High-speed oval.
- Acceleration tests.
- Brake tests.
- ISO double lane change.

A total of 85 maneuvers were executed across 11 different scenarios to capture the vehicle's behavior under a wide range of conditions. The density plot in Figure 4.34 illustrates the comprehensive coverage of the experimental data, ranging from longitudinal velocities of $10\,\frac{km}{h}$ to $120\,\frac{km}{h}$,

with lateral velocities varying between $10\frac{km}{h}$ and $3\frac{km}{h}$ as longitudinal velocity increases. The decision to exclude lower longitudinal velocities from the training and validation datasets was intentional, aimed at reducing noise interference in the reference sideslip angle calculation.

While higher velocities above $120\frac{km}{h}$ are theoretically possible, performing tight-radius turns at such speeds is impractical due to regulatory speed limits [91] and the lack of suitable road infrastructure for such maneuvers for safety reasons [92]. Furthermore, at speeds exceeding $120\frac{km}{h}$, the primary focus of control systems shifts towards enhancing longitudinal stability [93] rather than lateral stability.

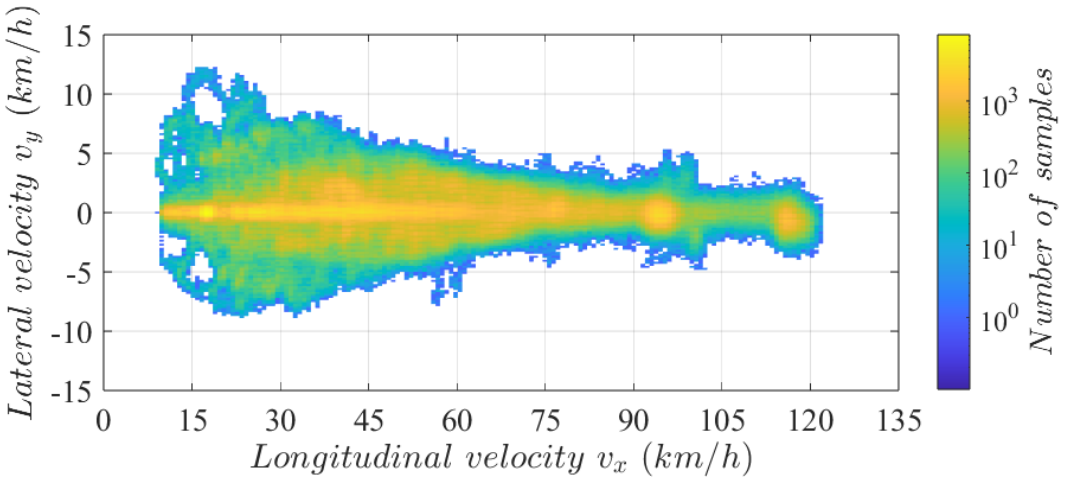


Figure 4.34 | Distribution of the samples belonging to the acquired data with respect to longitudinal and lateral velocities.

To improve convergence to the optimal set of model parameters, the data used as inputs for the network were normalized to have zero mean and unit standard deviation [73]. Additionally, all input-output samples were shuffled to enhance the generalization ability of the model [85]. The dataset was then split into training and validation sets in an $80-20$ ratio.

4.4.3 | NNs development

Feedforward Neural Networks (FFNNs) with time history were developed. The NN is structured as follows:

- Input layer;
- First hidden layer with ReLU activation function;
- Second hidden layer with ReLU activation function;
- Output layer.

To predict the sideslip angle at instant $k+1$, it was decided to use not only the inputs from instant k, but also those from previous time instants. This approach allows the NN to leverage more information to predict each sideslip angle value, even though the same set of measurements is used. Figure 4.11 illustrates this concept graphically. In each pair (i,j), the element i represents the physical quantity, while the element j represents the time instant.

Bayesian optimization was used to identify the set of hyperparameters that minimize the RMSE on the validation dataset and, as a result, determine the model parameters that minimize the cost function. The optimal hyperparameters identified through this optimization process are listed in Table 4.13.

Name	Range	Value
Initial learn rate	$[1^{-2} \div 1^{-1}]$	0.0456
Learn rate drop factor	$[0.1 \div 0.3]$	0.2461
Learn rate drop period	$[1 \div 2]$	1
Hidden units	$[500 \div 1500]$	607
L2 regularisation coefficient	$[1^{-6} \div 1^{-2}]$	0.0017
Dropout probability	$[0.1 \div 0.3]$	0.1196
Mini batch size	$[128, 256, 512, 1024]$	1024
Sample number	$[1 \div 10]$	10

Table 4.13 | Optimal hyperparameters for the original NN.

Following this, an analysis was conducted to determine which input variables had the most significant impact on the prediction of the sideslip angle. Since all inputs are normalized to have a zero mean, each input variable in the validation dataset was selectively set to its mean value (zero). Setting an input to its mean value effectively removes its influence on the predicted output, leading to an increase in the RMSE of the validation dataset. The inputs that cause the largest increase in RMSE are those that most significantly influence the prediction. Table 4.14 presents the hierarchy of physical quantities based on their impact on the prediction of the sideslip angle.

Input	RMSE of the validation dataset
$a_y = mean_{a_y}(= 0)$	1.6452
$\delta = mean_{\delta}(= 0)$	1.5536
$\dot{\psi} = mean_{\dot{\psi}}(= 0)$	1.2743
$bp_{ij} = mean_{bp_{ij}}(= 0)$	1.1756
$\omega_{ij} = mean_{\omega_{ij}}(= 0)$	1.1683
$a_x = mean_{a_x}(= 0)$	1.1423
$trq_{rj} = mean_{trq_{rj}}(= 0)$	1.1338
$\dot{\vartheta} = mean_{\dot{\vartheta}}(= 0)$	1.1271
No change in the inputs	1.1260

Table 4.14 | Hierarchy of most influencing physical quantities.

The results indicate that the most influential factors in predicting the sideslip angle are the lateral acceleration a_y and the steering wheel angle δ. A first neural network (NN1) was then trained using only these two variables. However, measuring these variables requires two different sensors: one for lateral acceleration [94] [95] and another for the steering wheel angle [96] [97]. Additionally, neither variable provides explicit information on the vehicle's longitudinal dynamics, despite longitudinal

velocity v_x appearing explicitly in Equation (3.1). For this reason, a second neural network (NN2) was trained using longitudinal acceleration a_x, lateral acceleration a_y, and yaw rate $\dot{\psi}$ as inputs. These quantities can be measured using a single sensor, typically an IMU, which is often already installed in modern vehicles. Figure 4.35 illustrates the structures of the two NNs.

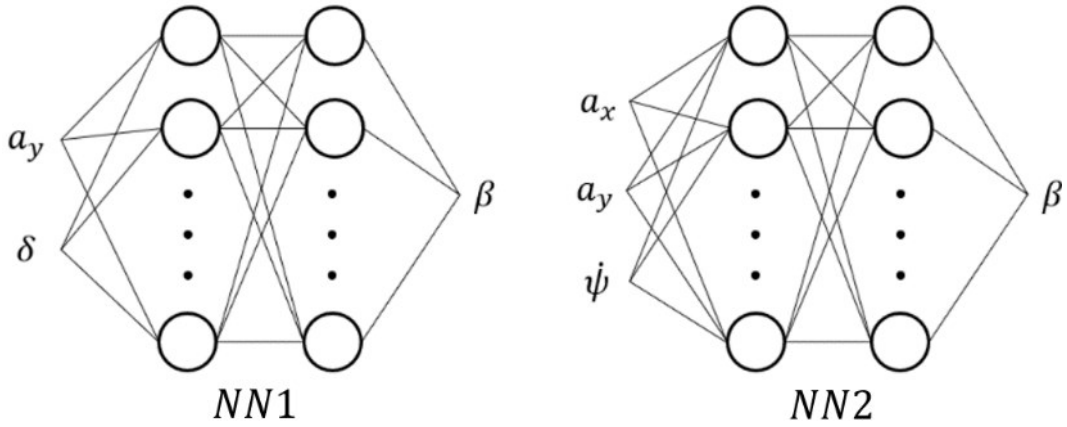


Figure 4.35 | NN architecture with evidence of inputs ($\boldsymbol{a_y}, \boldsymbol{\delta}$ for NN1, $\boldsymbol{a_x}, \boldsymbol{a_y}, \dot{\boldsymbol{\psi}}$ for NN2) and output ($\boldsymbol{\beta}$).

Both NNs were trained using Bayesian optimization, applying the same hyperparameters and variation ranges listed in Table 4.13. The identified hyperparameters and the RMSE on the validation dataset for both NN1 and NN2 are shown in Table 4.15.

Name	Value NN1	Value NN2
Initial learn rate	**0.0450**	**0.01746**
Learn rate drop factor	**0.1827**	**0.2461**
Learn rate drop period	**1**	**1**
Hidden units	**1446**	**1154**
L2 regularisation coefficient	**0.00073**	**0.000018**
Dropout probability	**0.1086**	**0.1098**
Mini batch size	**512**	**256**
Sample number	**9**	**9**
RMSE of the validation dataset	**1.1280°**	**1.1334°**

Table 4.15 | Optimal hyperparameters and RMSE of the validation dataset of NN1 and NN2.

The RMSE of the validation dataset for both the NN1 and NN2 is only slightly higher than that of the original NN (1.1260°) which used 15 physical input quantities. This analysis effectively reduced the number of inputs while maintaining a similar level of RMSE.

By calculating the MSE for each mini-batch at each iteration, the progress of the NN training was tracked. The NN was trained over an amount of six epochs. The decrease in training loss at subsequent iterations, the computed MSE on the last mini-batch (last Training Loss, FTL), and the resulting MSE on the validation dataset at the conclusion of training (Final Validation Loss, FVL) are all presented on a logarithmic scale in Figure 4.36 and 4.37. An epoch's end is indicated by vertical dotted black lines.

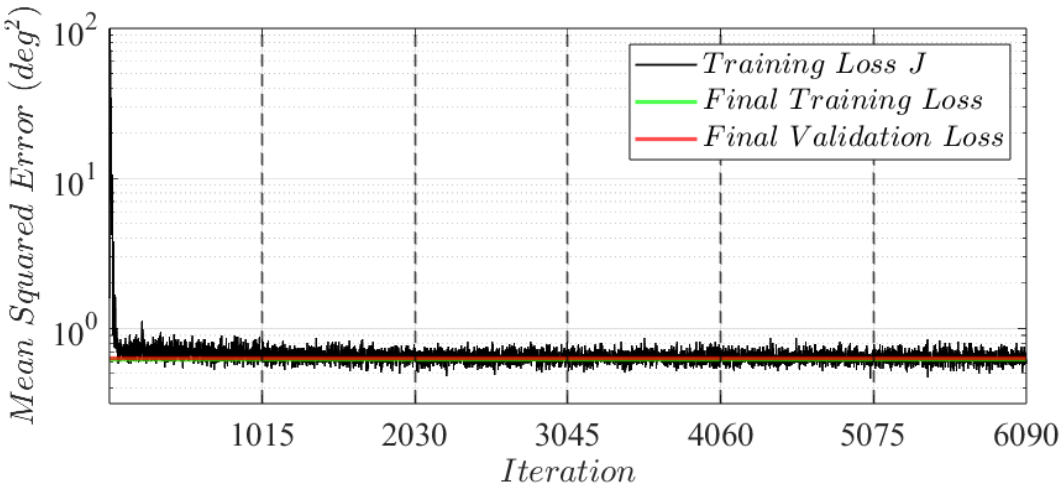


Figure 4.36 | Decrease in training loss at successive iterations (NN1).

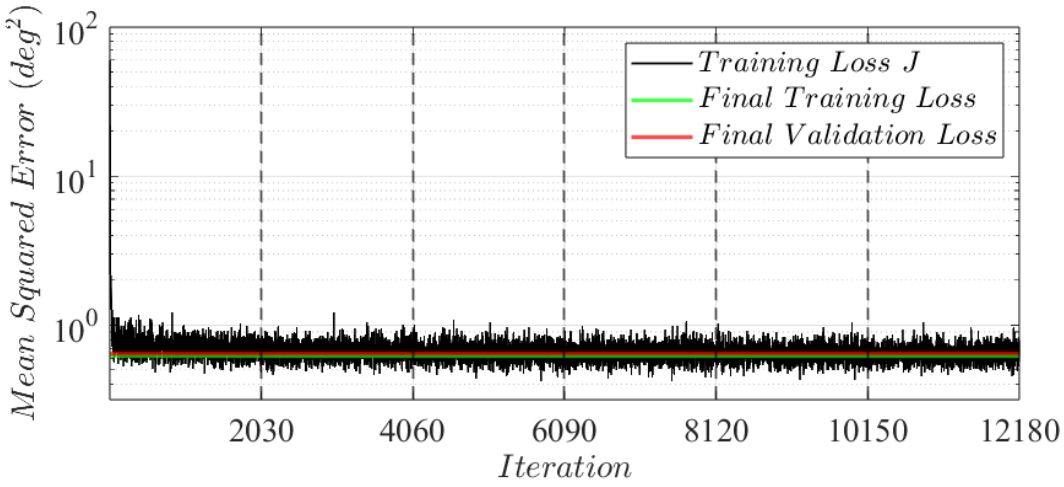


Figure 4.37 | Decrease in training loss at successive iterations (NN2).

NN1 and NN2 are not superior to each other in terms of estimation accuracy. Instead, they represent two viable alternatives that are more efficient compared to the original network which employed 15 different inputs, while maintaining comparable accuracy. The primary advantage of employing smaller NNs lies in the reduction of sensor requirements rather than solely in runtime optimization. The runtime is highly dependent on the hardware that could potentially enable even a complete network to be used. By employing fewer sensors, the proposed approach not only streamlines the NN architecture but also simplifies the overall system

complexity. This reduction extends beyond the NN itself to encompass various aspects such as wiring determinations, communication interface simplifications, and the mitigation of resampling requirements due to sensors operating at different sample times. While quantifying the exact extent of this overall improvement may prove challenging, its immediate intuitiveness underscores its significance.

4.5 | Measurement of unsprung mass displacement of road vehicles through a model-based virtual sensor

This work presents a virtual sensor designed to measure the vertical displacement of the unsprung masses in a road vehicle, which is crucial for controlling vehicle suspension systems. The virtual sensor is based on a 7-degree-of-freedom vehicle model, using the vehicle's measured longitudinal and lateral accelerations as inputs. The data used in the development of this paper were collected during a secondment at the Tenneco company conducted during the doctoral program.

4.5.1 | Estimation-oriented vehicle model

The vehicle has been modeled by applying D'Alembert's principle along the following axes:

- Longitudinal axis to vehicle x.
- Transverse axis to vehicle y.
- Vertical axis to vehicle z.
- Four vertical axes pass through the corners of the vehicle.

The first three axes form a levogyre orthogonal triad. The x-axis has a positive direction in the vehicle's direction of travel, the y-axis faces to the left, and the z-axis faces upwards. The origin of this reference system is at a

distance equal to h_G below the vehicle's center of gravity. The x and y axes are the roll and pitch axes of the vehicle. The four vertical axes related to the unsprung masses are positive upwards. The degrees of freedom associated with the vehicle model are given in Table 4.16. A scheme of the vehicle model is shown in Figure 4.38.

Symbol	Name
φ	Roll
ϑ	Pitch
z_o	Heave
$\tilde{z}_{FL}$	Front-left corner displacement
$\tilde{z}_{FR}$	Front-right corner displacement
$\tilde{z}_{RL}$	Rear-left corner displacement
$\tilde{z}_{RR}$	Rear-right corner displacement

Table 4.16 | List of degrees of freedom of the system.

Where:

$$\tilde{z}_{ij} = z_{1ij} - z_{2ij} \tag{4.24}$$

Specifically:

- z_{1ij} is the vertical displacement of the unsprung mass relative to corner ij only.
- z_{2ij} is the vertical displacement of the sprung mass relative to corner ij only.
- The index i refers to the front (F) or rear (R) axle of the vehicle.
- The index j refers to the left (L) or right (R) side of the vehicle.

According to the convention chosen, the displacement $\tilde{z}_{ij}$ is positive when the elastic and damping elements ij are in compression.

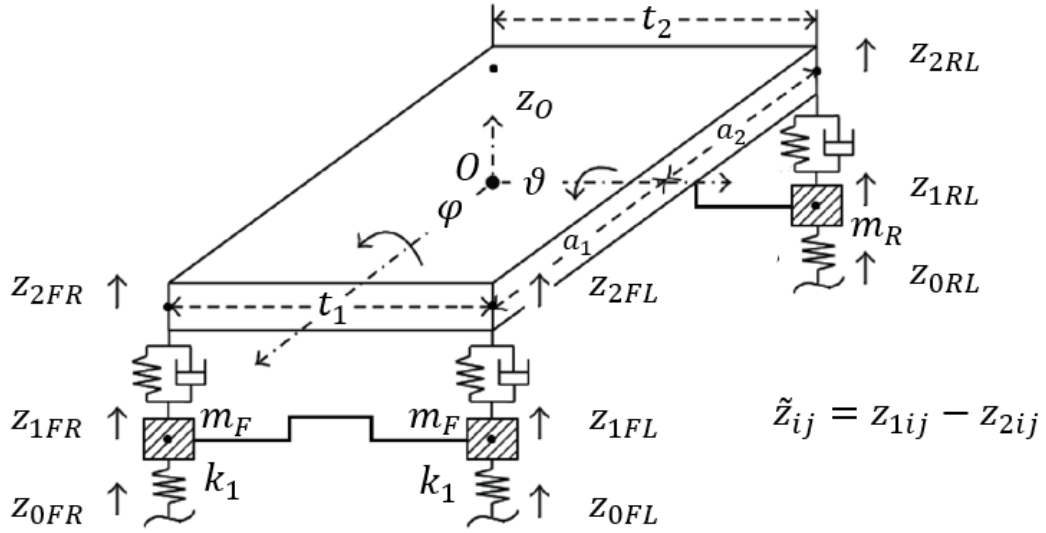


Figure 4.38 | Lumped-parameters diagram of the vehicle.

k_1 is the tire stiffness. z_{0ij} is the vertical displacement of the tire-road contact patch. Since it is impractical to know this value exactly [98], it will be assumed to be 0.

4.5.1.1 | Anti-roll torques

The vehicle has an active anti-roll torque for both the front and rear suspension. It has been modeled as shown in Figure 4.39. To achieve translational force equilibrium, each torque is replaced by the corresponding force pair. The torque values have been set as model inputs.

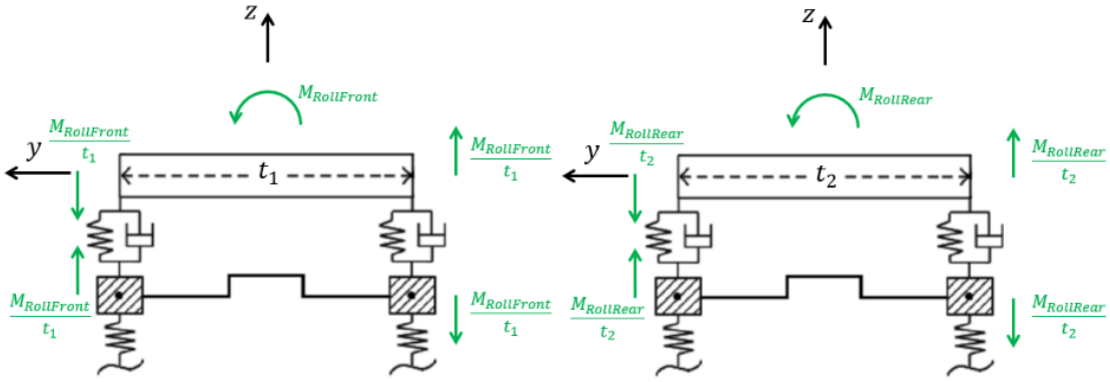


Figure 4.39 | Diagram of anti-roll bar torques and their equivalent force pairs.

4.5.1.2 | Damping forces

Damping forces are active for both the front and rear suspensions. The positive direction of the damping forces is shown in Figure 4.40.

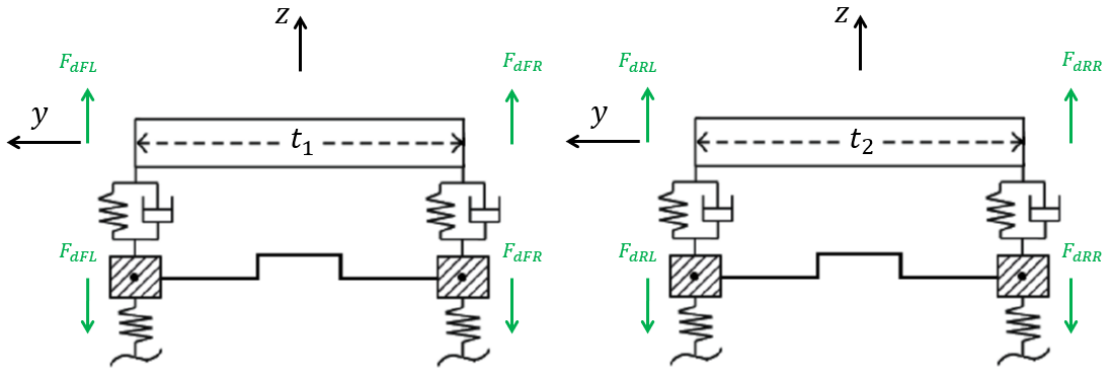


Figure 4.40 | Diagram of damping forces.

The damping force F_{dij} depends on the control electric current i_{dij} and the relative velocity between the sprung and unsprung mass, computed as follows:

$$\dot{\tilde{z}}_{ij} = \dot{z}_{1ij} - \dot{z}_{2ij} \tag{4.25}$$

The lookup table returns force values in the ranges $-1109.1 \div 1373.2\ N$ (front suspension) and $-1183.3 \div 1115.6\ N$ (rear suspension) for input currents in the range $0 \div 1.6\ A$ and relative velocities in the range $-2 \div 2\ \frac{m}{s}$.

The damping force F_{dij} applied in the model does not coincide with the force applied by the damping actuator, just as the speed $\dot{\tilde{z}}_{ij}$ of the model does not coincide with the speed of the actuator. The Motion Ratio MR_{ij} considers the different points of application of damping forces on the real vehicle and the vehicle model. Regarding the first one, damping forces are applied at the actuators, while concerning the second one, the previously mentioned forces are applied at the corners of the vehicle. MR_{ij} is equal to the ratio between the length of the suspension arm AL_{ij} and the distance

WL_{ij} between the pivot point of the arm and the wheel. The geometric meaning of these quantities is shown in Figure 4.41.

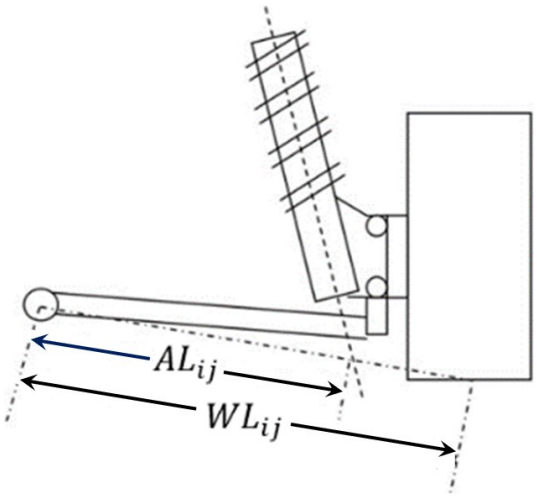


Figure 4.41 | Length of the suspension arm $\boldsymbol{AL_{ij}}$ and distance $\boldsymbol{WL_{ij}}$ between the pivot point of the arm and the wheel.

For the damping force applied in the model to be equivalent to that applied by the actual actuator, conversions must be made for the velocity input to the lookup table and the force output from the lookup table. The input velocity and the output force are relative to the actual actuator. Applying the lever rule [99], the velocity $\dot{\tilde{z}}_{ij}$ as input to the lookup table must be multiplied by the factor MR_{ij} before interpolating the values from the lookup table. The output force must be multiplied by MR_{ij} to obtain the force F_{dij}.

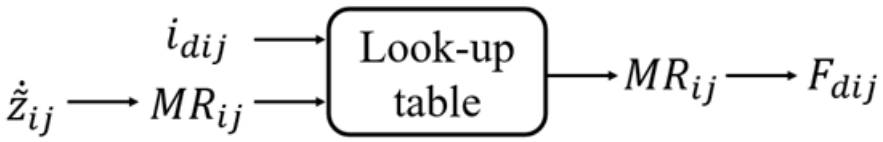


Figure 4.42 | Damping force calculation scheme with look-up table input and output representation.

4.5.1.3 | Spring forces

The springs within the front and rear suspensions are passive and pneumatic. The positive direction of the spring forces is shown in Figure 4.43.

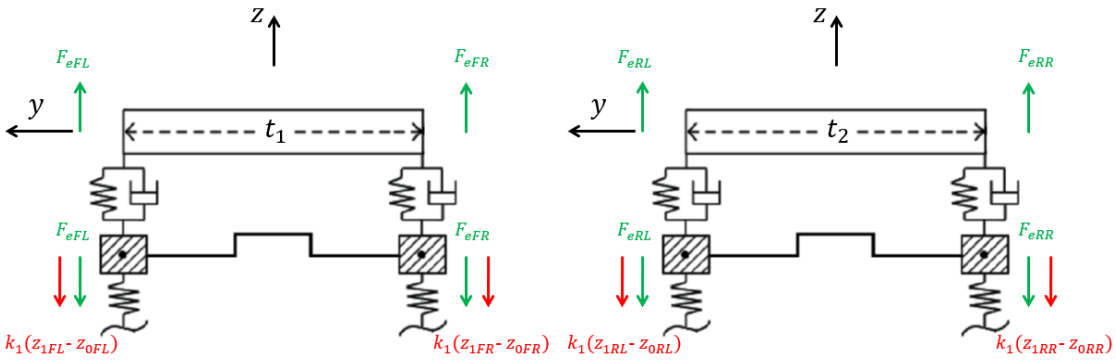


Figure 4.43 | Diagram of spring and tire forces.

The net elastic force F_{eij} depends on the relative displacement between the sprung and unsprung masses. The spring in the suspension is pneumatic. In this type of spring [100], the force F_{eij} increases exponentially as the difference $\tilde{z}_{ij} = z_{1ij} - z_{2ij}$, spring in compression, increases. Furthermore, this force takes on a zero value when compression is zero.

Equation (4.26) incorporates the desired characteristics:

$$F_{eFj} = \alpha_F(\beta^{\gamma\tilde{z}_{Fj}} - 1)$$
$$F_{eRj} = \alpha_R(\beta^{\gamma\tilde{z}_{Rj}} - 1) \quad (4.26)$$

The identification of the numerical values assumed by the parameters in Equation (4.26) is explained in the Section 4.5.1.5.

Figure 4.44 shows the elastic force F_{eij} at a displacement $\tilde{z}_{ij}$. The diagram is plotted in the range of variation from $-100\ mm$ to $100\ mm$.

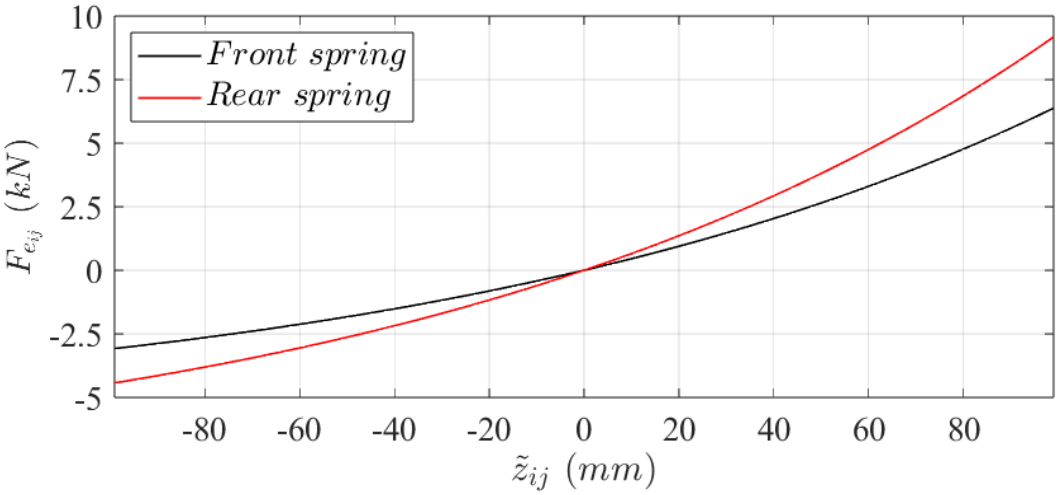


Figure 4.44 | Force exerted by the spring element $\boldsymbol{F_{eij}}$ with respect to displacement $\boldsymbol{\tilde{z}_{ij}}$.

The elastic force F_{eij} generated by tire ij has been considered linearly dependent on $z_{1ij} - z_{0ij}$. The tire stiffness coefficient $k_1 = 0.1882\ \frac{kN}{mm}$.

4.5.1.4 | Equilibrium Equations

The dynamical Equations related to the 7-degree-of-freedom vehicle model are explained in the present Section. The vehicle model is defined by the following equilibrium Equations:

- Around x

$$I_x\ddot{\varphi} = (F_{dFL} + F_{eFL})\frac{t_1}{2} - (F_{dFR} + F_{eFR})\frac{t_1}{2} + (F_{dRL} + F_{eRL})\frac{t_2}{2} - (F_{dRR} + F_{eRR})\frac{t_2}{2} - M_{RollFront} - M_{RollRear} + Mh_G a_y \quad (4.27)$$

- Around y

$$I_y\ddot{\vartheta} = -(F_{dFL} + F_{eFL})a_1 - (F_{dFR} + F_{eFR})a_1 + (F_{dRL} + F_{eRL})a_2 + (F_{dRR}+F_{eRR})a_2 - Mh_G a_x \quad (4.28)$$

- Along z

$$M\ddot{z}_O = F_{dFL} + F_{eFL} + F_{dFR} + F_{eFR} + F_{dRL} + F_{eRL} + F_{dRR} + F_{eRR} \quad (4.29)$$

- Body-wheel coordinate link

$$\begin{aligned} z_{2FL} &= z_O + \frac{t_1}{2}\varphi - a_1\vartheta \\ z_{2FR} &= z_O - \frac{t_1}{2}\varphi - a_1\vartheta \\ z_{2RL} &= z_O + \frac{t_2}{2}\varphi + a_2\vartheta \\ z_{2RR} &= z_O - \frac{t_2}{2}\varphi + a_2\vartheta \\ z_{1ij} &= z_{2ij} + \tilde{z}_{ij} \end{aligned} \tag{4.30}$$

- Along z – front left wheel

$$\begin{aligned} &m_{FL}\left(\ddot{\tilde{z}}_{FL} + \ddot{\tilde{z}}_O + \frac{t_1}{2}\ddot{\varphi} - a_1\ddot{\vartheta}\right) - \frac{M_{RollFront}}{t_1} \\ &= -F_{dFL} - F_{eFL} - k_1\left(z_O + \frac{t_1}{2}\varphi - a_1\vartheta + \tilde{z}_{FL} - z_{0FL}\right) \end{aligned} \tag{4.31}$$

- Along z – front right wheel

$$\begin{aligned} &m_{FR}\left(\ddot{\tilde{z}}_{FR} + \ddot{\tilde{z}}_G - \frac{t_1}{2}\ddot{\varphi} - a_1\ddot{\vartheta}\right) + \frac{M_{RollFront}}{t_1} \\ &= -F_{dFR} - F_{eFR} - k_1\left(z_O - \frac{t_1}{2}\varphi - a_1\vartheta + \tilde{z}_{FR} - z_{0FR}\right) \end{aligned} \tag{4.32}$$

- Along z – rear left wheel

$$\begin{aligned} &m_{RL}\left(\ddot{\tilde{z}}_{RL} + \ddot{\tilde{z}}_O + \frac{t_2}{2}\ddot{\varphi} + a_2\ddot{\vartheta}\right) - \frac{M_{RollRear}}{t_2} \\ &= -F_{dRL} - F_{eRL} - k_1\left(z_O + \frac{t_2}{2}\varphi + a_2\vartheta + \tilde{z}_{RL} - z_{0RL}\right) \end{aligned} \tag{4.33}$$

- Along z – rear right wheel

$$\begin{aligned} &m_{RR}\left(\ddot{\tilde{z}}_{RR} + \ddot{\tilde{z}}_O - \frac{t_2}{2}\ddot{\varphi} + a_2\ddot{\vartheta}\right) + \frac{M_{RollRear}}{t_2} \\ &= -F_{dRR} - F_{eRR} - k_1\left(z_O - \frac{t_2}{2}\varphi - a_2\vartheta + \tilde{z}_{RR} - z_{0RR}\right) \end{aligned} \tag{4.34}$$

Within the previously presented Equations, a_x and a_y represent the longitudinal and lateral accelerations of the centre of gravity and are measured with an IMU.

The developed model is characterized by lumped parameters. The modeling was developed using this approach for several reasons:

- Purpose of the analysis: the main objective is a concise and easily interpretable representation of the suspension to focus on the fundamental dynamics. The use of a multibody model would have introduced additional complexity not relevant to the specific objectives of this study.
- Computational resource limitations: implementation of a multibody model requires significant computational resources, especially in complex scenarios. A lumped-parameters model was preferred to ensure that the analysis could be conducted efficiently.
- Experimental validation: data to test the estimator indicate that the lumped-parameters model offers an accurate representation of suspension dynamics.

4.5.1.5 | Identification of unknown parameters

The parameters presented in Section 4.5.1.3 and the height of the center of gravity h_G have been identified through a parametric identification procedure. Several lane change maneuvers and one lap on track 7 of the Ford Lommel Proving Ground were performed on track 10. Track 7 is 4.3 km long and combines several transient and steady-state maneuvers, including elevation changes.

Figure 4.45 shows, schematically, the known quantities (red and green).

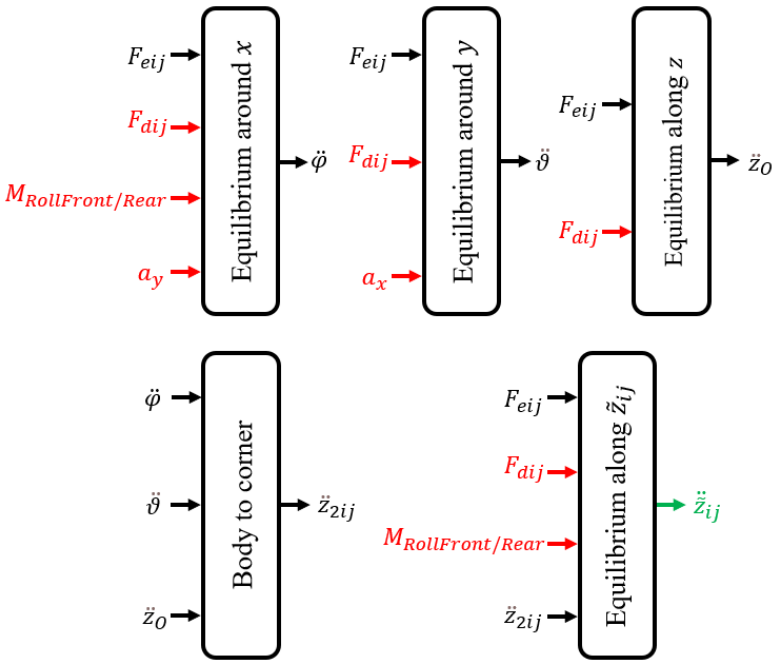


Figure 4.45 | Simplified graphical representation of the estimator subsystems with specifications of known (red) and estimated (black) quantities. The estimated acceleration of the unsprung masses relative to the chassis is indicated in green.

The anti-roll torques $M_{RollFront}$ and $M_{RollRear}$ are known because they are applied to the vehicle, independently of the displacements estimated with the virtual sensor. The damping forces F_{dij} are partially known because the relative velocities between the sprung and unsprung masses are computed through the proposed virtual sensor, while the applied control electric currents are known. The longitudinal and lateral accelerations a_x and a_y are measured via an IMU and preliminarily processed. For a correct identification of the unknown parameters, as well as a correct subsequent estimation of the system states, it is necessary to remove the DC Gain of the lateral and longitudinal accelerations and to convert them from signals expressed in multiples of the gravitational acceleration g to signals expressed in $\frac{m}{s^2}$. The presence of an offset in the measurements would lead to a rapid drift of the measured signals. The use of accelerations not expressed in $\frac{m}{s^2}$ would lead to the identification of physically invalid

parameters, and the estimation of displacements would be different from that expected. The initial offset was removed by removing the acceleration value measured with the vehicle in stationary conditions. The unit conversion was performed by multiplying the original signals by 9.80665, the gravitational acceleration. High-pass and low-pass filters were not used as, although they would allow the removal of high- and low-frequency components not characteristic of the signal of interest, they would result in unacceptable delays in the estimation of displacements and in the application of the estimator to suspension control systems that require rapid processing and responses to external inputs to preserve safety and ride comfort. Furthermore, it must be noted that the accelerations always oscillate around the null value, as do the states, since no offset has been introduced into the model. Therefore, any integration errors of $\ddot{\tilde{z}}_{ij}$ due to measurement noise are automatically compensated over time. From the knowledge of the quantities highlighted in red in Figure 4.45 and the physical model, it is possible to calculate $\ddot{\tilde{z}}_{ij}$ and then, by double integration, $\tilde{z}_{ij}$. The actual relative displacements between the unsprung masses and the sprung mass have been measured by means of a sensor placed inside each wheel. Subsequently, the values of the parameters a_F, a_R, b, c and h_G for minimizing the mean square error between the measurements taken and the $\tilde{z}_{ij}$ signals obtained through the dynamic model have been identified.

The identification procedure consists of the following steps:

1. Definition of the quadratic cost function:

$$J = \frac{1}{2} \sum_{i=F,R} \sum_{j=L,R} \sum_{k=1}^{n} (\hat{\tilde{z}}_{ij_k}(\alpha_F, \alpha_R, \beta, \gamma, h_G) - \tilde{z}_{ij_k})^2 \qquad (4.35)$$

- $\hat{\tilde{z}}_{ij_k}$ is the estimated displacement, parameterized with respect to h_G, related to the unsprung mass ij, obtained at the k-th time instant.
- $\tilde{z}_{ij_k}$ is the experimental measurement related to the unsprung mass ij, obtained at the k-th time instant.
- n is the number of samples.

2. Random selection from a normal Gaussian distribution of the initial values of $\alpha_F, \alpha_R, \beta, \gamma$ and h_G.
3. Calculation of the Gradient of the Cost Function. The gradient of the cost function with respect to the parameters α_F, α_R, β, γ and h_G, denoted as $\nabla J(\alpha_F,\ \alpha_R,\ \beta,\ \gamma,\ h_G)$, is a vector that holds the cost function's partial derivatives with regard to each parameter. This vector indicates the direction and magnitude of the maximum decrease in the cost function with respect to each parameter.
4. Update Parameters via Gradient Descent [77].
5. Iteration. Steps 3 and 4 were repeated until the cost function converged to a minimum value.
6. Convergence verification to a global minimum. Steps 3, 4, and 5 were repeated 100 times with different initial values to find the global minimum among all minima.

Table 4.17 shows the parameters related to Equation (4.26) identified through the parametric identification procedure.

Symbol	Description	Value
α_F	Vertical front scale factor	5955
α_R	Vertical rear scale factor	8166
β	Exponential form factor	4.796
γ	Horizontal scale factor	4.691
h_G	Distance of the CoG* from the origin O of the reference system	0.561 (m)

**CoG: Centre of Gravity*

Table 4.17 | Air spring force coefficients and distance of the CoG from the origin O

Table 4.18 shows the Root Mean Square Error (RMSE) of the displacement for the four unsprung masses.

	$\tilde{z}_{FL}$	$\tilde{z}_{FR}$	$\tilde{z}_{RL}$	$\tilde{z}_{RR}$
RMSE	0.0098868	0.0090681	0.0086620	0.0076287

Table 4.18 | RMSE between predicted and reference displacements of maneuvers carried out for parameter identification.

4.6 | Neural Network-based virtual measurement of road vehicle wheel displacements

This study introduces a virtual sensor designed to measure the vertical displacement of a road vehicle's wheels, specifically aimed at enhancing the control of the vehicle's suspension systems. The virtual sensor utilizes a multi-output NN to predict the displacement of all four wheels simultaneously. Inputs to the network include longitudinal, lateral, and vertical accelerations from an Inertial Measurement Unit (IMU), along with anti-roll torques from both front and rear anti-roll bars, and currents in the actuators of active suspensions.

4.6.1 | Choice of inputs and output

A vehicle reference system is defined with the origin at the center of gravity, the x-axis in the direction longitudinal to the vehicle, the y-axis in the direction lateral to the vehicle, and the z-axis pointing upwards. The

three axes are oriented in a way to make an orthogonal levorotatory triad. The subscript i denotes the axle, front f or rear r, and the subscript j the left vehicle side l or right vehicle side r. The selection of inputs for the NN is rooted in physical considerations. Nine inputs are identified, each crucial for capturing various aspects of the vehicle's behavior:

- *Longitudinal acceleration* a_x: this input reflects weight transfer during acceleration or deceleration, impacting the compression or expansion of the suspension at each axle; lateral acceleration a_y: Lateral forces during cornering result in asymmetrical load distribution, affecting the vertical displacement of wheels on the outside and inside of curves; Vertical acceleration a_z: changes in vertical acceleration correspond to suspension compression or extension, influencing the vertical displacement of wheels relative to the chassis.
- *Currents in the damping actuators* i_{ij}: actuators at each wheel respond to vehicle dynamics by applying forces to adjust the suspension, affecting the vertical position of the wheels.
- *Antiroll moments* M_i: these moments counteract chassis roll during cornering, redistributing load between the inner and outer suspensions to maintain vehicle stability.

The four outputs of the NN are the vertical displacements of the wheels w_{ij} with respect to the chassis, evaluated at the wheels. These displacements are positive if the wheel approaches the chassis and negative if it moves away. The reference values were measured by means of a physical sensor placed on each wheel.

4.6.2 | Data collection

The data used in the development of this paper were collected during a secondment at the Tenneco company conducted during the doctoral program. The data were acquired by performing on the Ford Lommel Proving Ground acceleration, deceleration, and steering maneuvers, which most stress the vertical displacements of the wheels, in a way that covers the normal operating conditions of a road vehicle. Manoeuvres were carried out for a total of 3 hours, which allowed a low computational burden in training while maintaining adequate generalization.

4.6.3 | Neural network development

The NN consists of an input layer, two hidden layers with ReLU activation functions, and an output layer, as shown in Figure 4.46.

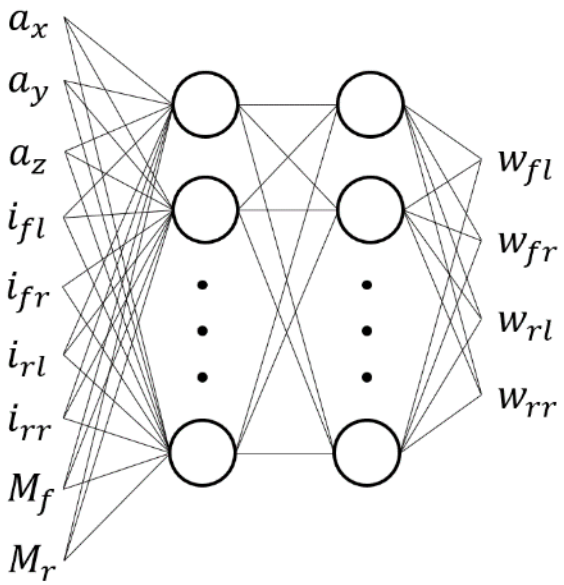


Figure 4.46 | NN architecture with evidence of inputs $(\boldsymbol{a_x}, \boldsymbol{a_y}, \boldsymbol{a_z}, \boldsymbol{i_{fl}}, \boldsymbol{i_{fr}}, \boldsymbol{i_{rl}}, \boldsymbol{i_{rr}}, \boldsymbol{M_f}, \boldsymbol{M_r})$ and outputs $(\boldsymbol{w_{fl}}, \boldsymbol{w_{fr}}, \boldsymbol{w_{rl}}, \boldsymbol{w_{rr}})$.

In order to increase the amount of information used to estimate the output at instant $k+1$ without increasing the number of measurements

required, samples from m previous instants were also used as input according to the scheme shown in Figure 4.11.

The hyper-parameters were tuned by Bayesian optimization according to the ranges shown in Table 4.19.

Name	Range	Type	Value
Initial learn rate	$[10^{-2} \div 10^{-1}]$	Real	0.8983
Hidden units	$[500 \div 1500]$	Integer	873
L2 regularisation coefficient	$[10^{-6} \div 10^{-2}]$	Real	0.0043
Dropout probability	$[0.1 \div 0.3]$	Real	0.1324
Mini batch size	$[128, 256, 512, 1024]$	Integer	512
Sample number	$[1 \div 20]$	Integer	19

Table 4.19 | Range of variation of hyper-parameters and identified values.

Using mid-range hardware, the estimator is able to return the four outputs with a sample time of $0.01\,s$, making a real-time application feasible.

4.7 | Enhancement of the wheel vertical displacement estimation in road vehicles through integration of model-based estimator with artificial intelligence

In the automotive industry, accurately estimating wheel displacements is crucial for optimizing vehicle suspension systems. Traditional model-based approaches often struggle to predict these displacements accurately due to the complex dynamics of road-vehicle interactions. This study proposes integrating a multi-output NN that compensates for estimation errors inherent in model-based approaches.

The physical model used in this study is shared with the one presented in Section 4.5.

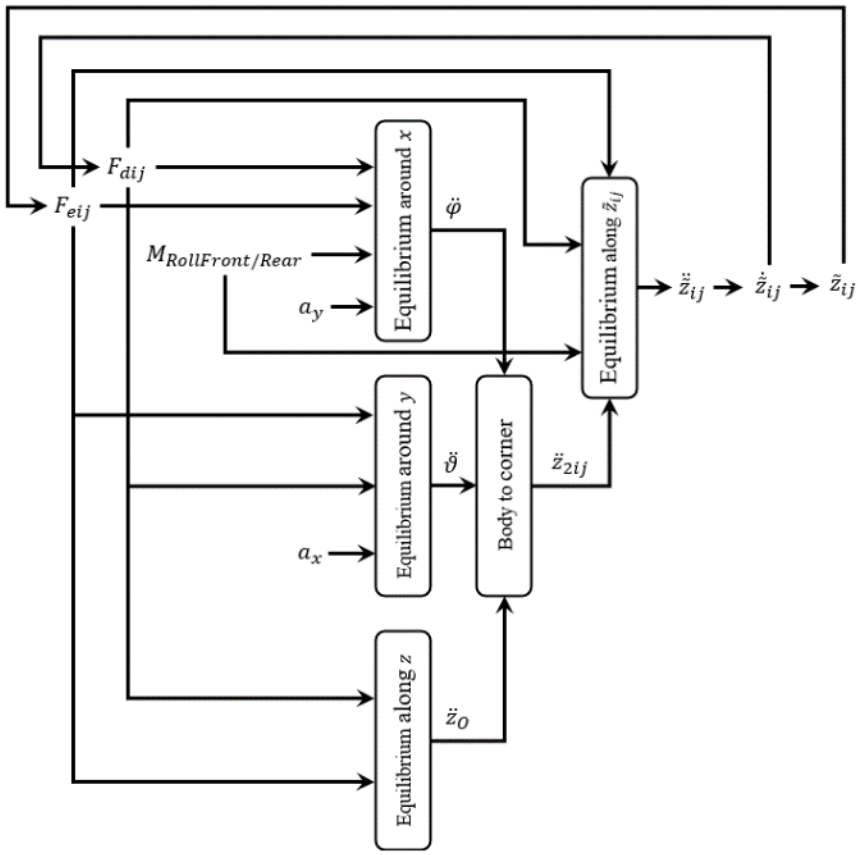


Figure 4.47 | Schematic representation of the model-based estimator of wheel displacements.

4.7.1 | Corrective neural network

The model-based estimator cannot accurately estimate wheel displacement relative to the vehicle body when this displacement is caused by road inputs, as it lacks information about the road profile. To address this limitation, a multi-output NN was integrated to work in parallel with the model-based estimator, compensating for the missing road input data. The NN leverages vertical acceleration information to estimate the suspension displacement resulting from road inputs. It can identify nonlinear relationships between these inputs and wheel displacements

without requiring a physical model, which is often complex due to the unpredictable nature of road inputs.

Initially, the NN was trained using forces and torques from the actuators and acceleration data from an IMU along three axes as inputs, with the output being the displacement of the four wheels. This training was conducted using experimental data from the same vehicle for which the model-based estimator was developed. Once properly trained, the NN learns the intrinsic correlations between the inputs without needing a physical model of the system.

The inputs to the trained NN were normalized to have a mean of zero and a standard deviation of one. The key idea of this study is that by setting some of the inputs to zero—effectively their mean value—the NN can isolate the effects of the remaining inputs based on the relationships it learned during training. In this case, the model-based estimator cannot provide accurate wheel displacement estimates due to road inputs, but this information is captured in the vertical acceleration data, which is not used by the model-based estimator but is included in the NN's inputs.

NNs learn to map inputs to outputs based on patterns in the training data, with the network's weights adjusted to minimize errors between predicted and actual outputs. When input features are normalized, setting them to their mean value removes their contribution from the network’s calculations. Consequently, the NN can only rely on the non-zero features to make predictions, limiting its ability to fully use the learned relationships involving the zeroed-out features.

In the context of model-based estimators, the input features are crucial for accurate predictions. However, the estimator cannot account for the influence of road inputs on wheel displacement because it lacks the

necessary input variability. As a result, predictions rely solely on other inputs, which do not provide sufficient information for accurate estimation.

Road inputs cause vertical movement of the vehicle's center of gravity, leading to measurable vertical acceleration. These inputs do not cause horizontal motion and do not result in longitudinal or lateral accelerations. Since the model-based estimator uses longitudinal and lateral accelerations—which do not contain road input information—it cannot estimate vertical wheel displacement due to road inputs on its own. To overcome this, all inputs to the NN, except for vertical acceleration, were set to zero (the mean value of the normalized inputs). The output from the NN, representing the displacement due to road inputs, was then added to the estimate obtained from the model-based estimator.

The overall displacement is calculated using the following Equation:

$$\tilde{z}_{ij} = \tilde{z}_{ij}^{MB} + \tilde{z}_{ij}^{NN} \tag{4.36}$$

Where:

- $\tilde{z}_{ij}^{MB}$ is the wheel displacement ij due to the model-based estimator.
- $\tilde{z}_{ij}^{NN}$ is the wheel displacement ij due to the NN.

The overall estimator is shown in Figure 4.48.

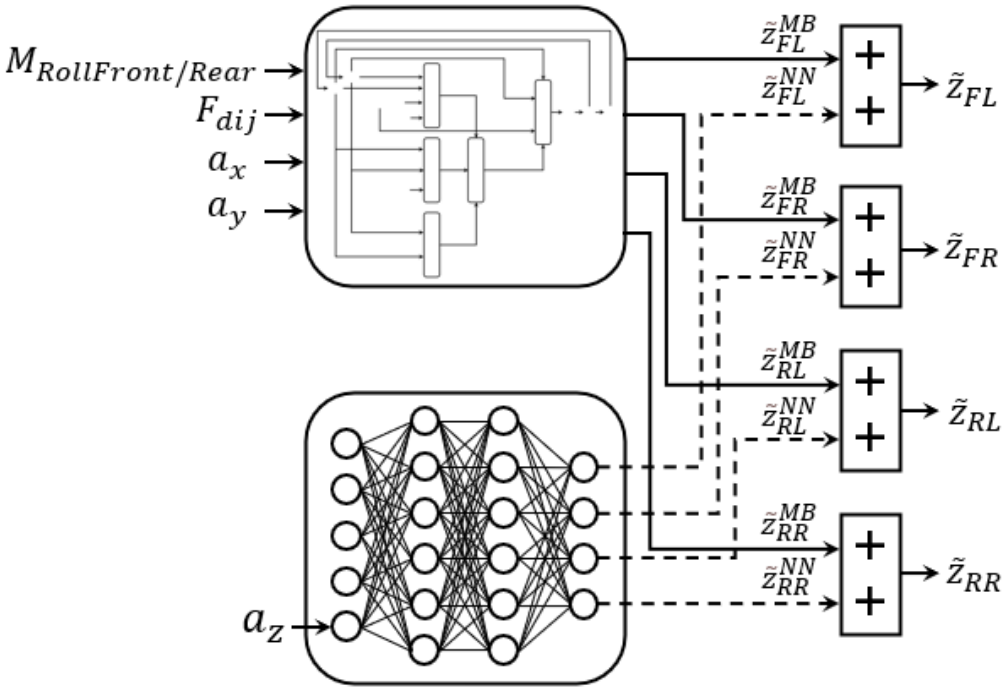


Figure 4.48 | Schematic representation of the overall estimator of wheel displacements.

4.7.2 | Neural network development

The NN used is a feedforward NN [101] with two hidden layers and a ReLU activation function [102].

To effectively train an NN, three distinct datasets are essential: the training dataset, used to adjust model parameters; the validation dataset, employed to fine-tune hyperparameters; and the test dataset, which evaluates the network's performance independently. The data used in the development of this paper were collected during a secondment at the Tenneco company conducted during the doctoral program. The data were collected by performing natural maneuvers in line with state-of-the-art indications. The objective was to develop a significant test scenario that isn't specific to a single car and aligns the estimator with emerging patterns in human-machine decision-making that are indicative of intelligent cars [103] [104]. The training and validation sets were acquired by asking the driver to perform normal driving on bends and roads with bumps in order to

stress the suspension and then dividing the samples, randomly shuffled, according to an 80-20 proportion. The test set consisted of different data, in order to verify the absence of overfitting of the training data, in which the driver was asked to stress the suspension again. The NN demonstrates proficiency by performing effectively on both the training and validation sets, while also fitting the test set accurately, thereby guarding against overfitting.

Hyperparameters are essential to the NN's learning process because they affect the values that the model parameters learn. These parameters are set before training and significantly impact the model's performance [22]. Optimizing hyperparameters, known as hyperparameter optimization, is essential for effective model training [23] [24]. In this study, several hyperparameters were optimized: initial learning rate [26], hidden units [30], L2 regularization coefficient [32], dropout probability [33], and mini-batch size [34], number of samples.

To estimate the wheel displacement at instant $k+1$, it was decided to consider the inputs from both instant k and prior instants. With this approach, the NN can use additional data for the prediction without additional measures. In each pair (i, j), the time instant is represented by the element j, while the measure is represented by the element i. For each input quantity used in the output prediction, m is the number of samples and constitutes the hyperparameter to be tuned. n, which in this case is equal to 9, is the number of input quantities. In Figure 4.11, the concept is depicted.

The variation ranges of the hyper-parameters and the identified value are shown in Table 4.20.

Name	Range	Type	Value
Initial learn rate	$[10^{-2} - 10^{-1}]$	Real	0.0320

Hidden units	$[100 - 1000]$	Integer	113
L2 regularization coefficient	$[10^{-6} - 10^{-2}]$	Real	$2.7 \cdot 10^{-5}$
Dropout probability	$[0.1 - 0.3]$	Real	0.291
Mini batch size	[128,256,512,1024]	Integer	128
Number of samples	$[1 - 50]$	Integer	19

Table 4.20 | Range of variation of hyperparameters and identified values.

Figure 4.49 shows the decrease in Training Loss and Validation Loss during NN training.

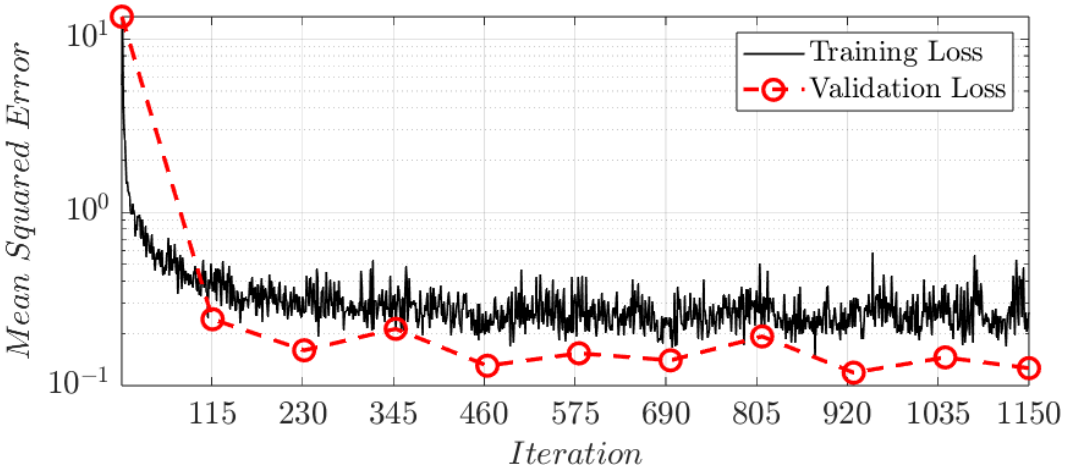


Figure 4.49 | Decrease of MSE during NN training.

4.8 | Addressing integration challenges in vehicle roll and pitch estimation using Neural Network

This study introduces a novel multioutput estimator that utilizes longitudinal and lateral accelerations to simultaneously estimate a vehicle's roll and pitch angles. Training data were obtained by performing laps around the Nürburgring track in both directions and a slalom maneuver using CarMaker software. OU noise was applied to the accelerations to simulate real-world signals. Subsequently, the NN was tested during a double lane change maneuver, with OU noise added to the input signals. A comparison between the NN's estimations and those obtained through the

integration of roll rate and pitch rate signals demonstrates the advantages of the NN approach. By employing algebraic calculations instead of signal integration, the NN provides consistent estimations over time.

4.8.1 | Choice of inputs and output

The model's inputs were carefully chosen to reflect the real-world physical aspects of the problem:

- *Longitudinal acceleration* a_x: When the vehicle accelerates forward, the force of inertia tends to shift the weight of the vehicle towards the rear. This weight shift causes the vehicle to pitch, with the front end rising and the rear end lowering. When braking, the opposite happens. The weight shifts towards the front, causing the vehicle to pitch, with the front end lowering and the rear end lifting.
- *Lateral acceleration* a_y: When the vehicle accelerates sideways in a curve, the force of inertia tends to shift the weight of the vehicle towards the outside of the curve. This shift in weight causes the vehicle to roll, with the outer part of the vehicle falling and the inner part rising.

The two outputs of the NN are the roll φ and pitch ϑ of the vehicle, calculated as rotation about the x and y axes of the reference system SyV.

4.8.2 | Data collection

The data for training and validation of the NN were acquired by carrying out in CarMaker two laps of the Nürburgring track, shown in Figure 4.21, in the opposite direction and a slalom maneuver between ten obstacles placed 18 m apart.

To achieve greater robustness of the estimator with respect to noise, an OU noise was added to the accelerations used as input. White noise is a

stationary process with independent and identically distributed increments. In practice, this means that its values change randomly and abruptly, without any memory of the past. OU noise is a stochastic process with a memory component. Its values change gradually and continuously, with a tendency to return towards a predetermined mean. This behavior makes OU noise more realistic for modeling many natural and engineering phenomena, which often exhibit some inertia and regression toward the mean. The superimposed noise has a standard deviation rate of 0.2 and a mean attraction constant of 5. Figure 4.50 shows the noise over a time interval of 5 s.

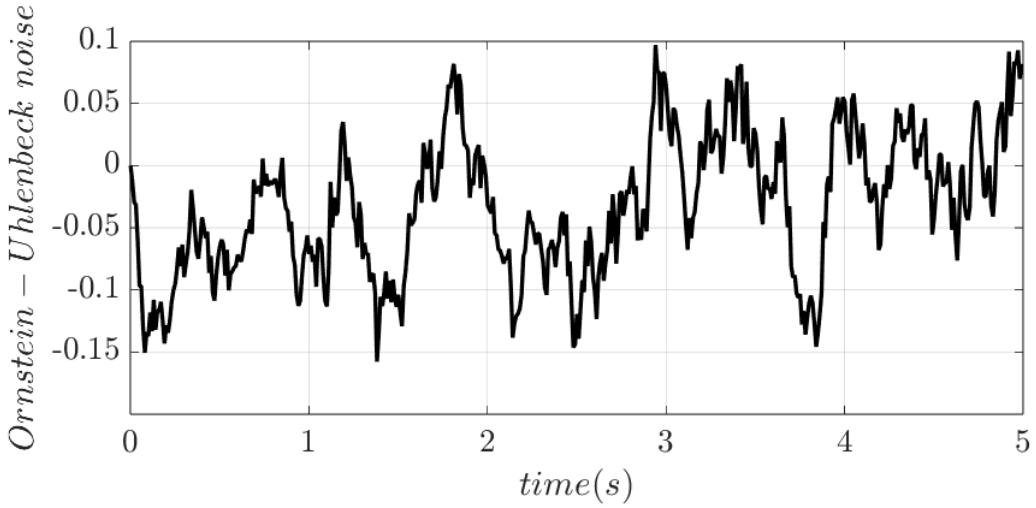


Figure 4.50 | OU noise using the training of the RL algorithm.

4.8.3 | Neural network development

The NN consists of an input layer, two hidden layers with swish activation functions, and an output layer.

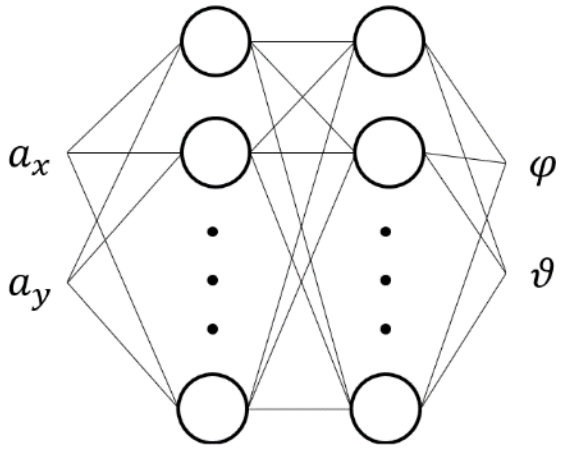


Figure 4.51 | NN architecture with evidence of inputs (a_x, a_y) and outputs (φ, ϑ).

The swish activation function is shown in (1).

$$f(x) = \frac{x}{1 - e^{-x}} \tag{4.37}$$

offers several advantages over other commonly used activation functions in NNs. Swish has a non-monotonic gradient, similar to ReLU (Rectified Linear Unit), but with a smoother slope in the negative region. This can facilitate learning and gradient propagation during NN training. It allows for a greater flow of information through the network compared to other functions like ReLU, which zeros out negative information. This can lead to better problem representation and higher model accuracy. Swish has been shown to achieve better performance than other activation functions in various ML tasks, such as image classification, machine translation, and natural language processing [29]. To improve the accuracy of the output estimate at time t, past samples were incorporated as additional inputs to the model, as illustrated in Figure 4.11. This approach leverages historical data without requiring extra real-time measurements.

The Bayesian optimization method was employed to train the two NNs. The hyperparameter variation ranges provided in Table 4.21 were used to guide this training process.

Name	Range	Type	Value
Initial learn rate	$[10^{-2} \div 10^{-1}]$	Real	0.8983
Hidden units	$[100 \div 500]$	Integer	873
L2 regularisation coefficient	$[10^{-6} \div 10^{-2}]$	Real	$1.3 \cdot 10^{-6}$
Dropout probability 1st layer	$[0.1 \div 0.2]$	Real	0.1929
Dropout probability 2nd layer	$[0.1 \div 0.2]$	Real	0.1131
Mini batch size	$[128, 256, 512, 1024]$	Integer	512
Sample number	$[1 \div 20]$	Integer	20

Table 4.21 | Range of variation of hyper-parameters and identified values.

4.9 | Improving roll reduction in road vehicles on uneven surfaces through the fusion of proportional control and reinforcement learning

This research addresses the pivotal role of active anti-roll bars in mitigating vehicle body roll during cornering, thereby enhancing overall stability, maneuverability, and comfort. The proposed approach integrates two distinct control methodologies—a straightforward error proportional controller and a Reinforcement Learning (RL) based controller. Each front and rear active anti-roll bar applies a roll-reducing torque computed by the proportional controller during cornering. However, this torque alone proves insufficient in effectively damping roll oscillations induced by road irregularities. The RL-based controller leverages observations encompassing Inertial Measurement Unit data (roll rate, pitch rate, and vertical acceleration), and wheel vertical displacements and employs the roll as a reward signal. This controller calculates two additional corrective torques. These torques are seamlessly incorporated by both front and rear

anti-roll bars alongside the proportional controller, effectively minimizing cornering oscillations. The results demonstrate the efficacy of the solution in significantly reducing vehicle roll, even in challenging road conditions. This novel hybrid control strategy combines the simplicity of proportional feedback with the adaptability of RL, offering a robust anti-roll system that excels in both cornering dynamics and rough terrain scenarios.

4.9.1 | Controller design

The realized controller, schematized in Figure 4.52, consists of two components, one proportional to the error between the null reference and the roll angle and the other based on Reinforcement Learning.

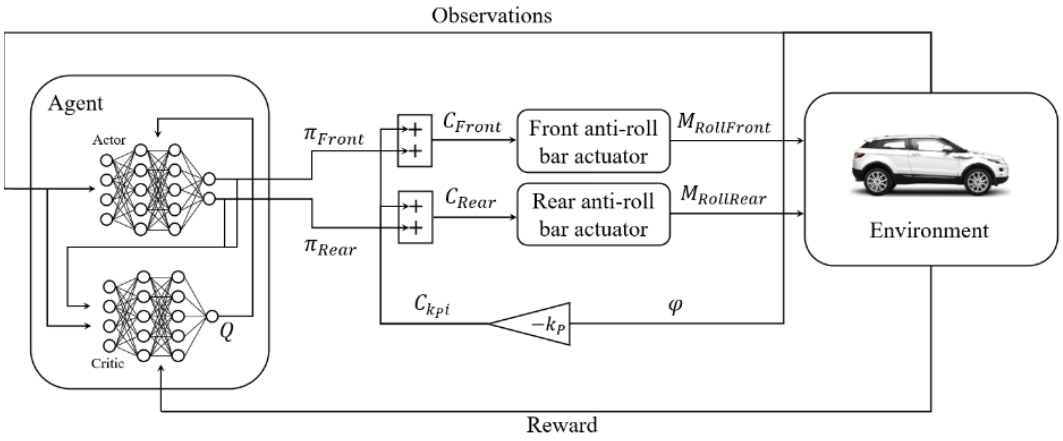


Figure 4.52 | Diagram of the overall control system with details of the signals and components involved.

The proportional controller receives the control error as input and returns a control signal, which is the same for both front and rear anti-roll bars:

$$C_{kpi} = k_P \cdot (0 - \varphi) = -k_P \cdot \varphi \tag{4.38}$$

Where $k_P = 2500 \frac{V}{rad}$ has been tuned to obtain a roll angle of less than $0.0035\, rad$ (0.2°) at a lateral acceleration of $3.5 \frac{m}{s^2}$ and in the absence of bumps in the road.

The control signal is a voltage signal that can vary between $-48V$ and $+48V$.

4.9.2 | Reinforcement learning enhancement

To achieve an efficient roll angle decrease even on bumpy roads, the proportional controller was combined with an RL-based controller acting in parallel.

The RL-controller is based on the interaction between an environment and an agent. The training of the RL-controller is divided into two phases, the first occurs offline where the environment is represented by a simplified physical model, the second occurs online where the environment is represented by the actual physical system. In the second phase, the controller reinforces the learning that took place initially offline. In the specific case of the controller realized in this study, the environment is represented by a 7-degree-of-freedom model, common with the model in Section 4.5, in which longitudinal and lateral accelerations and inputs from the road are signals acquired using the IPG CarMaker simulation software.

The agent belongs to the Deep Deterministic Policy Gradient (DDPG) category [105]. This agent can apply continuous actions over time.

In this study, the observations S are: roll rate $\dot{\varphi}$, pitch rate $\dot{\vartheta}$, vertical acceleration $\ddot{z}_O$, relative displacements of the wheels $\tilde{z}_{ij}$ concerning the chassis; the actions are the corrective signals π_{Front} and π_{Rear}; the reward is given by $-100 \cdot \varphi^2$, i.e. penalizes non-zero roll angle values.

The overall control signal C_i, the sum of the error proportional contribution C_{kpi} and the one based on Reinforcement Learning π_i, enters as an input to the actuator of the active anti-roll bar, which uses it to calculate the anti-roll torque. The relationship between the anti-roll bar

torque of the front and rear bars and the control signal is governed by a second-order transfer function:

$$\frac{M_{Roll_i}(s)}{C_i(s)} = \frac{G}{(\tau_1 s + 1) \cdot (\tau_2 s + 1)} \tag{4.39}$$

Where $G = 400;\ \tau_1 = 69ms;\ \tau_1 = 14ms.$

The Reinforcement Learning NNs were trained by having the vehicle perform a slalom maneuver in the IPG CarMaker software. Longitudinal and lateral accelerations were recorded and used as input to the 7-degree-of-freedom lumped-parameter model. From this model, which constitutes the Reinforcement Learning environment, observations are sampled, and rewards are generated.

Figure 4.53 shows the longitudinal and lateral acceleration of the vehicle realized during the training maneuver.

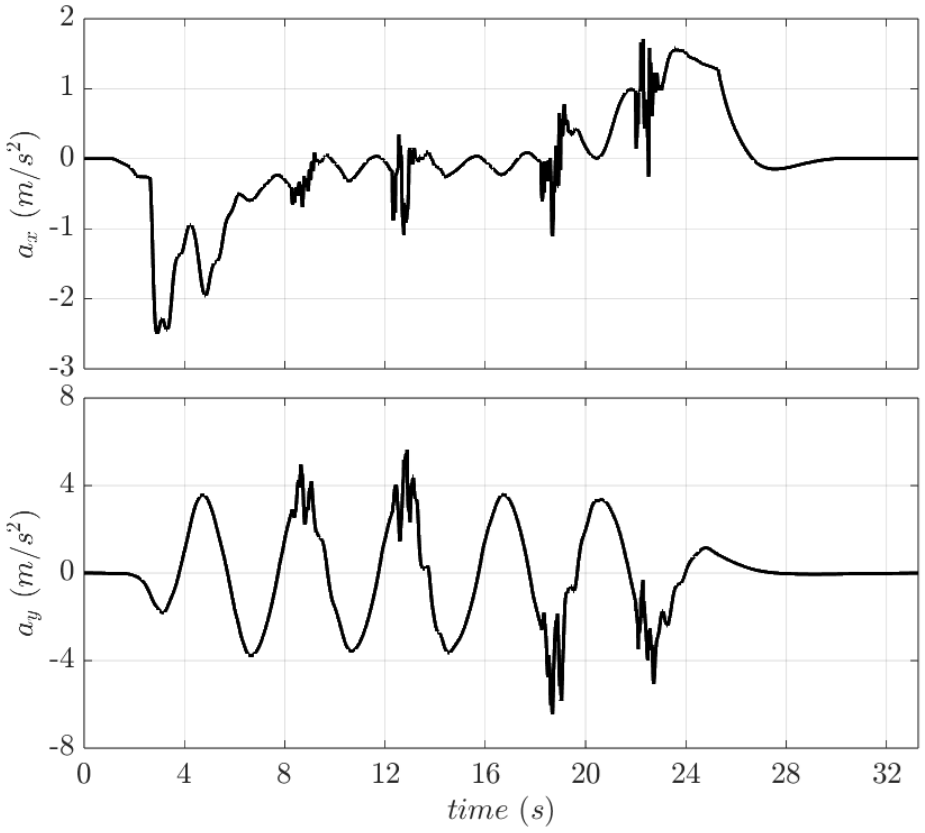


Figure 4.53 | Longitudinal and lateral accelerations of the vehicle of the performed slalom maneuver in the case of the training maneuver.

Figure 4.54 shows the vertical displacement of the road wheel contact points due to road bumps.

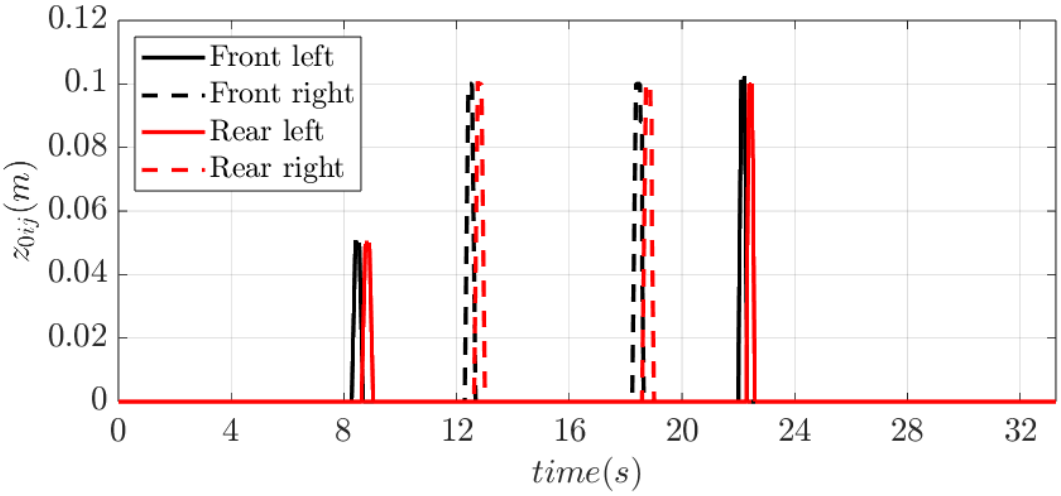


Figure 4.54 | Vertical displacement of the road wheel contact points due to road bumps in the case of the training maneuver.

As a result of proportional control only, the vehicle's roll decreases, however, road bumps cause oscillations that are not attenuated by proportional control solely. An RL agent was trained offline to decrease such oscillations. The training lasted 100 episodes. Each episode had a duration of 33.27 s with a sample time of 0.01 s. Consequently, the number of steps per episode was 3327 and the total number of steps was 3327000.

To ensure the generalization of the results, OU noise was added during training. Table 4.22 shows the values characterizing the OU noise for this study. Figure 4.55 shows the noise versus steps.

Symbol	Description	Value
μ	Noise mean value	0 (V)
$\hat{\mu}$	Constant specifying how quickly the noise model output is attracted to the mean	0.7
σ	Initial value of noise standard deviation	10 (V)
$d\sigma$	Decay rate of the standard deviation	10^{-5}

σ_{min}	Minimum value of σ	0 (V)
T_S	Agent sample time	0.01 (s)
$v(1)$	Initial noise	0 (V)

Table 4.22 | Values characterizing the OU noise.

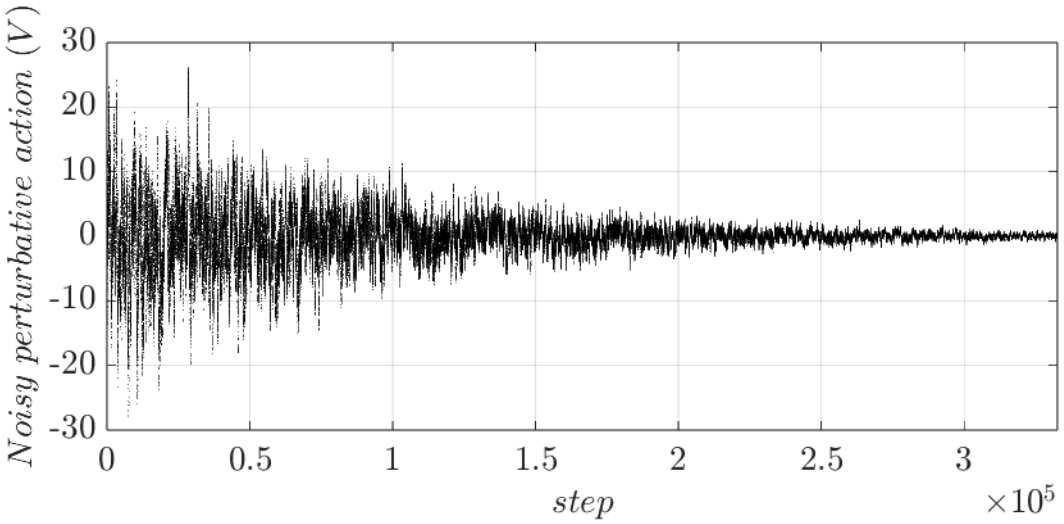


Figure 4.55 | OU noise trend by training steps.

The training of the NNs took place with a progressive increase in reward, as shown in Figure 4.56.

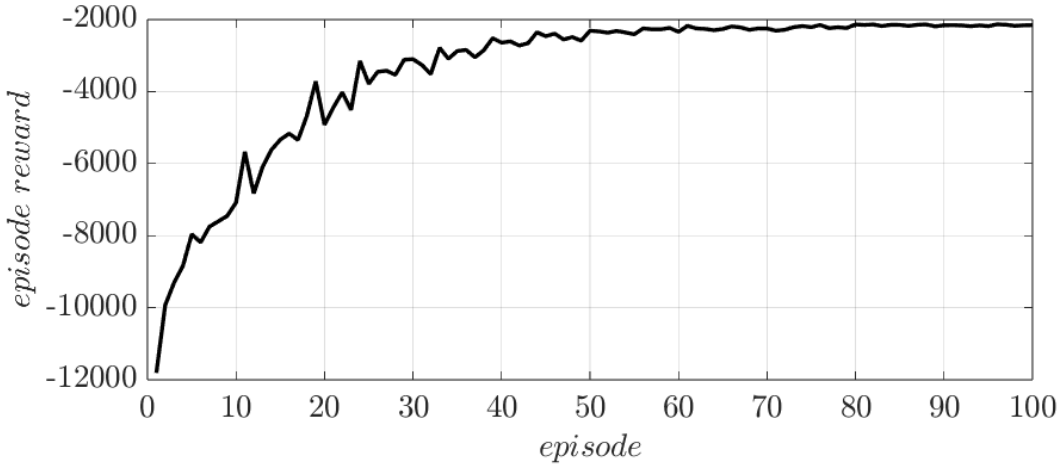


Figure 4.56 | Trend of total reward per episode as training progresses.

Figure 4.57 shows a comparison of roll angle in the absence of anti-roll torque, in the presence of proportional control, and in the presence of

proportional control with simultaneous correction by Reinforcement Learning in the case of the training maneuver.

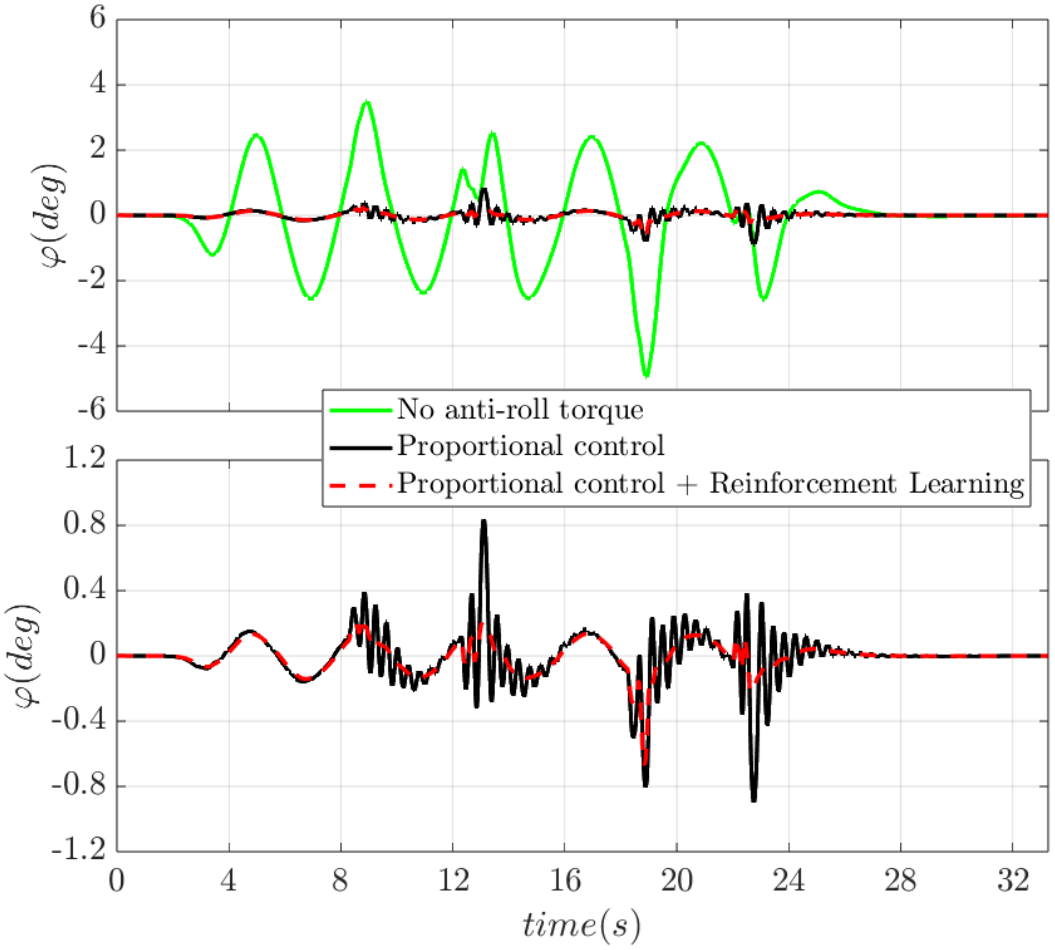


Figure 4.57 | Comparison of roll angle in the absence of anti-roll torque, in the presence of proportional control, and in the presence of proportional control and correction by Reinforcement Learning in the case of the training maneuver.

Table 4.23 shows RMSE in the presence of proportional control and in the presence of proportional control with simultaneous correction by Reinforcement Learning in the case of the training maneuver.

	Proportional control	Proportional control + RL
RMSE (deg)	0.1585	0.0930

Table 4.23 | RMSE of the roll with respect to the null reference in the absence of control, in the presence of proportional control, and in the presence of proportional control with simultaneous correction by Reinforcement Learning in the case of the training maneuver.

5 | RESULTS OF PROPOSED APPLICATIONS

This section provides the results of the studies presented in Section 4.

Root Mean Square Error (RMSE) was used to evaluate the performance of the proposed estimation and control solutions.

The RMSE is defined as follows:

$$RMSE = \sqrt{\sum_{j=1}^{n} \frac{(\hat{x}_j - x_j)^2}{n}} \tag{5.1}$$

Where:

- $\hat{x}_j$ is the j-th estimated sample of the test maneuver.
- x_j is the j-th reference sample of the test maneuver.
- n is the number of samples of the test maneuver.

5.1 | Multi-output physically analyzed neural network for the prediction of tire–road interaction forces

5.1.1 | Test maneuvers

The NN test was carried out using maneuvers to highlight the contribution of each of the two inputs in the prediction of the road-tire interaction forces in the three spatial directions of the wheel reference system SyW:

- *Constant Radius Cornering*: in this maneuver, the vehicle travels a curve with a constant radius of 30 m in a clockwise direction, first accelerating and then decelerating. Figure 5.1 shows the non-

normalized accelerations along the three directions of *SyV* for this maneuver.

- *Slalom*: in this maneuver, the vehicle advances through seven obstacles positioned 17 m from each other and at a speed of approximately 60 $\frac{km}{h}$, making rapid changes in direction. Figure 5.2 shows the non-normalized accelerations along the three directions of *SyV* for this maneuver.

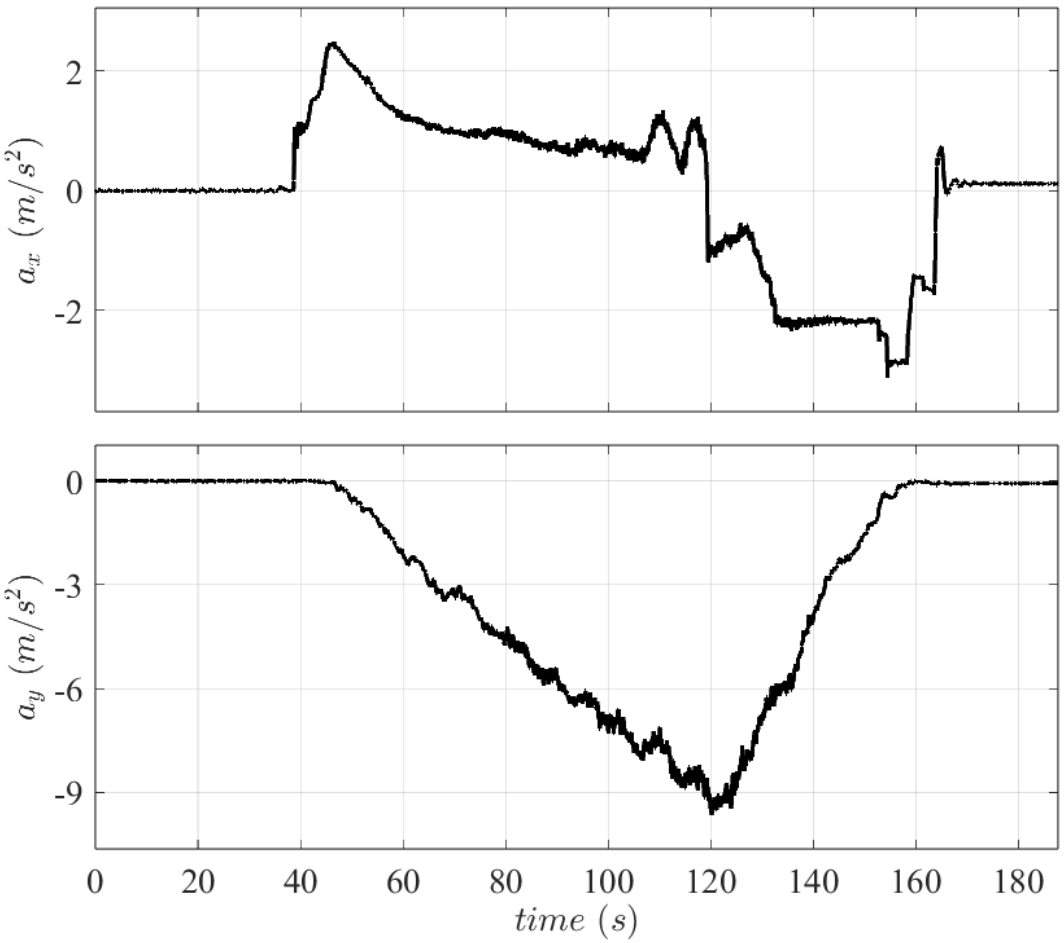


Figure 5.1 | Non-normalized accelerations in ***SyV*** | Constant Radius Cornering.

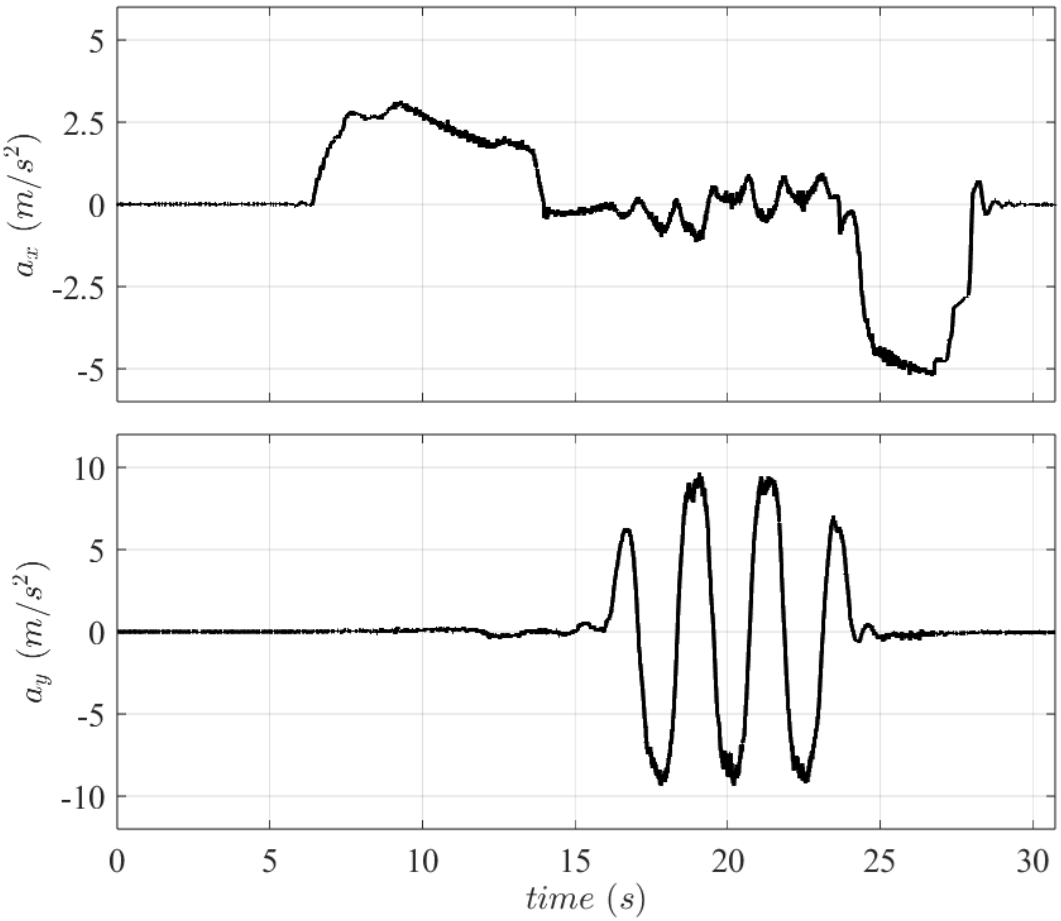


Figure 5.2 | Non-normalized accelerations in ***SyV*** | Slalom.

5.1.2 | Input and output denormalization

The trained NN requires normalized inputs and returns normalized outputs. The inputs must be normalized using the mean and standard deviation used to normalize the inputs of the training and validation datasets, as shown in Equation (5.2). The outputs must be denormalized using the mean and standard deviation used to normalize the outputs of the training and validation datasets, as shown in Equation (5.3).

$$\tilde{a}_{ij} = \frac{a_{ij} - \mu_{a_i}}{\sigma_{a_i}} \tag{5.2}$$

Where:

- $\tilde{a}_{ij}$ is the j-th normalized sample of the test maneuver relative to the acceleration measured in the i-th direction of SyV.

- a_{ij} is the j-th non-normalized sample of the test maneuver relative to the acceleration measured in the i-th direction of SyV.
- μ_{a_i} is the mean of all accelerations of the training and validation datasets in the i-th direction of SyV.
- σ_{a_i} is the standard deviation of all accelerations of the training and validation datasets in the i-th direction of SyV.
- $i = x, y$.

$$\hat{F}_{i_{Rk_j}} = \hat{\hat{F}}_{i_{Rk_j}} \cdot \sigma_{F_{i_{Rk}}} + \mu_{F_{i_{Rk}}} \tag{5.3}$$

Where:

- $\hat{\hat{F}}_{i_{Rk_j}}$ is the j-th normalized sample of the test maneuver relative to the predicted force of the k-th tire in the i-th direction of SyW.
- $\hat{F}_{i_{Rk_j}}$ is the j-th non-normalized sample of the test maneuver relative to the predicted force of the k-th tire in the i-th direction of SyW.
- $\mu_{F_{i_{Rk}}}$ is the mean of all forces of the training and validation datasets of the k-th tire in the i-th direction of SyW.
- $\sigma_{F_{i_{Rk}}}$ is the standard deviation of all forces of the training and validation datasets of the k-th tire in the i-th direction of SyW.
- $i = x, y, z$.
- $j = R, L$.

5.1.3 | Predicted forces

The predicted forces relative to the reference are shown in Figures 5.3 and 5.4.

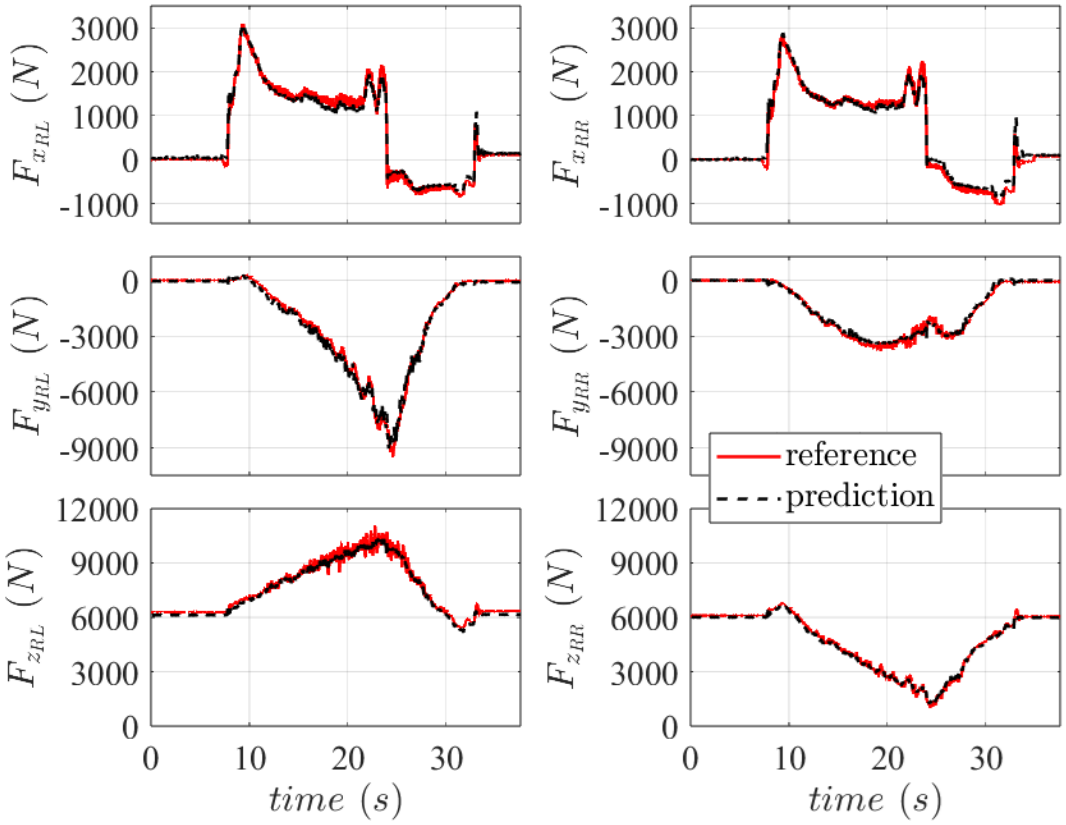


Figure 5.3 | Reference vs predicted forces | Constant Radius Cornering.

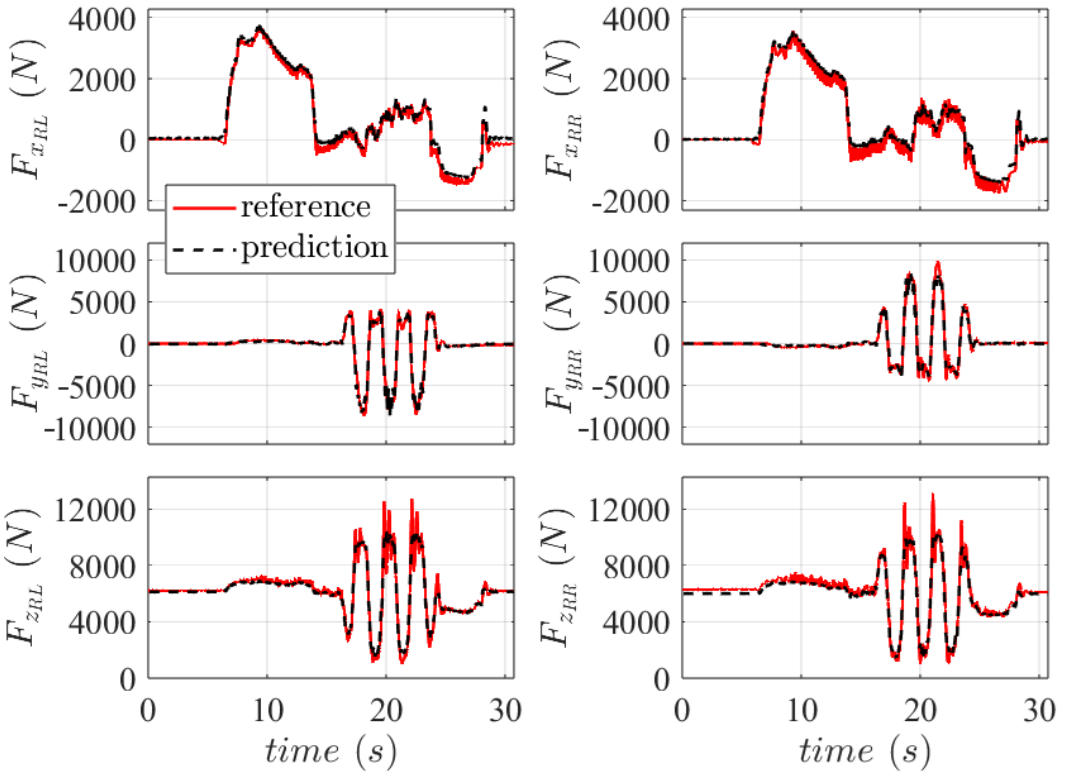


Figure 5.4 | Reference vs predicted forces | Slalom.

Table 5.1 shows RMSE between non-normalized predictions and non-normalized references obtained for each force and maneuver [75].

Test	$F_{x_{RL}}\ (N)$	$F_{y_{RL}}\ (N)$	$F_{z_{RL}}\ (N)$	$F_{x_{RR}}\ (N)$	$F_{y_{RR}}\ (N)$	$F_{z_{RR}}\ (N)$
Constant Radius	104.79	217.06	222.65	107.81	143.66	134.23
Slalom	168.89	365.34	453.22	206.79	421.66	449.36

Table 5.1 | RMSE between predicted and reference forces.

Figures 5.3 and 5.4, and Table 5.1 show that the NN is able to accurately predict the six output forces using only longitudinal and lateral acceleration measurements.

5.1.4 | Influence of lateral load transfer on predicted lateral forces

Remarks can be made on the prediction of lateral forces:

- Looking at the second row of Figure 5.3, we see that the lateral forces, although both negative, take different values. In particular, it is greater in absolute value for the left tire. This is due to the lateral load transfer that, by increasing the vertical force on the left tire and decreasing it on the right tire, amplifies the lateral force on the left tire compared to the right tire. The NN is able to predict such behavior while using a single lateral acceleration.

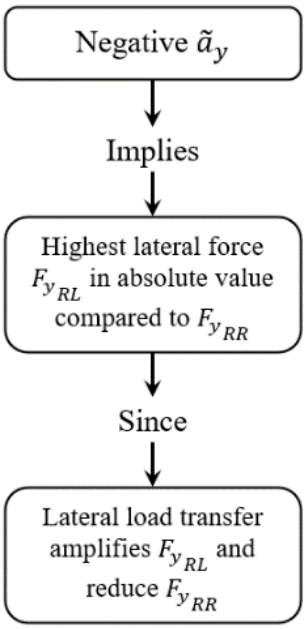


Figure 5.5 | Asymmetry of lateral forces due to lateral load transfer.

- The second row of Figure 5.4 shows between seconds 16 and 25 an asymmetry of the lateral force from the zero value. This is because the lateral load transfer, by alternately increasing and decreasing the vertical forces on the left and right tires, results in an alternative amplification of the lateral force on one tire relative to the other. The NN is able to predict such behavior despite the lateral acceleration oscillating symmetrically around zero, as shown in Figure 5.2.

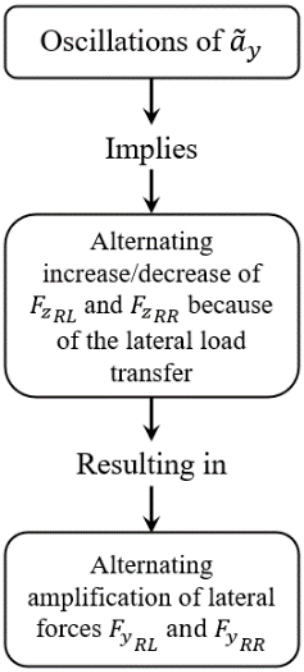


Figure 5.6 | Alternating amplification of lateral forces due to alternating lateral load transfer.

5.1.5 | Input sensitivity analysis

To analyze the influence of each input on the outputs of the NN, each normalized input was set equal to the mean value of the training and validation datasets, i.e., zero. In this way, all the variability of that input quantity was canceled out and consequently also its influence on the predicted outputs.

Tables 5.2 and 5.3 show RMSE and percent variation obtained for each force and each maneuver following the zeroing of each normalized input. Figures 5.7 and 5.8 show the variation in RMSE for each maneuver with and without zeroing the normalized inputs. The increase in RMSE is a consequence of the impossibility of the NN to find the interconnections between the inputs necessary to predict the outputs.

Test		$F_{x_{RL}}$ (N)	$F_{y_{RL}}$ (N)	$F_{z_{RL}}$ (N)	$F_{x_{RR}}$ (N)	$F_{y_{RR}}$ (N)	$F_{z_{RR}}$ (N)
Constant radius	$\tilde{a}_{xj} = 0$	1073.83	274.30	484.29	996.97	234.61	350.91
Slalom		1421.79	404.11	733.91	1361.89	461.61	725.61
Constant radius	$\tilde{a}_{yj} = 0$	202.95	3212.93	2040.00	254.84	1978.47	2091.31
Slalom		255.12	2370.79	1733.73	250.09	2271.87	1759.72

Table 5.2 | RMSE after zeroing of normalized inputs.

Test		$F_{x_{RL}}$ (%)	$F_{y_{RL}}$ (%)	$F_{z_{RL}}$ (%)	$F_{x_{RR}}$ (%)	$F_{y_{RR}}$ (%)	$F_{z_{RR}}$ (%)
Constant radius	$\tilde{a}_{xj} = 0$	924.74	26.37	117.51	824.75	63.31	161.42
Slalom		741.84	10.61	61.93	558.59	9.47	61.48
Constant radius	$\tilde{a}_{yj} = 0$	93.67	1380.20	816.24	136.38	1277.19	1458.00
Slalom		51.06	548.93	282.54	20.94	438.79	291.61

Table 5.3 | Percent variation of RMSE after zeroing of normalized inputs.

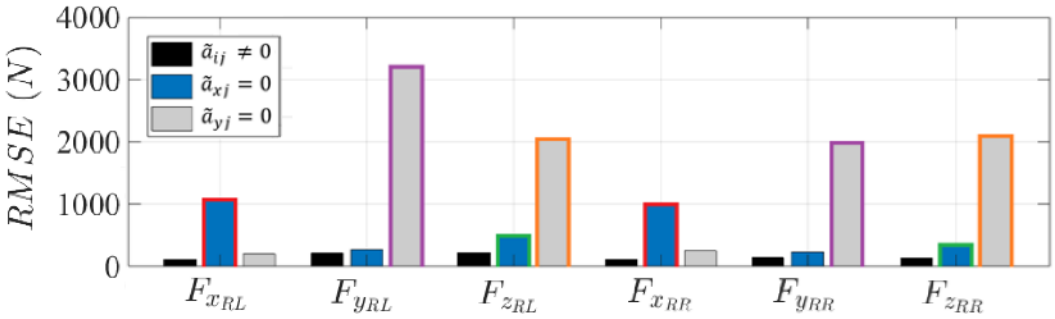


Figure 5.7 | RMSE with and without zeroing of normalized inputs | Constant Radius Cornering.

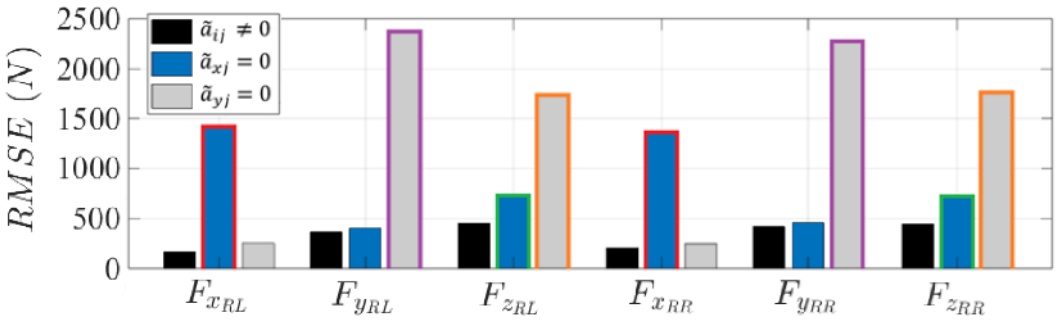


Figure 5.8 | RMSE with and without zeroing of normalized inputs | Slalom.

Figures 5.9-5.12 show the predicted forces relative to the reference following the zeroing of each input.

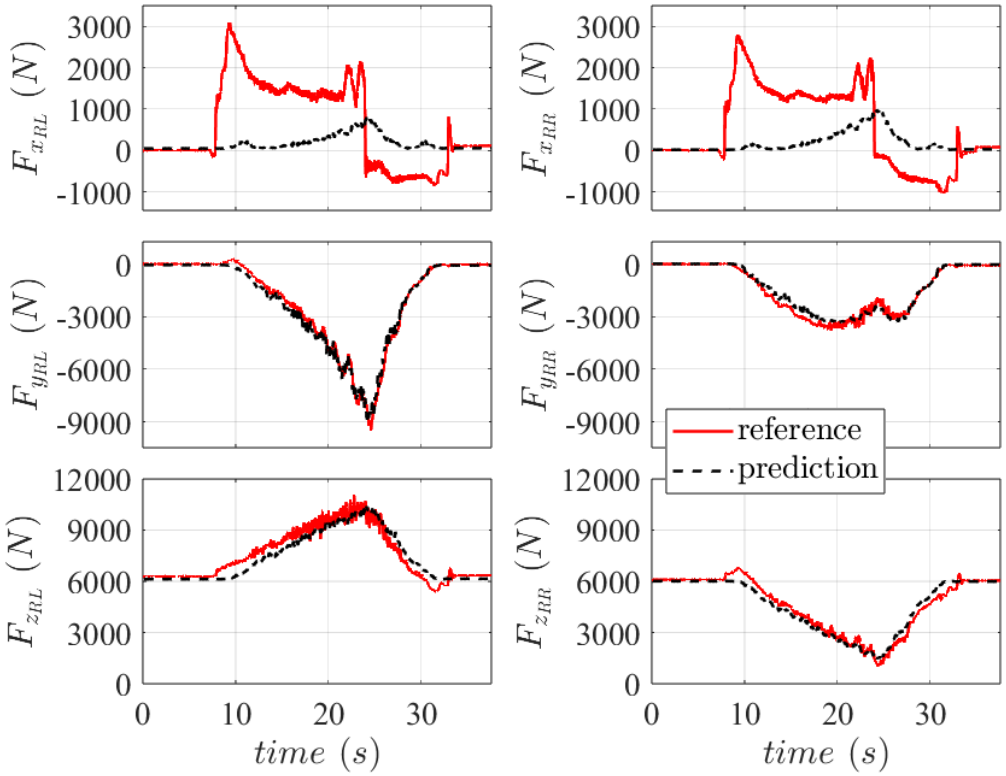


Figure 5.9 | Reference vs predicted forces ($\tilde{a}_{xj} = 0$) | Constant Radius Cornering.

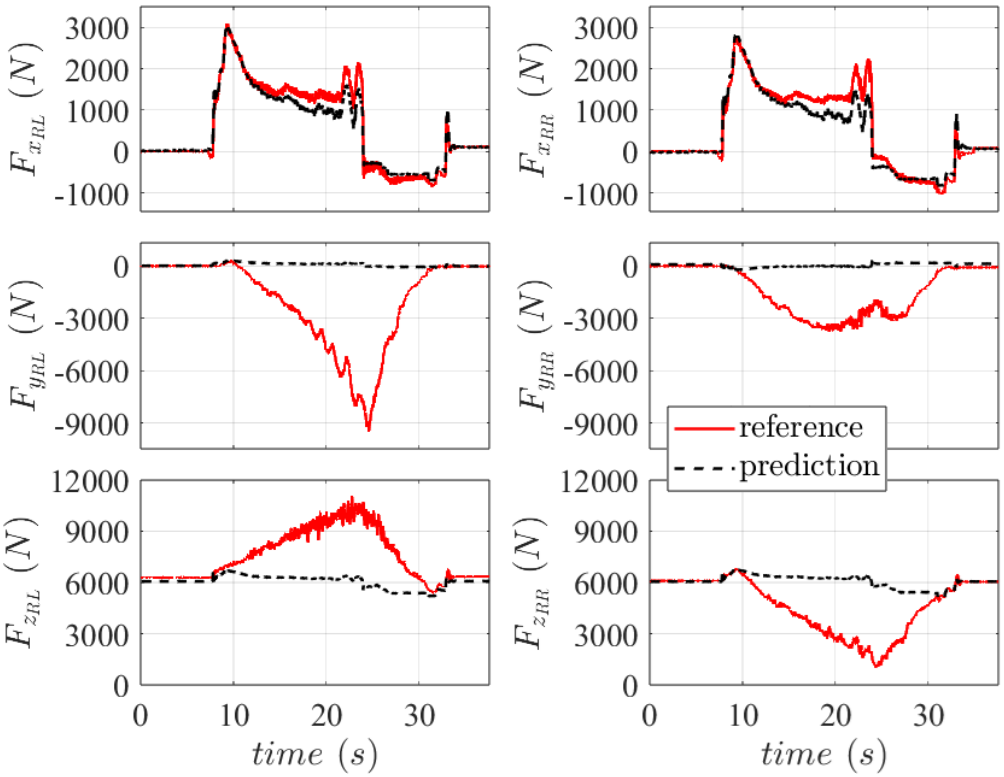


Figure 5.10 | Reference vs predicted forces ($\tilde{a}_{yj} = 0$) | Constant Radius Cornering.

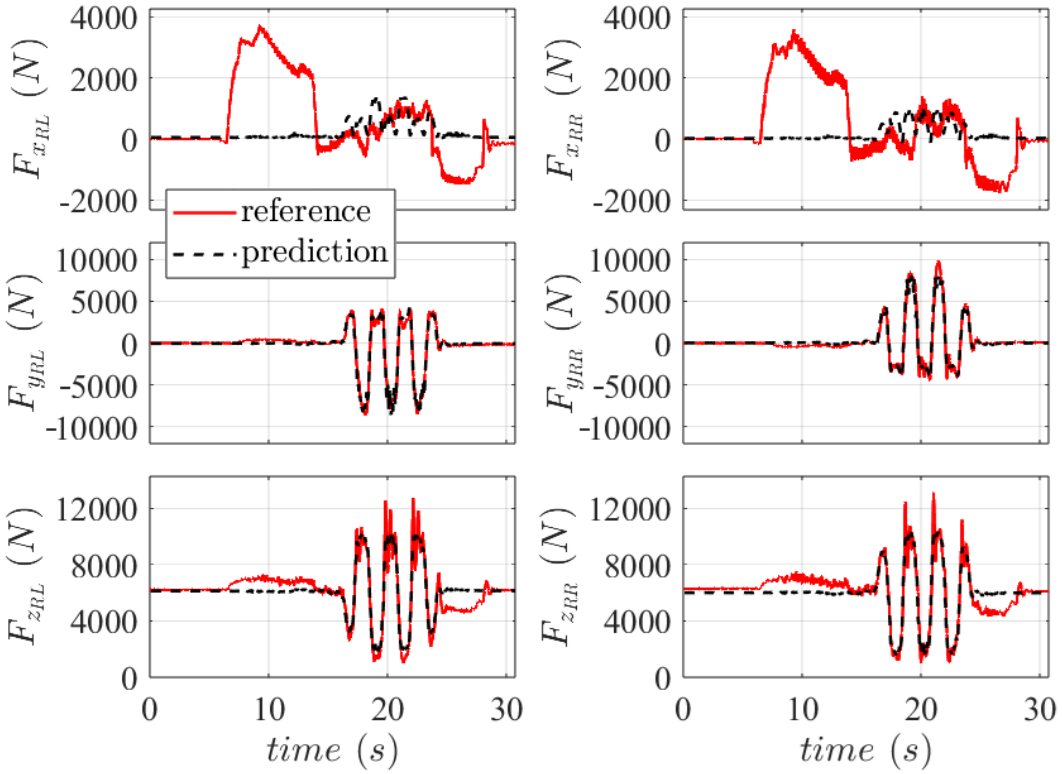


Figure 5.11 | Reference vs predicted forces ($\tilde{a}_{xj} = 0$) | Slalom.

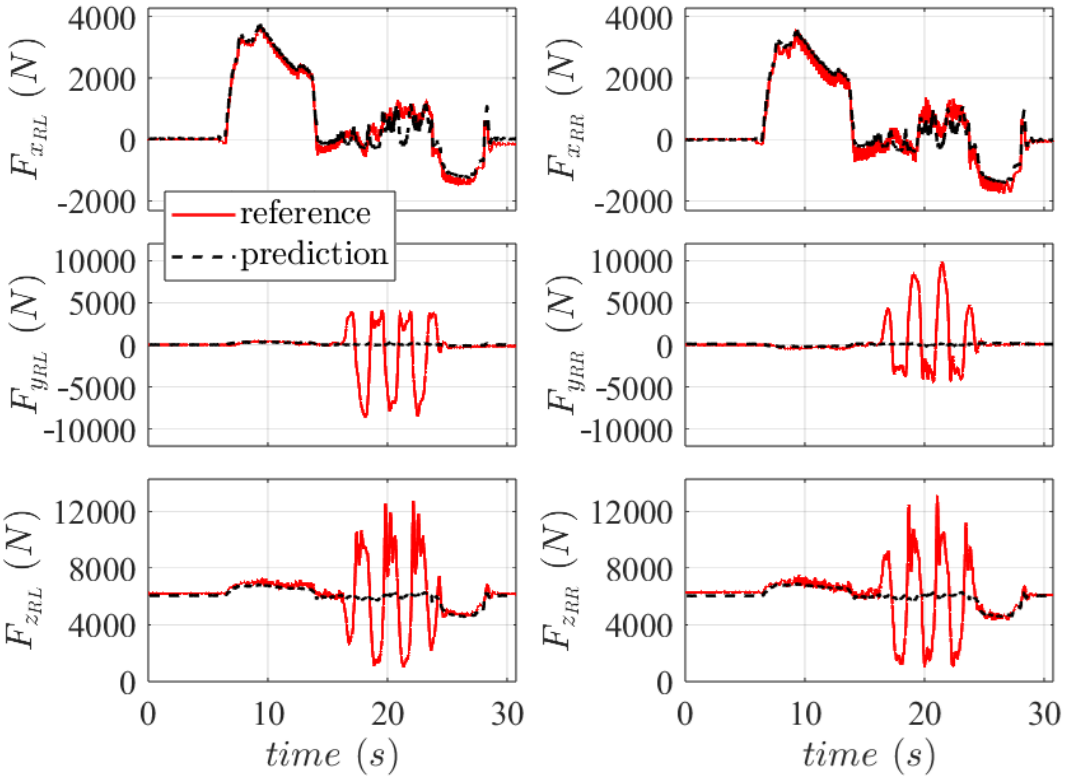


Figure 5.12 | Reference vs predicted forces ($\tilde{a}_{yj} = 0$) | Slalom.

Tables 5.2 and 5.3 and Figures 5.7 and 5.8 highlight specifically by which input the prediction of each force is most affected:

- Zeroing longitudinal acceleration $\tilde{a}_x$ resulted in a large increase in RMSE for longitudinal forces $F_{x_{Rk}}$, highlighted in red. When the driver accelerates or brakes, the road exerts positive or negative longitudinal forces on the tires in the forward direction that allow the vehicle to increase speed, positive acceleration, or decrease speed, negative acceleration. Thus, the physical dependence between these quantities is evident, allowing longitudinal acceleration measurements to be employed to predict the longitudinal forces that the road exerts on the tires. The first rows of Figures 5.9 and 5.11 clearly show that the NN can no longer accurately predict longitudinal forces in the absence of longitudinal acceleration information.
- Zeroing longitudinal acceleration $\tilde{a}_x$ resulted in increased RMSE for vertical forces $F_{z_{Rk}}$, highlighted in green. When the driver accelerates, the vehicle's center of gravity shifts backward due to vehicle inertia, causing an increase in load on the rear tires. Similarly, when the driver brakes, the vehicle's center of gravity shifts forward, causing a decrease in the vertical load on the rear tires. Longitudinal load transfer is the term for this phenomenon [68]. This means that when the longitudinal acceleration is positive, the vertical forces on the rear tires increase; conversely, when the longitudinal acceleration is negative, the vertical forces on the rear tires decrease. Looking at the third row in Figure 5.9 between seconds 8 and 23, i.e., when the longitudinal acceleration is positive, the vertical forces are underestimated. This is due to the inability of the NN to predict the

load increase on the rear tires due to vehicle inertia in the absence of longitudinal acceleration information. A similar observation can be made by looking at the third row of Figure 5.11 between seconds 6 and 14. Looking at the third row of Figure 5.9 between seconds 23 and 33, i.e., when the longitudinal acceleration is negative, the vertical forces are overestimated. This is due to the inability of the NN to predict the decrease in vertical load on the rear tires due to vehicle inertia in the absence of longitudinal acceleration information. A similar observation can be made by looking at the third row in Figure 5.11 between seconds 24 and 28.

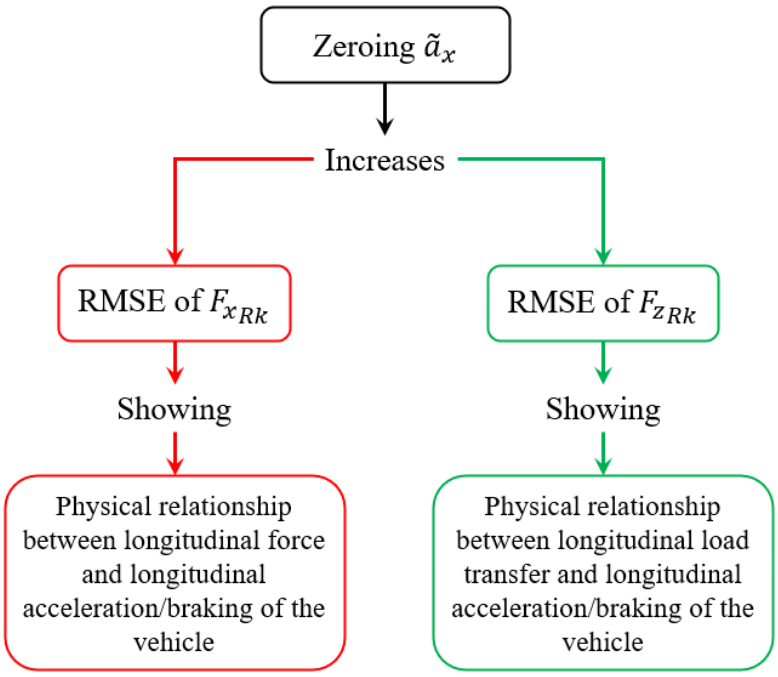


Figure 5.13 | Influence of $\tilde{a}_x$ zeroing on RMSE of longitudinal forces $F_{x_{Rk}}$ and vertical forces $F_{z_{Rk}}$.

- Zeroing lateral acceleration $\tilde{a}_y$ resulted in a large increase in RMSE for lateral forces $F_{y_{Rk}}$, highlighted in purple. When the driver drives through a curve, the road exerts lateral adhesion forces on the tires that are directed toward the center of the curve. These lateral

adhesion forces oppose the centrifugal force that pushes the vehicle toward the outside of the curve. These forces result in lateral acceleration of the vehicle toward the center of the curve. Thus, the physical dependence between these quantities is clear, allowing lateral acceleration measures to be employed to predict the lateral forces exerted on the tires by the road. The second rows of Figures 5.10 and 5.12 clearly show that the NN can no longer accurately predict lateral forces in the absence of lateral acceleration information.

- Zeroing lateral acceleration $\tilde{a}_y$ resulted in an increase in RMSE for vertical forces $F_{z_{Rk}}$, highlighted in orange. When the driver travels through a curve, the centrifugal acceleration, equal in modulus and direction and opposite in direction to the acceleration $\tilde{a}_y$, results in a shift of the vehicle's center of gravity outward with a consequent increase in vertical load on the outer tires and a decrease in vertical load on the inner tires. Lateral load transfer is the term for this phenomenon [68]. This means that when the lateral acceleration is positive, the vertical forces on the right tires increase and those on the left tires decrease; conversely, when the lateral acceleration is negative, the vertical forces on the right tires decrease and those on the left tires increase. Looking at the third row of Figure 5.10 between seconds 8 and 33, we see that the vertical forces are underestimated for the left tire and overestimated for the right tire. This is due to the inability of the NN to predict the increase in vertical load on the left tire and the decrease in vertical load on the right tire due to vehicle inertia in the absence of lateral acceleration information. A similar

observation can be made by looking at Figure 5.12 between seconds 16 and 25.

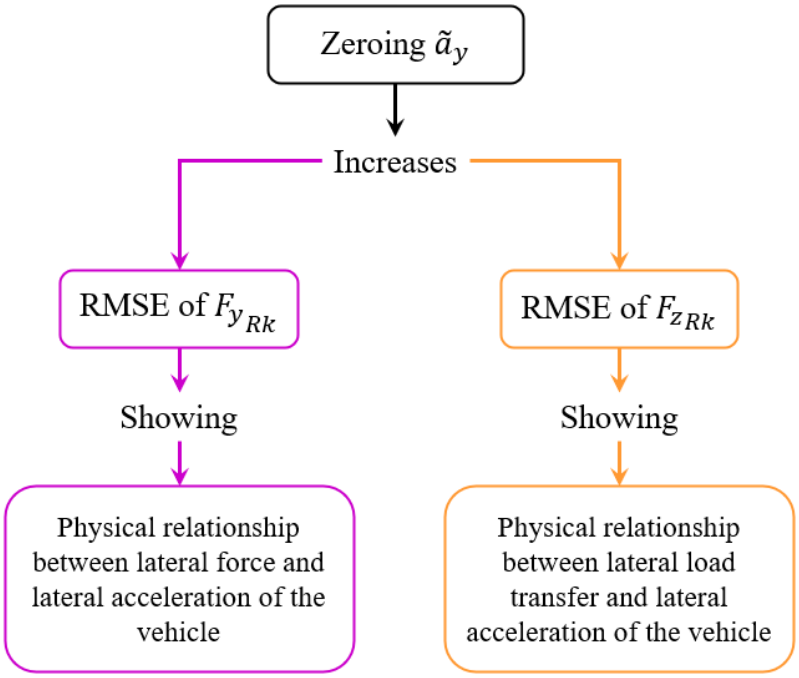


Figure 5.14 | Influence of $\tilde{a}_y$ zeroing on RMSE of lateral forces $F_{y_{Rk}}$ and vertical forces $F_{z_{Rk}}$.

- Looking at the second rows of Figures 5.9 and 5.10 between seconds 8 and 33, it can be seen that in both cases, the NN succeeds in predicting a variation from the static vertical force. In the first case, due to lateral load transfer only, in the second case, due to longitudinal load transfer only. In Figure 5.3, the resulting forces are accurately predicted because the NN managed to combine the two accelerations, longitudinal and lateral, to simultaneously predict both load transfers.

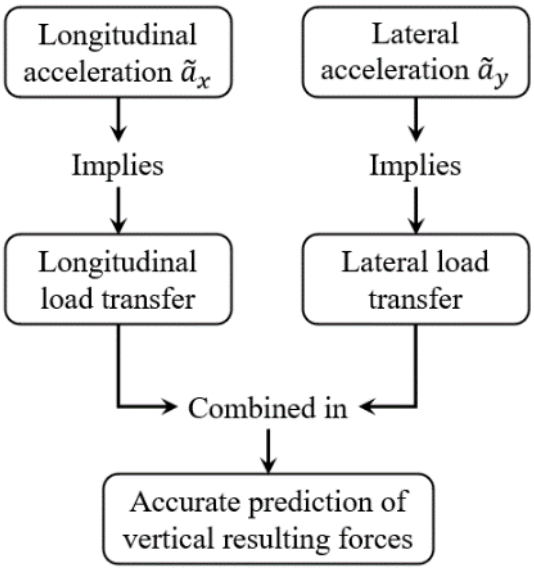


Figure 5.15 | Combination of longitudinal and lateral load transfers for estimating vertical forces.

5.2 | Estimation of the tire-road interaction forces by using Pacejka's formulas

The suitability of the proposed technique to estimate tire-road interaction forces has been assessed by comparing estimation results with ones obtained through simulations made in the CarMaker environment employing an experimentally validated vehicle on realistic circuits, Nürburgring and Hockenheim. Furthermore, a Gaussian white noise has been added to the measurements to make them more realistic. The graphs and tables show the results obtained. The graphs, shown in Figures 5.16-5.21, overlap references and estimates. Tables 5.4-5.9 show the Root Mean Square Error (RMSE) as a performance indicator of the proposed estimator.

5.2.1 | Lap on the Nürburgring circuit

The comparisons between estimated forces and simulated ones related to the Nürburgring circuit are shown in Figures 5.16-5.18. The obtained results demonstrate the capability of the proposed estimator to capture the behavior of tire-road interaction forces with a good estimation quality.

The values of RMSE confirm the quality of the estimation obtained through the model-based estimator designed around the Central Difference Kalman filter.

	$F_{x11}\ (N)$	$F_{x12}\ (N)$	$F_{x21}\ (N)$	$F_{x22}\ (N)$
RMSE	24	41	42	51

Table 5.4 | RMSE between predicted and reference longitudinal forces.

	$F_{y11}\ (N)$	$F_{y12}\ (N)$	$F_{y21}\ (N)$	$F_{y22}\ (N)$
RMSE	80	61	55	70

Table 5.5 | RMSE between predicted and reference lateral forces.

	$F_{z11}\ (N)$	$F_{z12}\ (N)$	$F_{z21}\ (N)$	$F_{z22}\ (N)$
RMSE	81	86	93	89

Table 5.6 | RMSE between predicted and reference vertical forces.

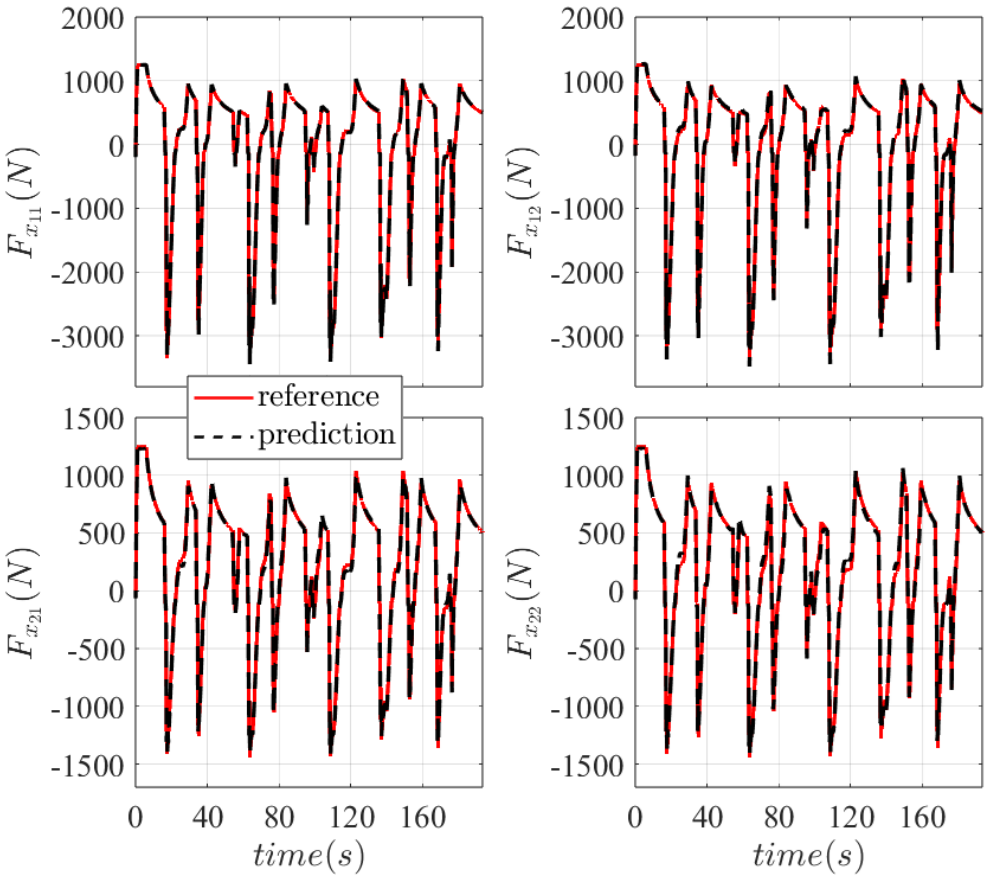


Figure 5.16 | Reference and estimated longitudinal forces (Nürburgring).

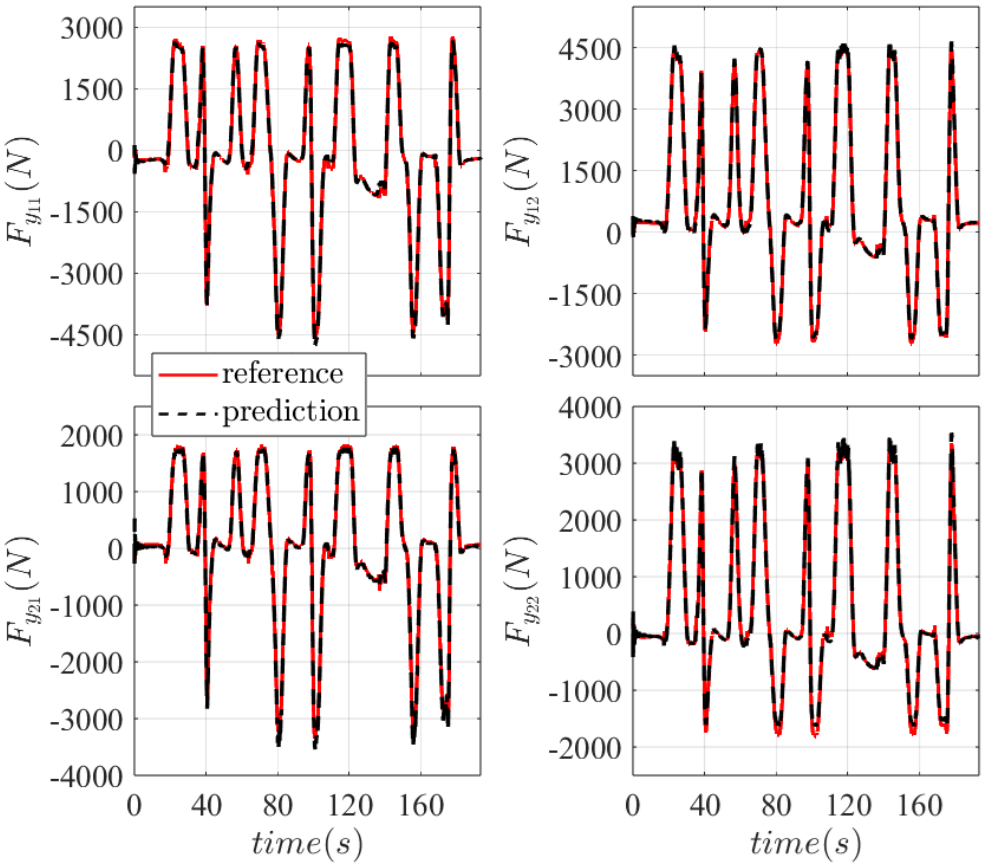


Figure 5.17 | Reference and estimated lateral forces (Nürburgring).

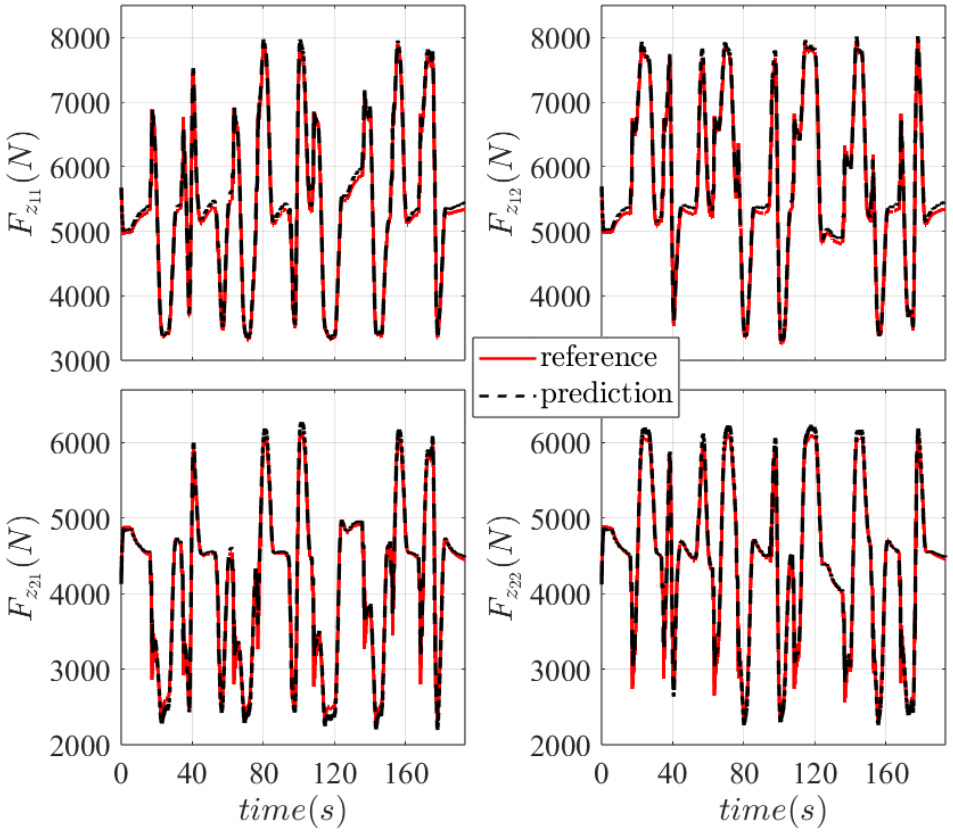


Figure 5.18 | Reference and estimated vertical forces (Nürburgring).

5.2.2 | Lap on the Hockenheim circuit

The comparisons between estimated forces and simulated ones related to the Hockenheim circuit are shown in Figures 5.19-5.21. The obtained results demonstrate the capability of the proposed estimator to capture the behavior of tire-road interaction forces with a good estimation quality.

The values of RMSE confirm the quality of the estimation obtained through the model-based estimator designed around the Central Difference Kalman filter.

	$F_{x11}\ (N)$	$F_{x12}\ (N)$	$F_{x21}\ (N)$	$F_{x22}\ (N)$
RMSE	53	67	67	70

Table 5.7 | RMSE between predicted and reference longitudinal forces.

	$F_{y11}\ (N)$	$F_{y12}\ (N)$	$F_{y21}\ (N)$	$F_{y22}\ (N)$
RMSE	87	80	92	86

Table 5.8 | RMSE between predicted and reference lateral forces.

	$F_{z11}\ (N)$	$F_{z12}\ (N)$	$F_{z21}\ (N)$	$F_{z22}\ (N)$
RMSE	122	94	109	119

Table 5.9 | RMSE between predicted and reference vertical forces.

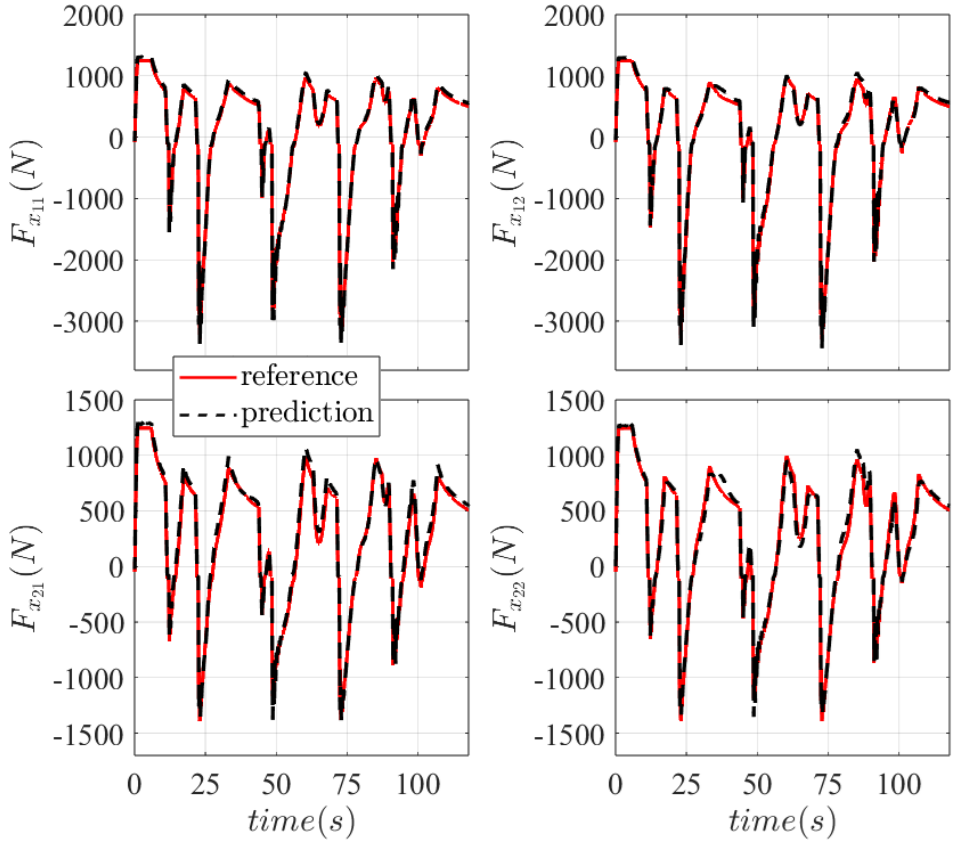


Figure 5.19 | Reference and estimated longitudinal forces (Hockenheim).

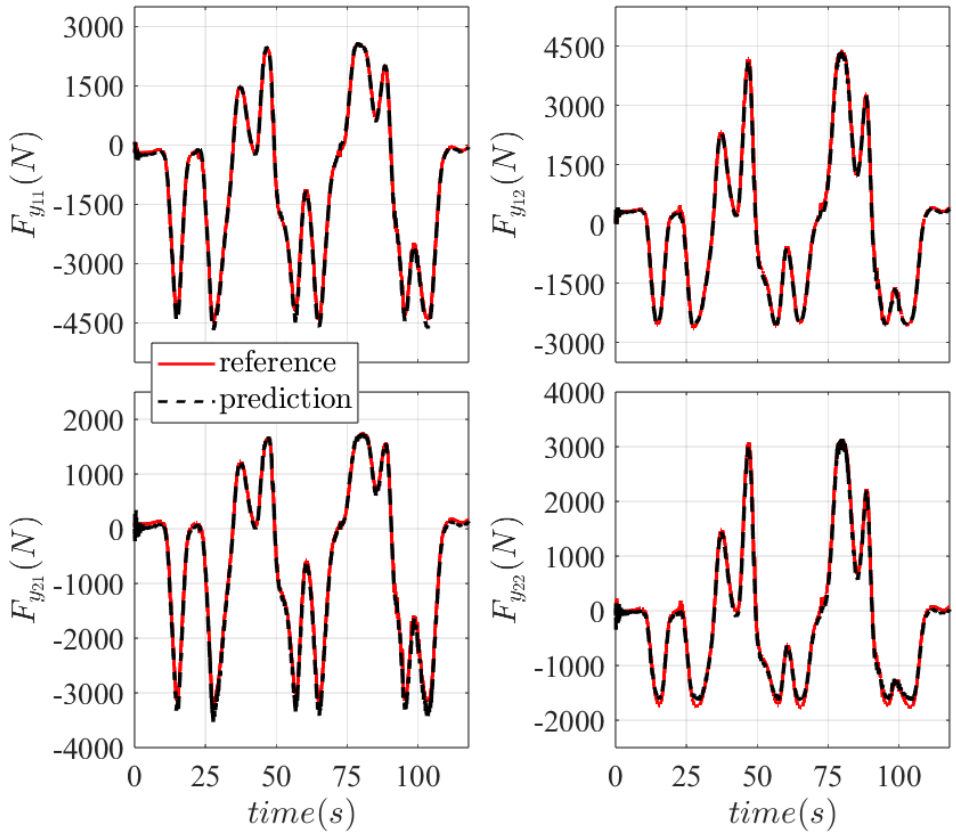


Figure 5.20 | Reference and estimated lateral forces (Hockenheim).

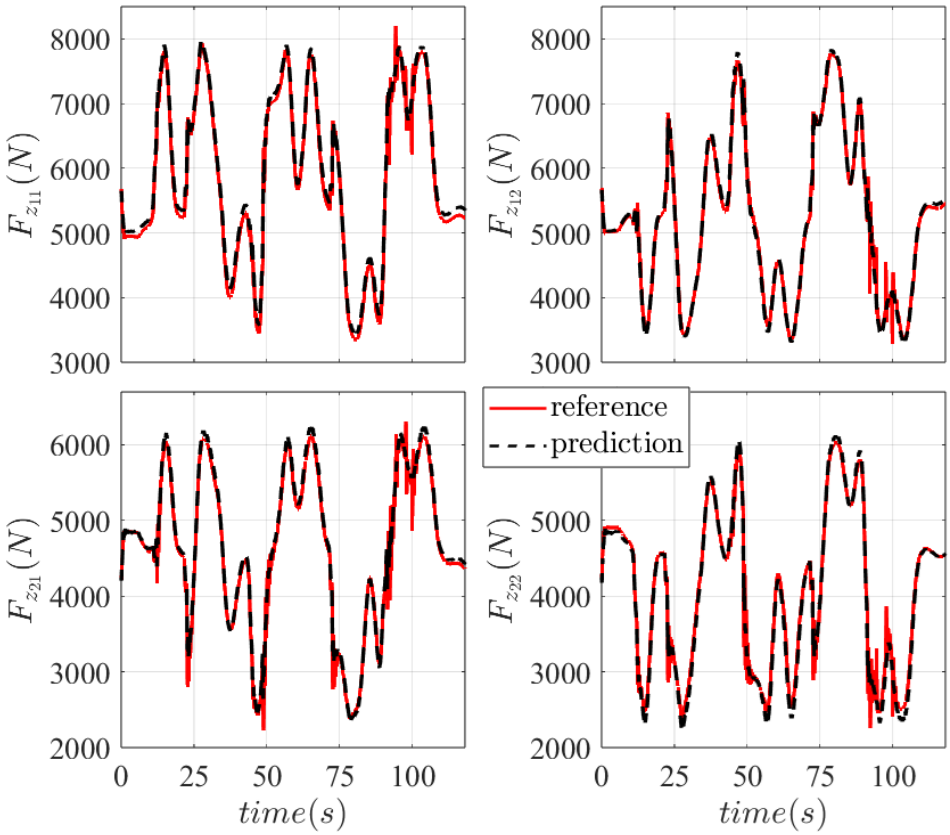


Figure 5.21 | Reference and estimated vertical forces (Hockenheim).

5.3 | Improvement of traction force estimation in cornering through neural network

The two NNs were tested by making two laps in the opposite direction of the Hockenheim track, shown in Figure 5.22.

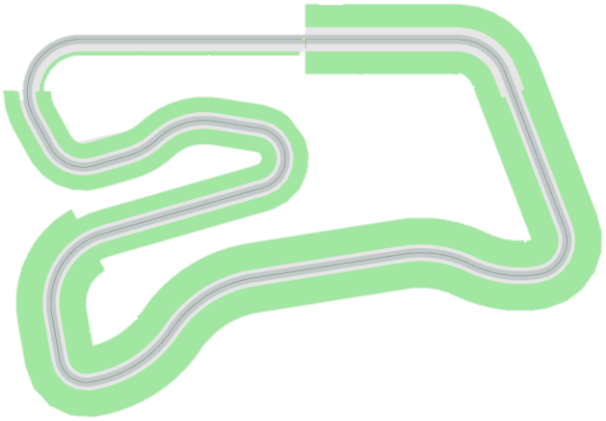


Figure 5.22 | Top view of the Hockenheim track used for training NNs.

5.3.1 | Traction force correction $\Delta\widehat{F}_{rj}$

Figures 5.23-5.26 show the predicted difference $\Delta\hat{F}_{rj}$ compared to the reference ΔF_{rj}. Along with the prediction, the inputs used by the two NNs are also shown.

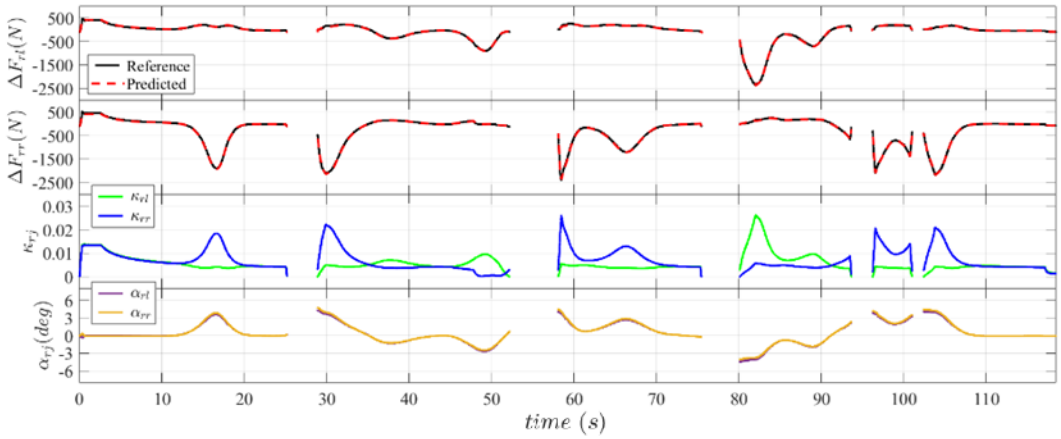


Figure 5.23 | Lap 1 | NN1: predicted $\Delta\hat{F}_{rj}$ vs reference ΔF_{rj} traction force correction.

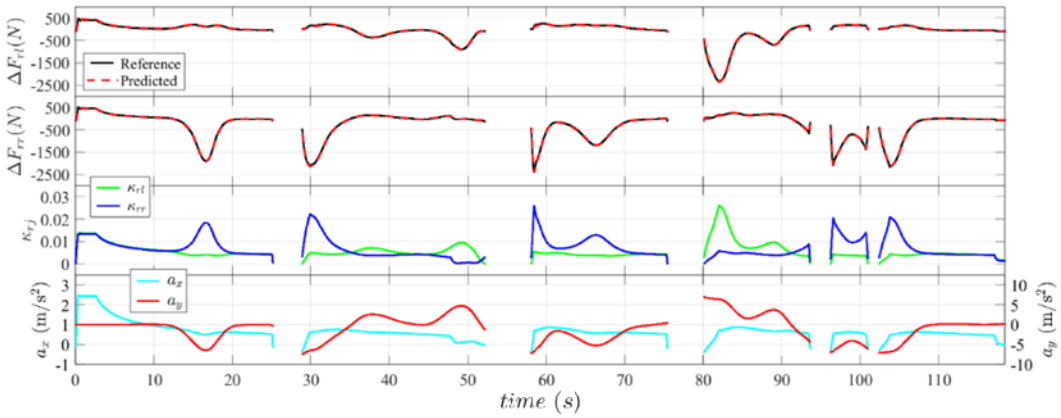


Figure 5.24 | Lap 1 | NN2: predicted $\Delta\hat{F}_{rj}$ vs reference ΔF_{rj} traction force correction.

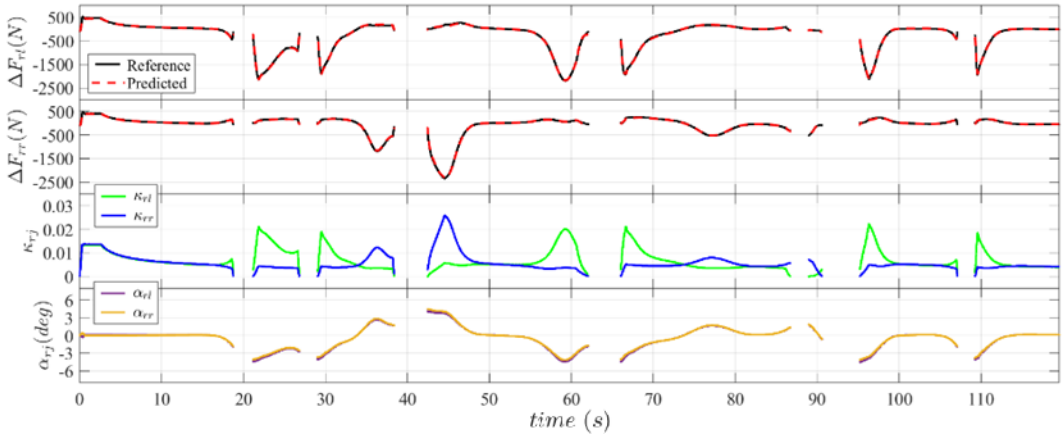


Figure 5.25 | Lap 2 | NN1: predicted $\Delta\hat{F}_{rj}$ vs reference ΔF_{rj} traction force correction.

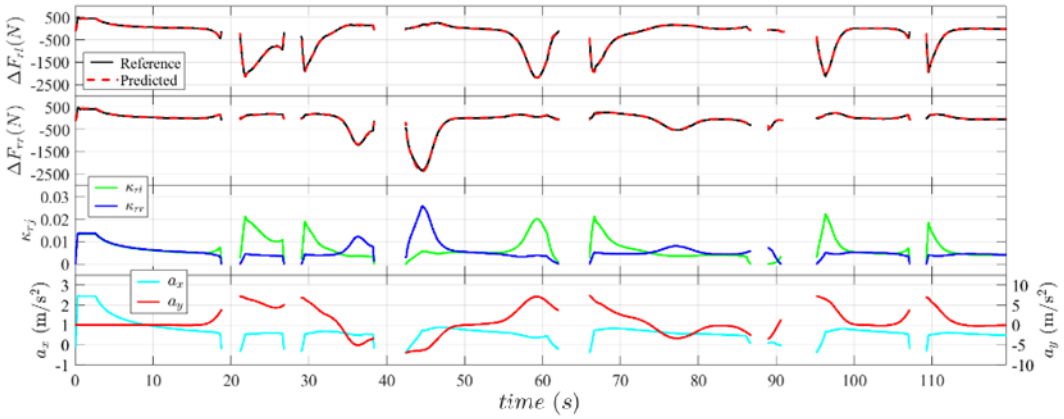


Figure 5.26 | Lap 2 | NN2: predicted $\Delta\hat{F}_{rj}$ vs reference ΔF_{rj} traction force correction.

It can be seen from the figures that the correction achieved is the greater slip angles of the two wheels in the case of the NN1 and the greater lateral acceleration a_y in the case of the NN2. This is consistent with the fact that Pacejka's formula is only capable of accurately predicting the traction force in the case of pure longitudinal slip, i.e., not in curves where the quantities mentioned assume non-zero values.

5.3.2 | Overall estimated traction force

Figures 5.27-5.30 show the overall prediction, the sum of the longitudinal force calculated by means of Pacejka's formula, and the correction obtained by means of NN1 and NN2.

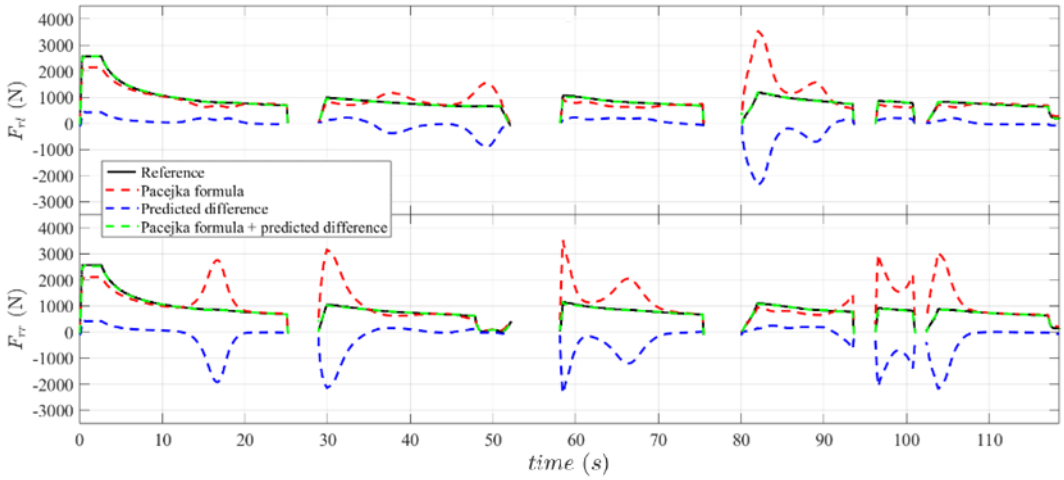


Figure 5.27 | Lap 1 | Traction force calculated with Pacejka's formula corrected by the NN1.

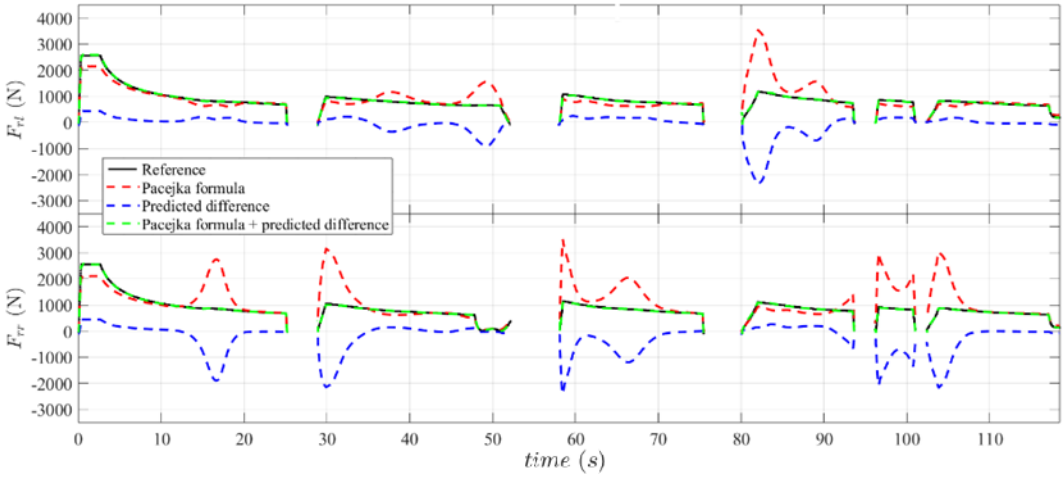


Figure 5.28 | Lap 1 | Traction force calculated with Pacejka's formula corrected by the NN2.

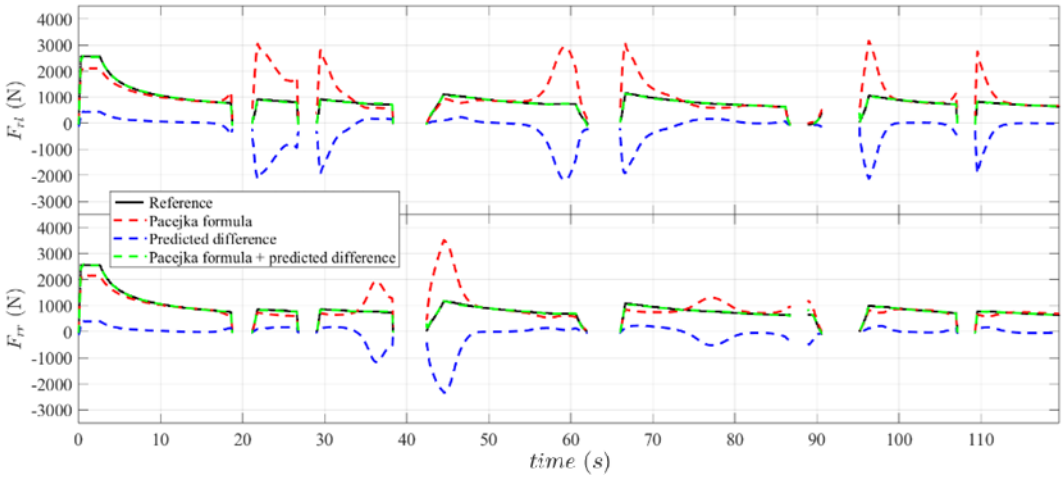


Figure 5.29 | Lap 2 | Traction force calculated with Pacejka's formula corrected by the NN1.

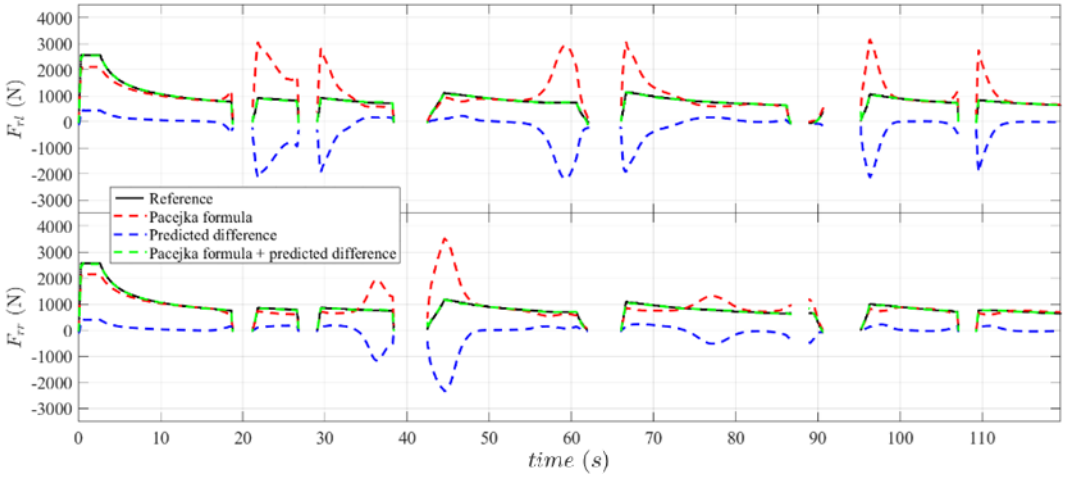


Figure 5.30 | Lap 2 | Traction force calculated with Pacejka's formula corrected by the NN2.

Table 5.10 shows the Root Mean Square Error (RMSE) for the two laps and the two tires for each predictor compared with the analogous values obtained exclusively using Pacejka's formula.

Lap	Force	Pacejka's formula	Pacejka's formula with NN1 correction	Pacejka's formula with NN2 correction
1	F_{rl} (N)	416.746	13.377	13.487
	F_{rr} (N)	652.468	18.031	14.620
2	F_{rl} (N)	653.514	18.147	17.587
	F_{rr} (N)	431.861	12.941	17.187

Table 5.10 | RMSE between predicted and reference forces.

5.4 | Prediction of the sideslip angle using dynamic Neural Networks

5.4.1 | Test maneuvers

The two NNs were tested by carrying out two maneuvers that prompted a change in the sideslip angle. The predicted sideslip angle relative to the reference is shown in Figure 5.31-5.34.

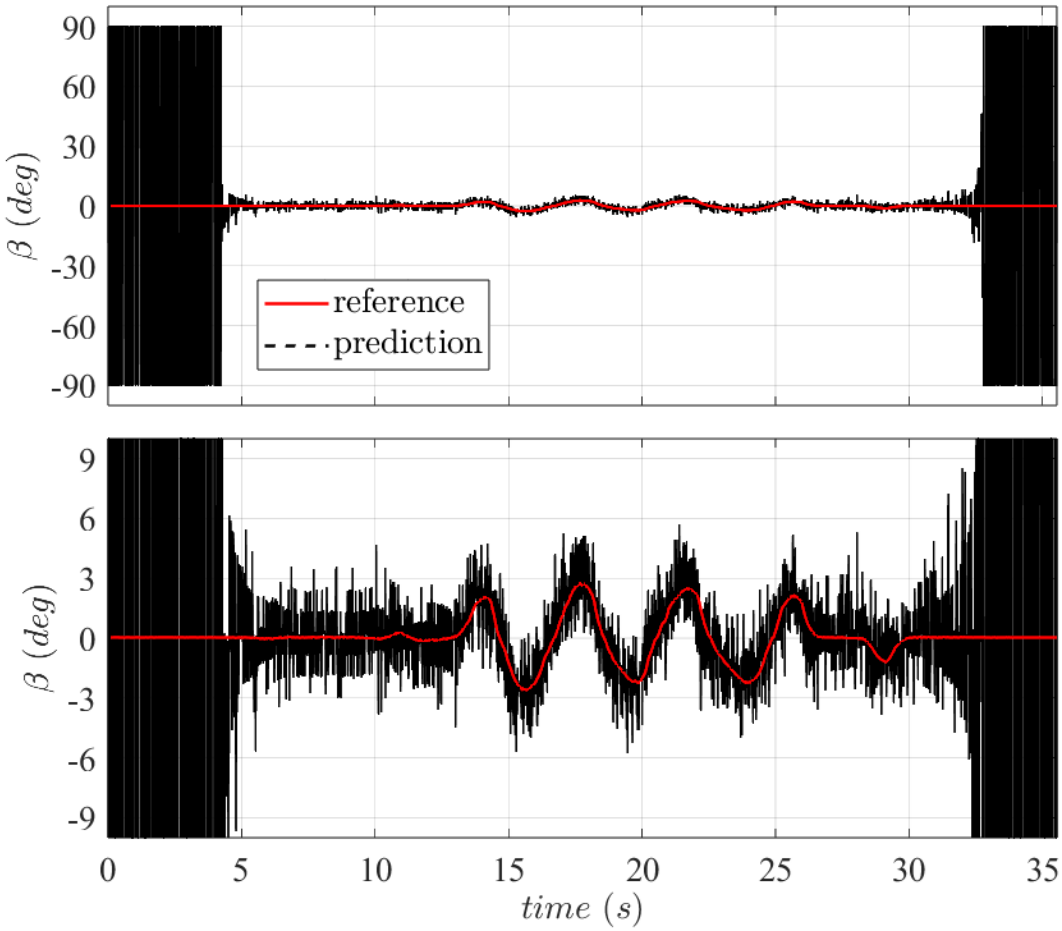


Figure 5.31 | Reference vs predicted (NN1) sideslip angle | Slalom.

- *Slalom*: the vehicle moves through a series of cones in a close pattern. Slalom tests are frequently used in automotive testing and assessment, such as analyzing a vehicle's handling qualities or testing suspension system performance. In this case, the cones were positioned at 17 m and the vehicle was advancing at a speed of 30 $\frac{km}{h}$.

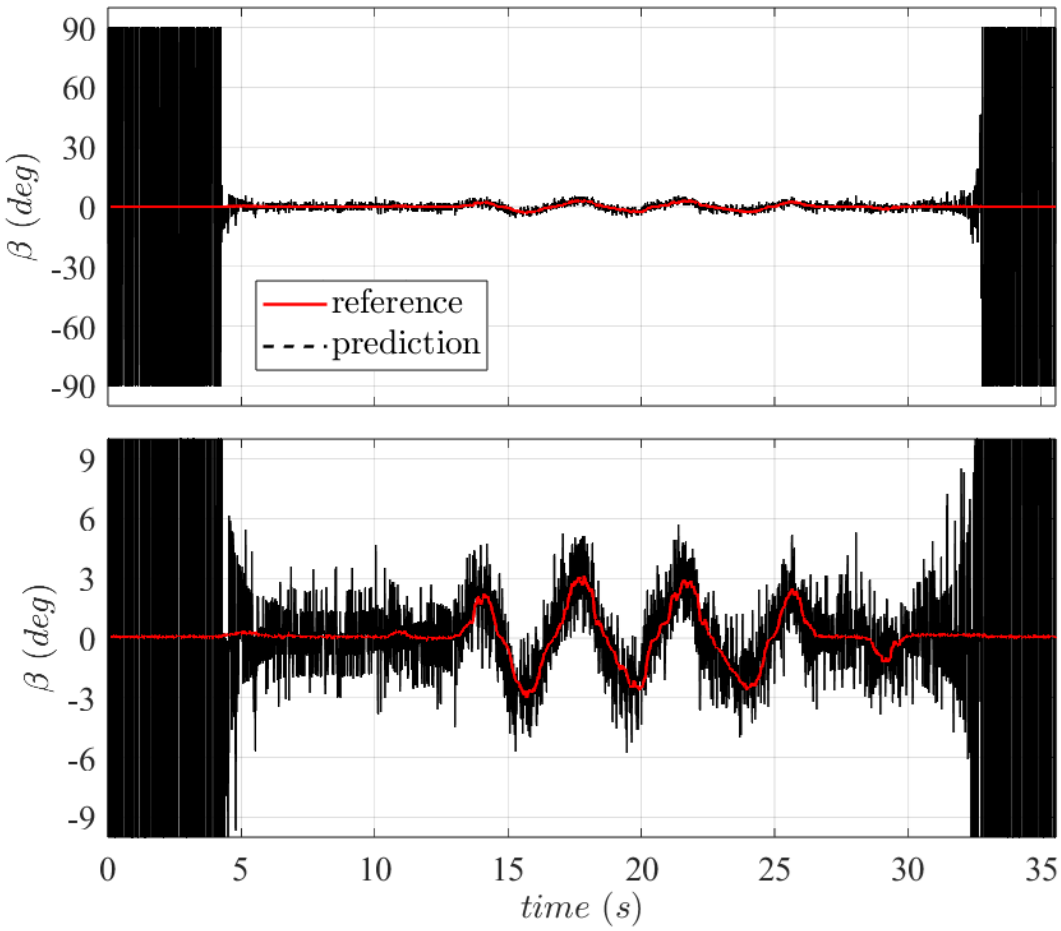


Figure 5.32 | Reference vs predicted (NN2) sideslip angle | Slalom.

- *ISO double lane change*: it is a vehicle dynamics driving test or maneuver used to evaluate a vehicle's handling capabilities during rapid lateral transitions, such as an emergency lane change. The ISO double lane change is a standardized test that is carried out in accordance with the International Organisation for Standardisation (ISO) criteria in ISO 3888-2. In this case, the vehicle was advancing at a speed of 30 $\frac{km}{h}$.

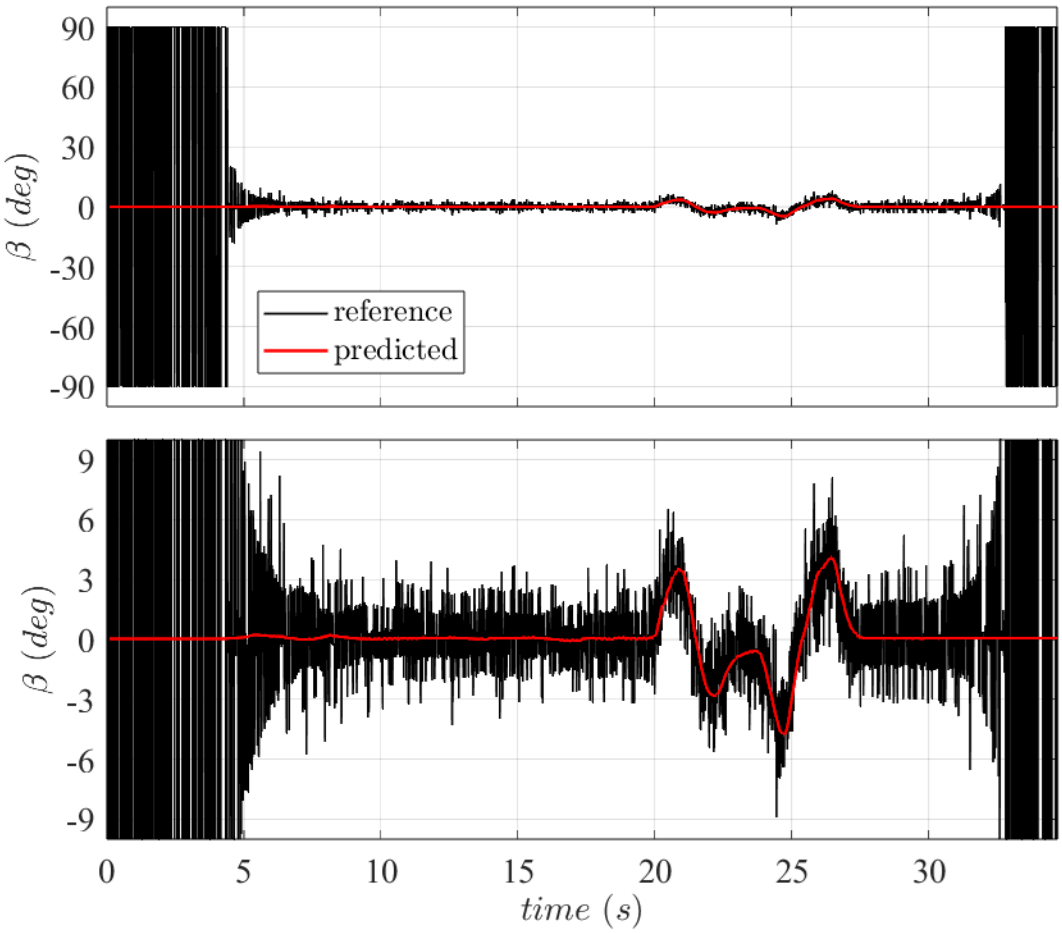


Figure 5.33 | Reference vs predicted (NN1) sideslip angle | ISO double lane change.

Unlike the training dataset and the validation dataset, in the test dataset, the initial and final samples in which the longitudinal velocity was less than 10 $\frac{km}{h}$ were not removed. This is due to the desire to analyze the behavior of the NN operation also during the initial and final transients.

In each Figure, the first graph shows the calculated reference values in their entirety, including the initial and final phases in which the sideslip angle, calculated according to Equation (3.1), oscillates numerically between -90° and $+90^{\circ}$, the second graph shows an enlargement in the area of interest.

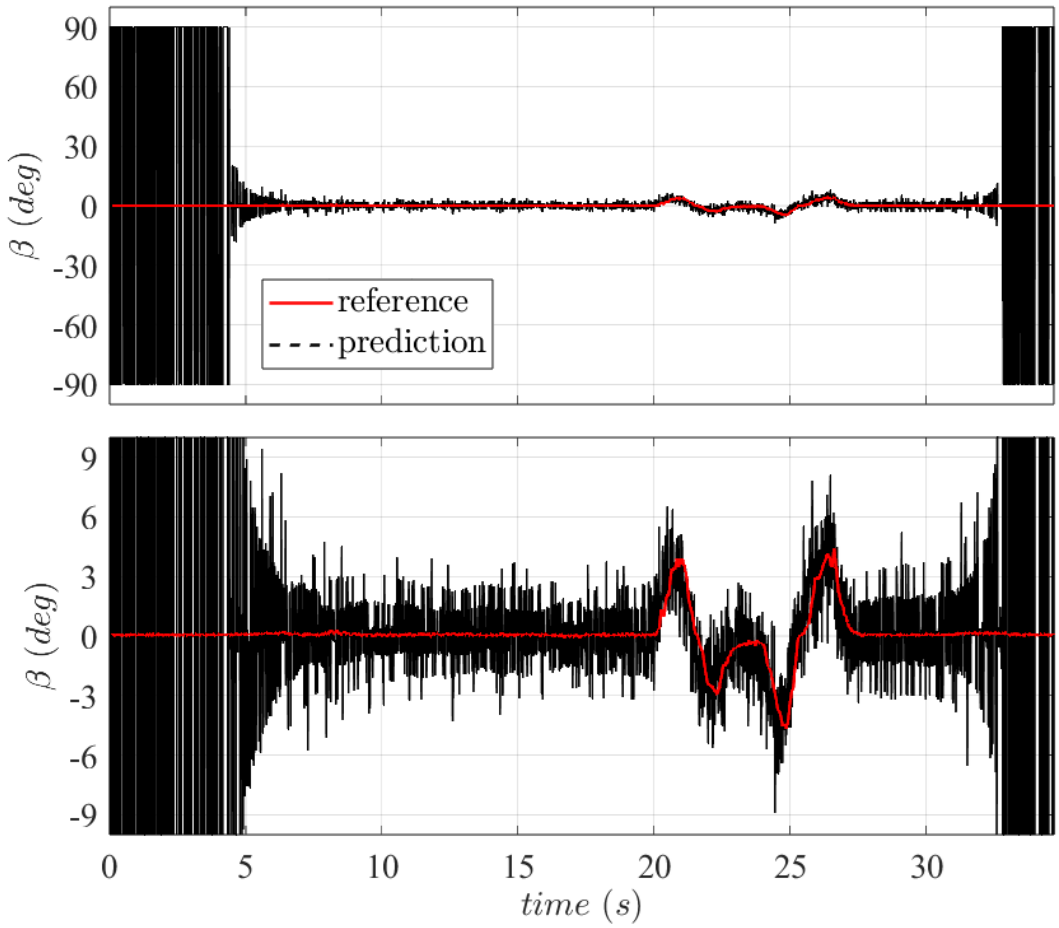


Figure 5.34 | Reference vs predicted (NN2) sideslip angle | ISO double lane change.

5.4.2 | Remarks

There are two important remarks to be made for both NNs:

- Although the calculated sideslip angle values oscillate between -90° and $+90^{\circ}$ in the initial and final transients and no information was given during training for these phases, the NNs are able to correctly predict a null value.
- The NNs return a signal unaffected by noise during maneuvering.

The reason for this behavior lies in the network's ability to detect dynamic links between inputs and output. The oscillating values during transients are exclusively due to numerical motivation, the velocity v_x tending to 0 in the denominator of (3.1) at the beginning and end of

maneuvers. The noise in the reference, on the other hand, is due to the presence of noise in the quantities used to calculate the reference sideslip angle β. There can be several causes of noise: electromagnetic interference [106], radio frequency interference, impedance mismatch [107], thermal noise [108], etc. Both numerical oscillations and noise are unrelated to vehicle dynamics. The initial and final oscillations cannot be predicted by the NN as they were excluded from the training dataset. Noise, instead, was not removed from the training dataset, which uses unfiltered signals. However, as there is no deterministic cause-effect link between the noise in the inputs and the noise in the output, the NN cannot reproduce it in the predictions. In contrast, dynamic links are dictated by physical laws that relate to inputs and outputs. It has been shown that NNs are able to detect such ties by returning a null value of the sideslip angle in transients and a filtered value during maneuvering.

Table 5.11 shows the RMSE for the two maneuvers and the two NNs. The results are shown for the signal without the initial and final samples in which the velocity is less than 10 $\frac{km}{h}$ as such samples, affected by numerical errors, corrupt the result.

	Manoeuvre	RMSE (deg)
NN1	Slalom	1.4090
	ISO double lane change	1.5747
NN2	Slalom	1.4319
	ISO double lane change	1.6142

Table 5.11 | RMSE between predicted and reference sideslip angle for the test datasets.

It must be pointed out that RMSE are calculated with respect to a reference signal that is noisy, so the error obtained is partially due to estimation error, partially due to noise.

To obtain a more reliable metric that does not take into account measurement noise in the reference, it was decided to calculate a filtered reference using a Savitzky–Golay (SG) filter [109]. The SG filter is able to effectively reduce the noise in the signal while maintaining the main characteristics of the original signal. This is particularly useful when there is a need to remove noise from a signal without compromising useful information. The SG filter is less sensitive to sudden changes in the signal than the moving average filter [110]. This makes it particularly suitable for applications where it is important to maintain the overall shape of the signal. Compared to a low-pass filter, it is able to preserve the temporal characteristics of the signal by not introducing delays [111]. The disadvantage is that it cannot be used in real-time due to the need for future data to calculate the current output.

Figures 5.35 and 5.36 show the FFTs of the original reference signal and the reference signal filtered through a Savitzky–Golay filter. The two FFTs are shown both in their entire frequency range and in the limited range from $0\,Hz$ to $5\,Hz$. A considerable amplitude attenuation for high frequencies and an unchanged amplitude for low frequencies can be seen.

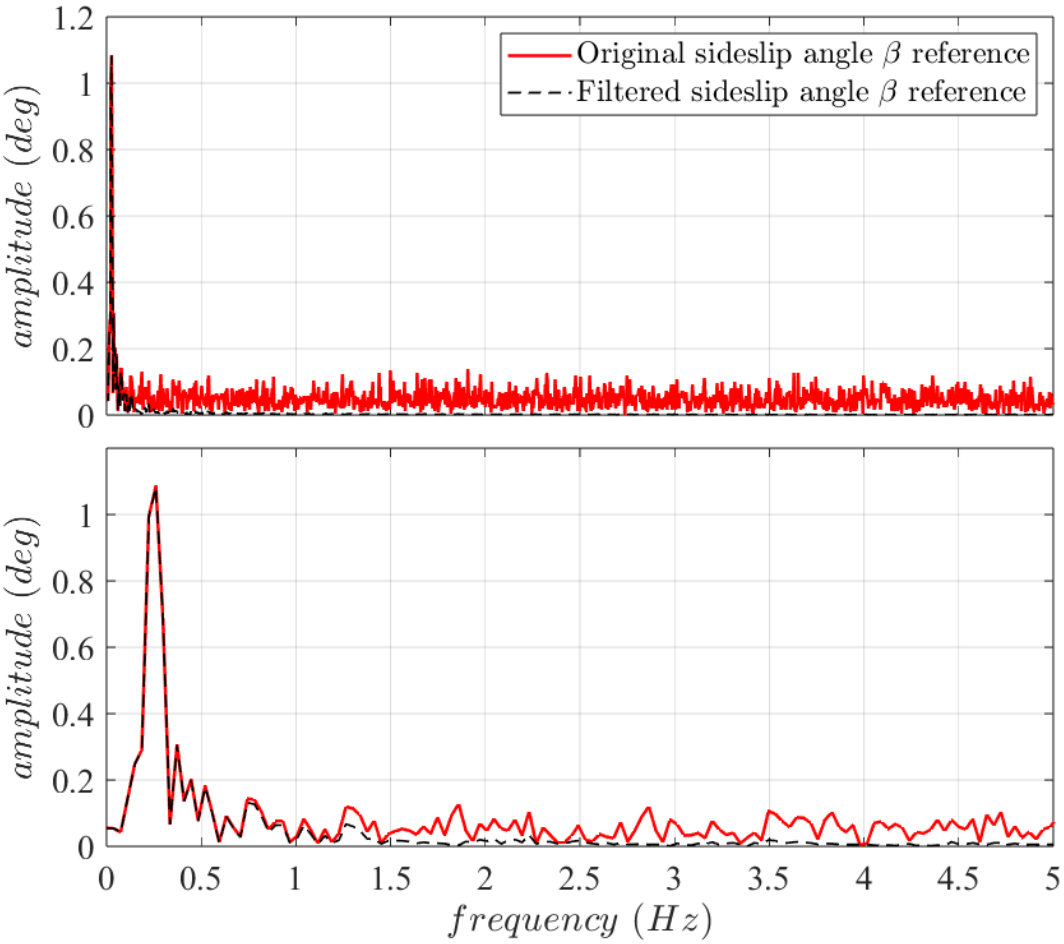


Figure 5.35 | FFTs of the original and filtered sideslip angle β reference | Slalom.

Having filtered out the reference signal makes it possible to calculate more reliable metrics, less affected by noise, to evaluate the ability of the estimator to predict the actual signal, and to adequately compare the performance of this estimator to those already presented in the literature.

Table 5.12 shows the RMSE for the two maneuvers and the two NNs using an SG filtered reference signal.

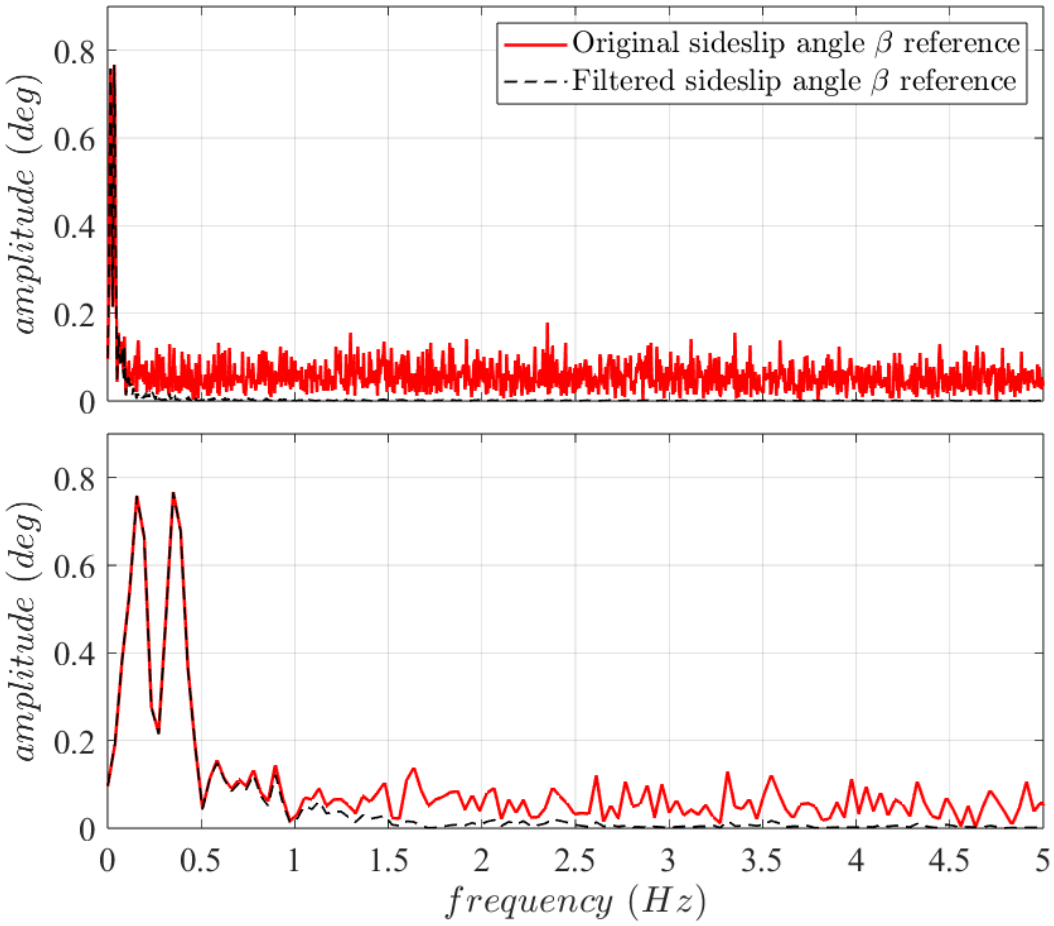


Figure 5.36 | FFTs of the original and filtered sideslip angle β reference | ISO double lane change.

	Manoeuvre	RMSE (deg)
NN1	Slalom	0.3292
	ISO double lane change	0.3195
NN2	Slalom	0.4089
	ISO double lane change	0.4594

Table 5.12 | RMSE between predicted and reference sideslip angle for the test datasets using a Savitzky–Golay filtered reference signal.

5.5 | Measurement of unsprung mass displacement of road vehicles through a model-based virtual sensor

The outputs provided by the virtual sensor are compared with the experimental reference data in this Section. The virtual sensor was tested by

carrying out an additional lap on track 7. The choice was made to test the estimator on different data than those with which the parameters were identified to ensure that there was no overfitting of the data used for its development. The comparisons between the estimated displacements $\tilde{z}_{ij}$ related to the four unsprung masses and the experimental displacements are portrayed in Figures 5.37-5.40.

Table 5.13 shows the RMSE [112] of $\tilde{z}_{ij}$ concerning the measurements obtained with the physical sensor.

	$\tilde{z}_{FL}$ (m)	$\tilde{z}_{FR}$ (m)	$\tilde{z}_{RL}$ (m)	$\tilde{z}_{RR}$ (m)
RMSE	0.0156893	0.0131200	0.0133240	0.0120560

Table 5.13 | RMSE between predicted and reference displacements of virtual sensor test maneuver.

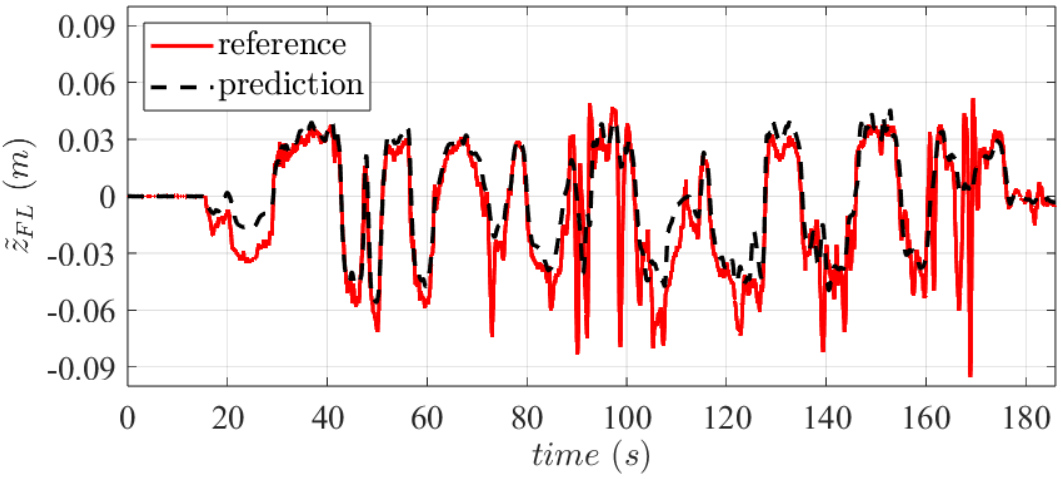


Figure 5.37 | Relative displacement of the front-left displacement $\tilde{z}_{FL}$ versus time t.

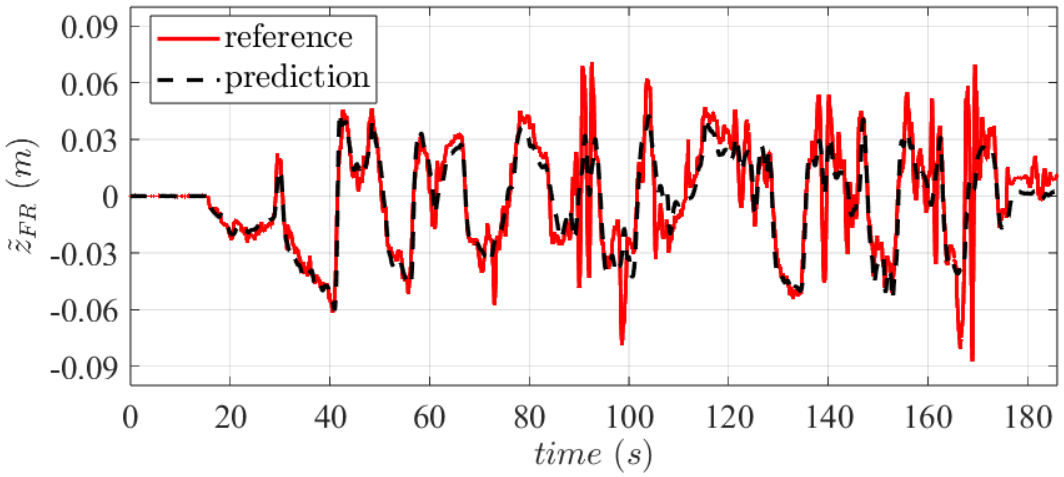


Figure 5.38 | Relative displacement of the front-right displacement $\tilde{z}_{FR}$ versus time t.

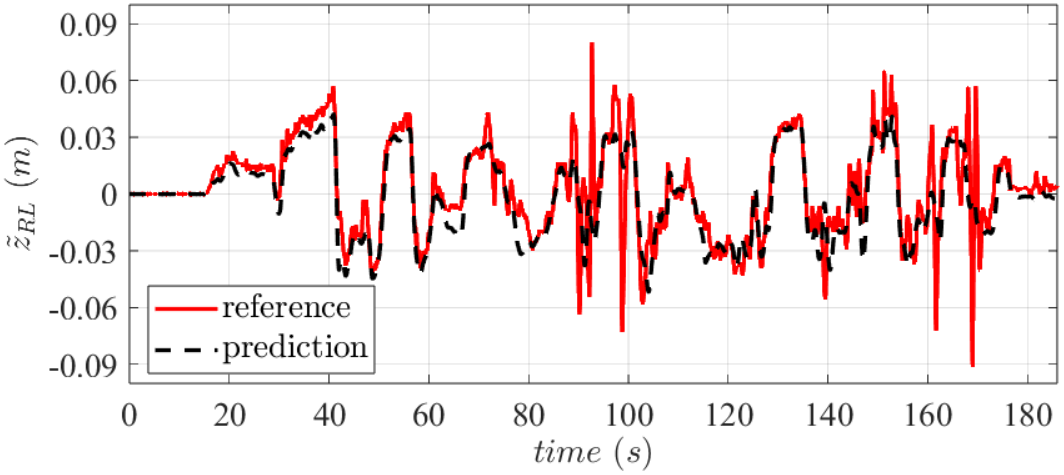


Figure 5.39 | Relative displacement of the rear-left displacement $\tilde{z}_{RL}$ versus time t.

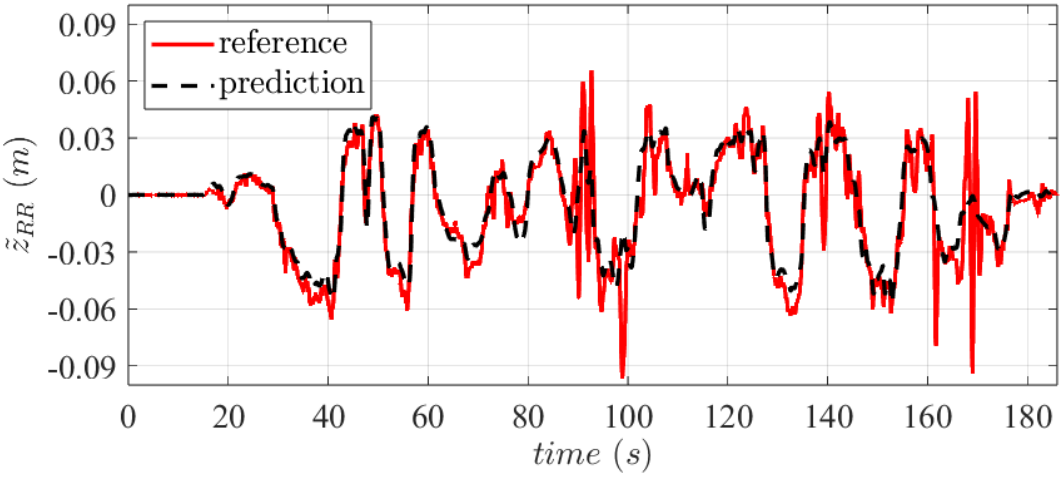


Figure 5.40 | Relative displacement of the rear-right displacement $\tilde{z}_{RR}$ versus time t.

The dynamics characterizing the suspension are effectively captured, as shown in the correspondence between the peaks and valleys of the estimated and measured signals. This correspondence is not present only in areas where the inputs from the road are relevant. The virtual sensor can convert the information from the accelerations into the corresponding displacements of the unsprung masses. These accelerations are the result of chassis dynamics and not inputs from the road. The present offsets and nonreplicated transients are due to inputs from the road and are beyond the scope of this study.

Figure 5.41 shows the histograms of the estimation error. Each bar has a width of 0.005 m. The error is distributed around zero.

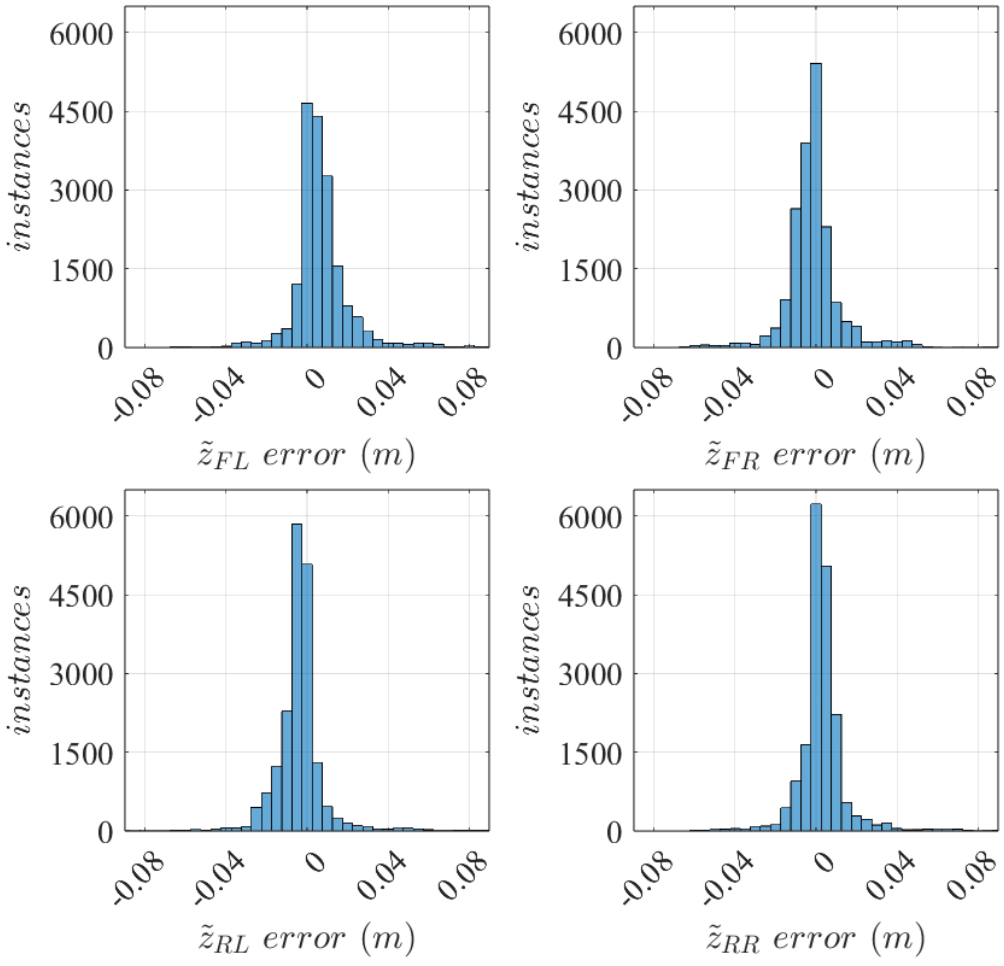


Figure 5.41 | Histograms of the error between the predicted and measured unsprung mass relative displacement.

The low estimation error demonstrates the effectiveness of the virtual sensor proposed in this paper to measure the relative displacement of unsprung masses belonging to a road vehicle.

5.6 | Neural Network-based virtual measurement of road vehicle wheel displacements

The estimator was tested by carrying out additional acceleration, braking, and steering maneuvers to stress the suspension. Figure 5.42 shows the comparison between the reference and estimated data. The presented results reveal the suitability of the proposed approach based on NNs for estimating the vertical displacements of the wheels with respect to the chassis. The RMSEs between reference and predicted displacements for the four wheels are shown in Table 5.14, which confirms the goodness of the proposed estimation technique.

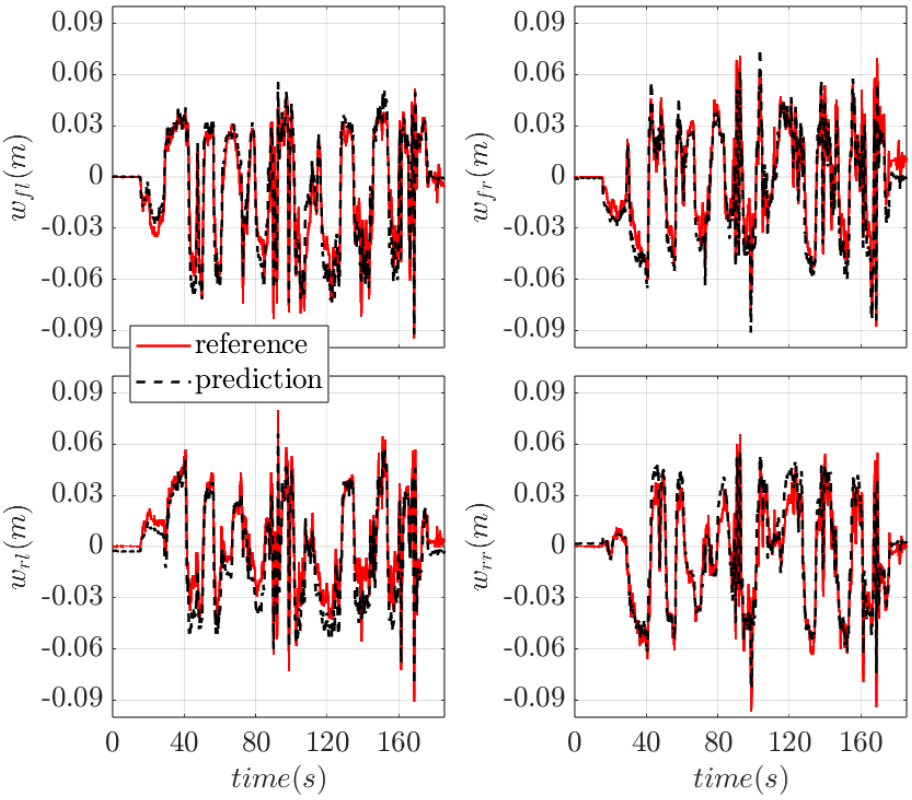


Figure 5.42 | Predicted wheel displacements relative to the reference.

	$w_{fl}\ (m)$	$w_{fr}\ (m)$	$w_{rl}\ (m)$	$w_{rr}\ (m)$
RMSE	0.0089	0.0092	0.0101	0.0087

Table 5.14 | RMSE between predicted and reference displacements.

5.7 | Enhancment of the wheel vertical displacement estimation in road vehicles through integration of model-based estimator with artificial intelligence

In this Section, the estimator's outputs are compared to the experimental reference data. The estimator was tested by performing a maneuver that included accelerations, decelerations, and steering on a non-flat road. To make sure there wasn't any overfitting of the data employed for the NN's development, it was decided to test the estimator on another set of data than the ones used to train it. The comparisons between the estimated and experimental displacements are depicted in Figures 5.43 and 5.44.

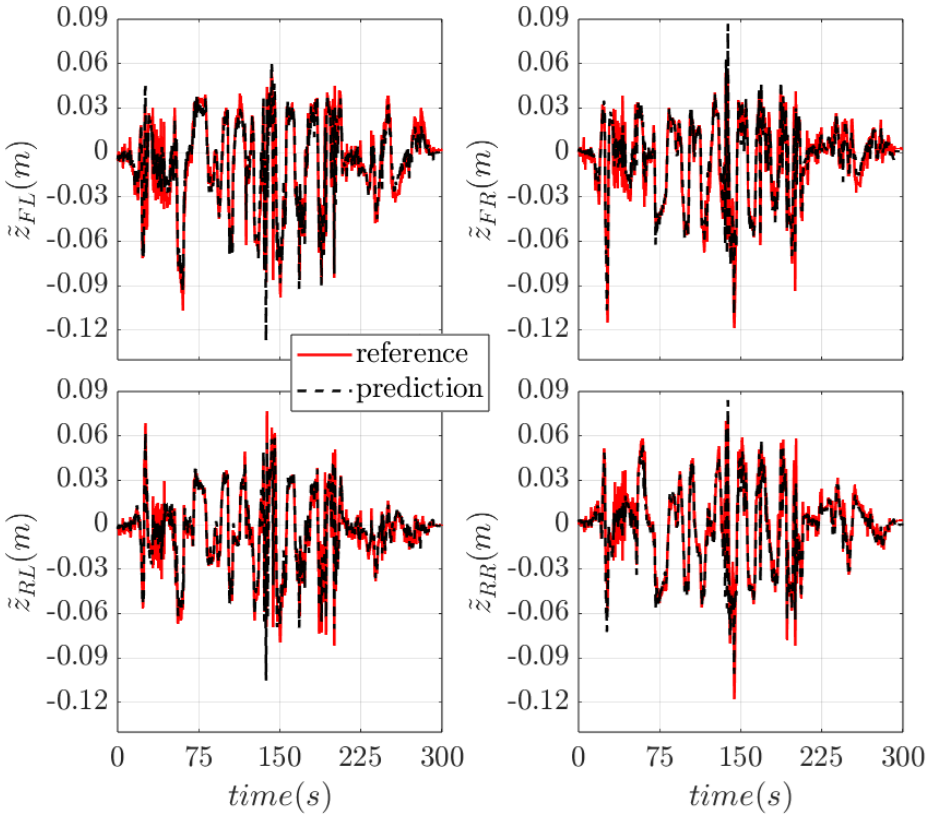


Figure 5.43 | Comparison of reference and estimated displacement.

The driver took bends and crossed roads with bumps, putting stress on the suspension. Figure 5.43 shows that the overall estimator accurately predicts the vertical displacement of the wheels throughout the maneuver. Figure 5.44 shows a detail of the maneuver in which it is evident that the model-based estimator alone is not able to accurately predict vertical wheel displacements under all conditions. In particular, it is unable to predict them in the presence of road bumps as it contains no information on their presence. Vertical wheel displacement can be accurately estimated when it is stressed by longitudinal and lateral accelerations of the vehicle, which are the only acceleration inputs to the model. These cause the vehicle to pitch and roll, resulting in compression and extension of the suspension. The AI-based estimator, on the other hand, using only vertical acceleration, is only able to predict vertical wheel displacement in the presence of road bumps. In contrast to the model-based estimator, the AI-based estimator is not able to estimate the vertical displacement of the wheels when stressed by longitudinal and lateral accelerations as these are not present as input. The overall estimator, combining the characteristics of both, accurately predicts the signal of interest under all conditions, as can be seen by the reproduction of the peaks and valleys of the reference signal.

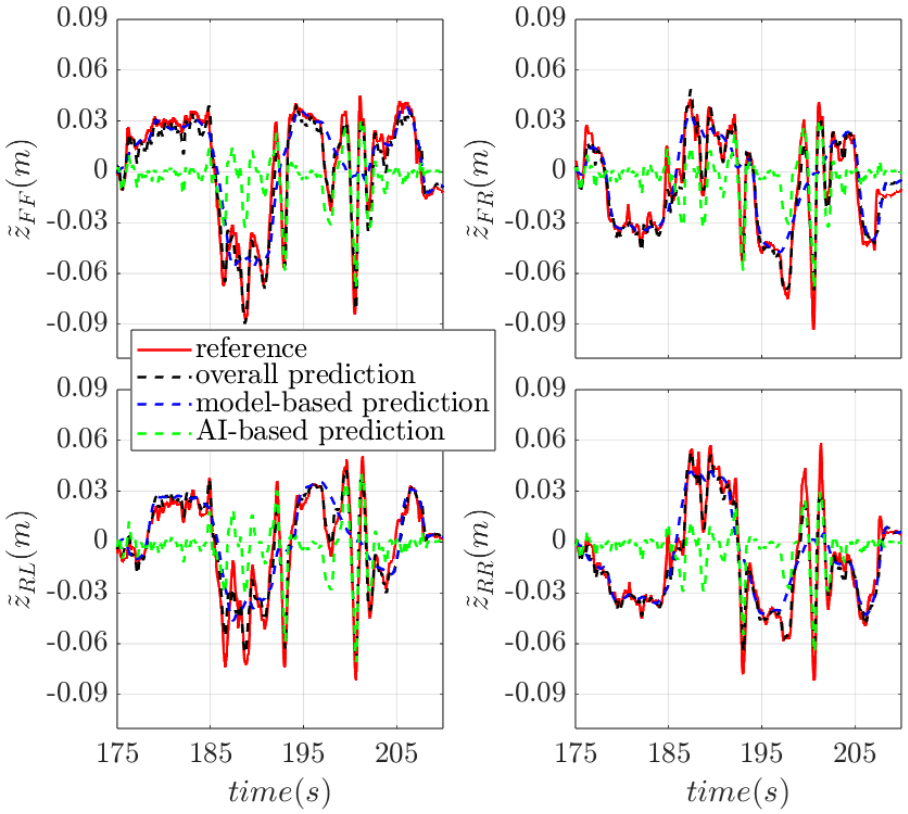


Figure 5.44 | Comparison of reference and estimated displacements, detailed with contributions from model-based and AI-based estimation.

Utilizing the same hardware configuration employed for offline training of the NN, subsequent estimation during the test maneuver, including 30000 samples, was achieved in 1.761 s. This implies that the estimator produced all four outputs approximately once every $5.87 \cdot 10^{-5}$ s, corresponding to a frequency of approximately 17000 Hz. Notably, this frequency surpasses the input rate of 100 Hz. Such rapid output generation, obtained with a low-range computer, demonstrates the technical feasibility of adopting the proposed solution in real-time application. The actual real-time application depends on the hardware employed.

Table 5.15 shows the RMSE [112] with and without AI compensation and the relative percentage improvement in the estimate.

	$\tilde{z}_{FL}$	$\tilde{z}_{FR}$	$\tilde{z}_{RL}$	$\tilde{z}_{RR}$
RMSE (m) of model-based estimation	0.0098884	0.0090696	0.0086634	0.0076300

RMSE (m) with AI-based compensation	0.0065767	0.0060694	0.0052854	0.0052882
Percentage improvement (%)	33.491	33.080	38.991	30.692

Table 5.15 | RMSE and percentage improvement between the predicted and reference displacements of virtual sensor test maneuvers.

Based on the results obtained, it can be seen that although the model-based estimator is able to provide good results, it shows inaccuracies due to inputs from the road. The presence of an AI component, which works in parallel with the model-based one, effectively compensates for this estimation error, allowing the RMSE to decrease by more than 30% compared to the model-based estimator. In the overall estimator, the dynamics characterizing the suspension are effectively captured, as shown in the correspondence between the peaks and valleys of the estimated and measured signals.

5.8 | Addressing integration challenges in vehicle roll and pitch estimation using Neural Networks

The NN was tested by performing an ISO Double Lane Change maneuver with Carmaker. OU noises were superimposed on the longitudinal and lateral accelerations, used as input to the network, and roll and pitch rate measures, to verify the robustness of the virtual sensor to noisy signals. Figures 5.45 and 5.46 show the comparison between the reference roll and pitch angles, the one estimated by the NN and the one obtained by integrating the roll and pitch rates.

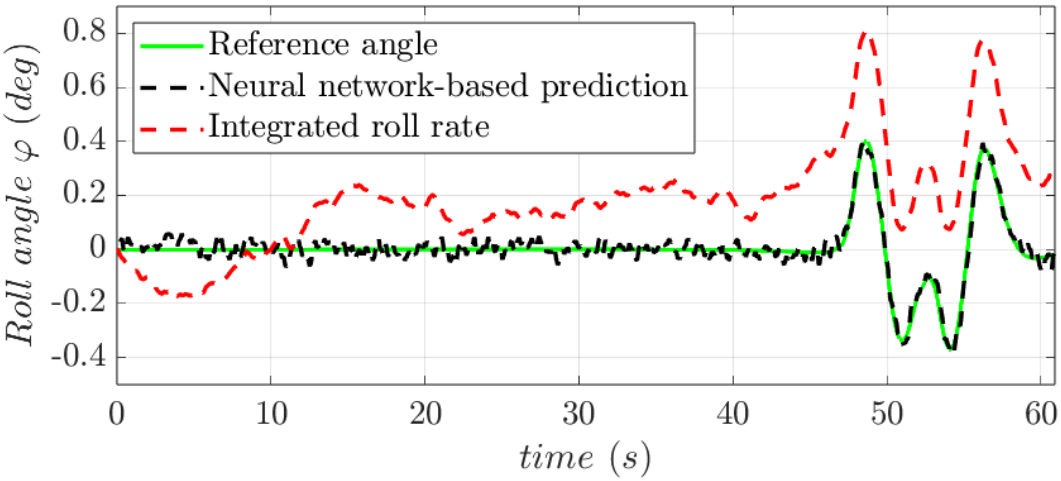


Figure 5.45 | Reference vs predicted roll angle φ.

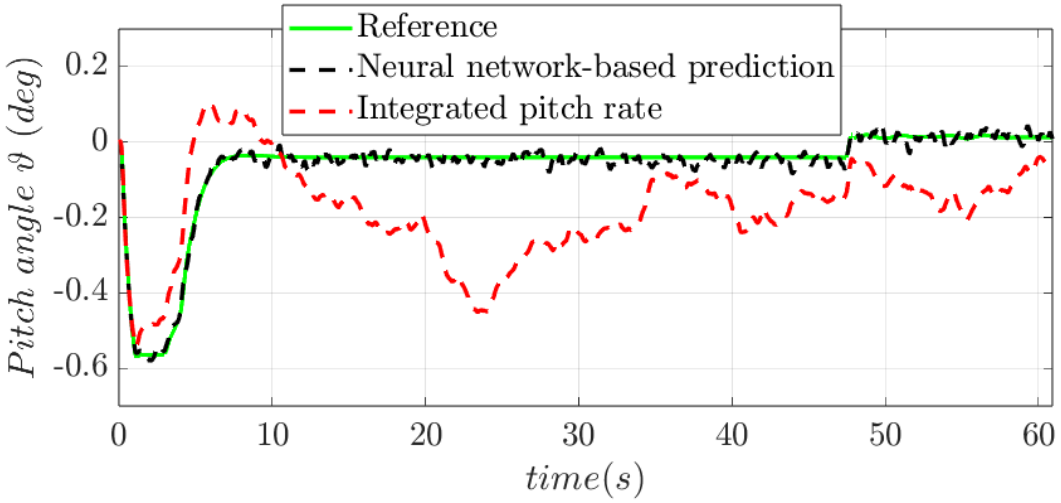


Figure 5.46 | Reference vs predicted pitch angle ϑ.

The NN was able to accurately estimate vehicle roll and pitch, despite the noise superimposed on the inputs. The integration of roll and pitch rate, on the other hand, does not allow accurate estimation as the integration of the signals results in a progressive accumulation of error over time. This does not happen in the case of NNs, which instead calculate their outputs through algebraic operations and not by integrating the inputs. The virtual sensor returned a Root Mean Square Error between reference and prediction for roll and pitch of 0.0137° and 0.0214°, demonstrating accuracy in estimation.

5.9 | Improving roll reduction in road vehicles on uneven surfaces through the fusion of proportional control and reinforcement learning

To test the controller, an additional slalom test was carried out with different road irregularities, as shown in Figure 5.47.

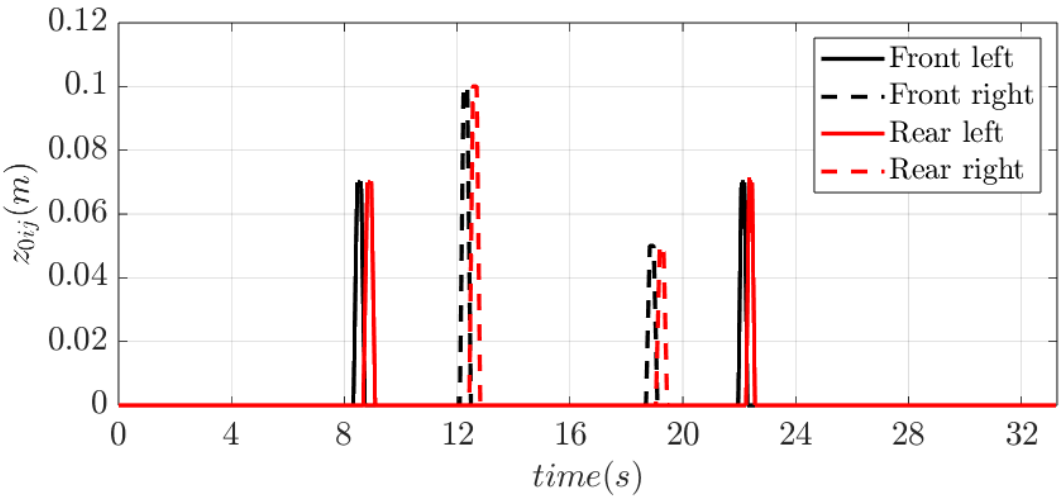


Figure 5.47 | Vertical displacement of the road wheel contact points due to road bumps in the case of the test maneuver.

Figure 5.48 shows the longitudinal and lateral acceleration of the vehicle realized during the test maneuver.

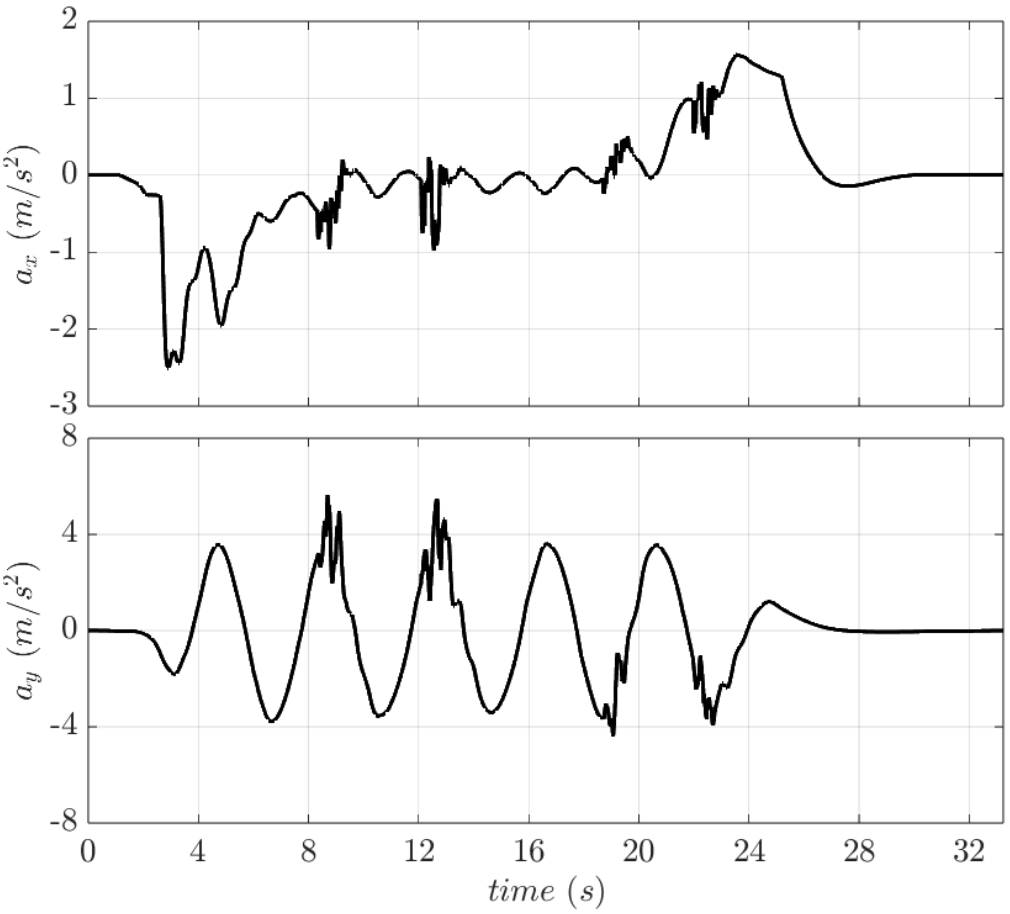


Figure 5.48 | Longitudinal and lateral accelerations of the vehicle of the performed slalom maneuver in the case of the test maneuver.

Figure 5.49 shows a comparison between roll angle in the absence of control, in the presence of proportional control, and in the presence of proportional control with simultaneous correction by Reinforcement Learning in the case of the test maneuver.

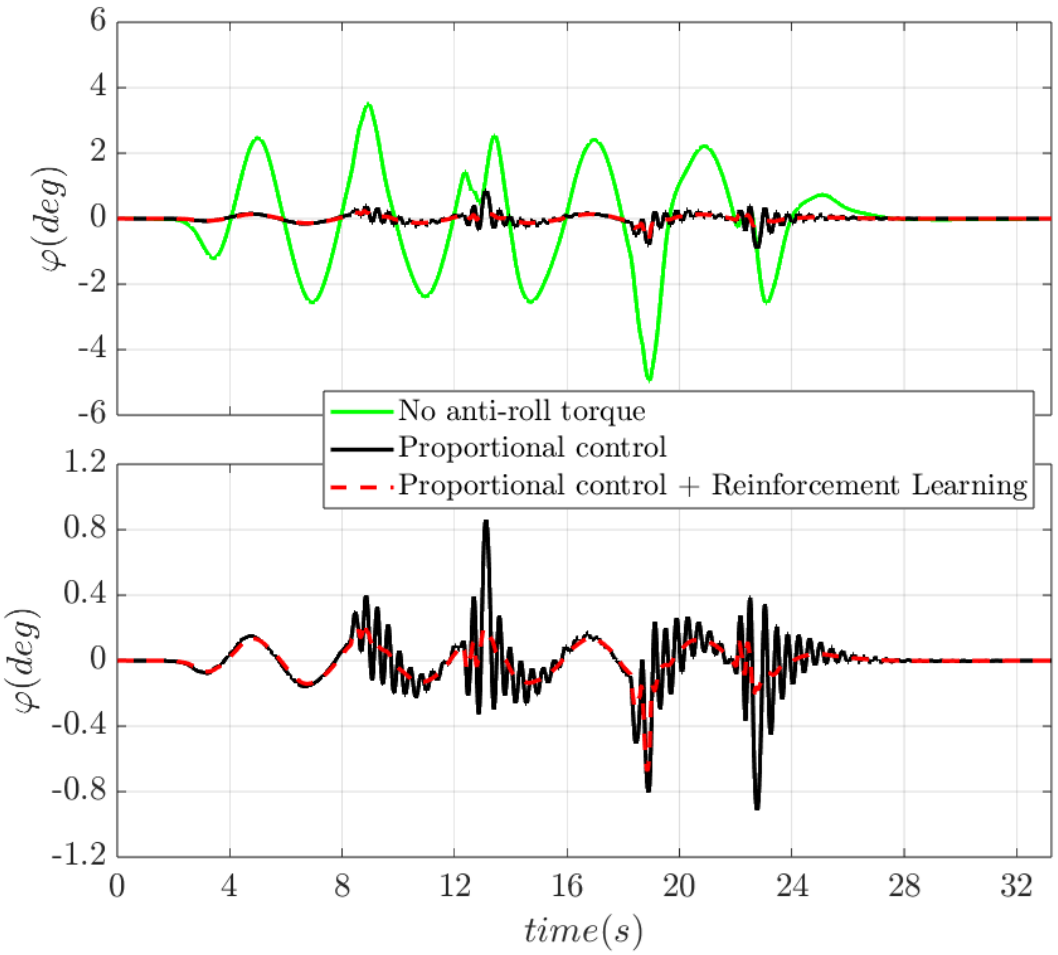


Figure 5.49 | Comparison of roll angle in the absence of anti-roll torque, in the presence of proportional control, and in the presence of proportional control and correction by Reinforcement Learning in the case of the test maneuver.

Table 5.16 shows the RMSE of the roll with respect to the null reference in the absence of control, in the presence of proportional control, and in the presence of proportional control with simultaneous correction by Reinforcement Learning in the case of the test maneuver.

	Proportional control	Proportional control + RL
RMSE (deg)	0.1626	0.0935

Table 5.16 | RMSE of the roll with respect to the null reference in the absence of control, in the presence of proportional control, and in the presence of proportional control with simultaneous correction by Reinforcement Learning in the case of the test maneuver.

The obtained results underscore the effectiveness of the proposed combined control strategy in mitigating vehicle roll, particularly in challenging conditions. The incorporation of road irregularities, as seen in Figures 4.54 and 5.47, serves as a real-world test for the controller's robustness.

While proportional control alone exhibits a reduction in roll, the oscillations induced by road bumps persist. In contrast, the simultaneous correction by RL, as seen in Figures 4.57 and 5.49, significantly diminishes these oscillations, highlighting the complementary nature of the two control methods.

With the RMSE metric, Tables 4.23 and 5.16 quantify the performance improvement. The substantial decrease in RMSE under the combined proportional control and RL strategy signifies a remarkable enhancement in roll control precision.

6 | CONCLUSIONS AND FUTURE DEVELOPMENTS

This doctoral thesis has explored the estimation and control of vehicle dynamics, with a focus on the integration of model-based techniques and artificial intelligence (AI). The research has demonstrated how this synergy can provide innovative solutions to the complex challenges associated with vehicle dynamics.

- *Tire-road interaction forces*:
  - *AI-based estimation*: One of the central contributions of this work is the development of a novel approach for estimating tire-road interaction forces. The proposed model, based on a multi-output neural network, uses longitudinal and lateral accelerations measured by an Inertial Measurement Unit (IMU), reducing both the costs and complexity of implementation by leveraging affordable sensors that are already present in commercial vehicles. Through the optimization of hyperparameters and the use of time-series samples from previous instants, the approach has proven to be particularly effective in predicting the forces involved, as validated by experimental tests and comparisons with reference models.
  - *Model and AI combined estimation*: Another important advancement presented in this thesis is the methodology for estimating tire-road interaction forces using Pacejka's formulas. This method does not rely on sensors placed inside the wheels but instead

utilizes indirect measurements and mathematical models, such as a Central Difference Kalman Filter and a double-track vehicle model.

The integration of neural networks with well-established models like the Pacejka formula has also led to notable improvements in traction force estimation during cornering. The combined use of AI-based techniques and physical models has reduced the estimation errors typically encountered when using Pacejka's formulas alone

- *Sideslip angle*:
  - *AI-based estimation*: Moreover, the thesis addressed the challenge of predicting the sideslip angle, a critical parameter for vehicle stability, especially in autonomous driving. The use of dynamic neural networks, fed by cost-effective sensor data (e.g., lateral acceleration, yaw rate), has proven to be an effective strategy for accurate sideslip angle prediction without the need for expensive sensors. This innovative approach reduces noise and oscillations in the output, further enhancing the reliability of vehicle stability control systems.
- *Unsprung masses*:
  - *Model-based estimation*: The model-based virtual sensor for unsprung mass displacement utilizes a 7-degree-of-freedom vehicle model, offering a precise understanding of how vehicle accelerations affect suspension displacements.
  - *AI-based estimation*: The neural network-based virtual sensor enables simultaneous estimation of vertical displacements for all

four wheels without physical modeling, simplifying the control and monitoring process.

- *Model and AI combined estimation*: Integrating AI with traditional model-based estimators significantly improves estimation accuracy, particularly when the system is influenced by road inputs.

- *Roll and pitch*:
  - *AI-based estimation*: The proposed approach eliminates the need for roll rate and pitch rate integration, which can suffer from drift and noise accumulation. The key advantage of the proposed method lies in its avoidance of signal integration. Unlike traditional methods, the neural network utilizes algebraic calculations, leading to consistent estimations over time, even under noisy conditions.
  - *AI-based control*: A hybrid control strategy combines proportional control with reinforcement learning (RL) for improving vehicle roll reduction, especially on uneven road surfaces. This innovative approach integrates active anti-roll bars controlled by a proportional feedback controller and an RL-based system, demonstrating significant improvements in vehicle stability and comfort. The RL-based controller effectively compensates for the limitations of proportional control alone, particularly in reducing oscillations caused by road irregularities.

The research presented in this thesis offers several avenues for future exploration and development:

- Incorporating data from external sensors (e.g., road surface sensors, weather conditions) to enhance the accuracy of dynamic estimators.
- Applying the estimation and control methodologies to SAE Level 4 and 5 autonomous vehicles, where dynamic control systems must handle all driving scenarios without human intervention.
- Developing adaptive AI models capable of real-time learning from vehicle behavior to optimize control for various environments and driving styles.
- Extending experimental validation to real-world driving scenarios, including urban environments, highways, and off-road conditions, to assess the robustness of the proposed solution.

REFERENCES

[1] K. B. Singh, "Literature review and fundamental approaches for vehicle and tire state estimation," *Vehicle System Dynamics,* 2018.

[2] P. Maybeck, "The Kalman Filter: An Introduction to Concepts," *Autonomous Robot Vehicles,* 1990.

[3] P. Cunningham, M. Cord and S. J. Delany, "Supervised Learning," in *Machine Learning Techniques for Multimedia*, Springer, 2008.

[4] R. Caruana and A. N. Mizil, "An empirical comparison of supervised learning algorithms," in *Proceedings of the 23rd international conference on Machine learning*, 2006.

[5] S. Xiaogang, Y. Xin and T. L. Chih, WIREs Computational Statistics, 2012.

[6] D. Kleinbaum and M. Klein, Logistic Regression, Springer, 2010.

[7] L. Rokach and O. Maimon, "Decision Trees," in *Data Mining and Knowledge Discovery Handbook*, Springer, 2005.

[8] L. Breiman, "Random Forests," *Machine Learning,* 2001.

[9] M. Hearst, Dumais, Osuna, J. Platt and B. Schölkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications,* 1998.

[10] S. C. Wang, "Artificial Neural Network," *Interdisciplinary Computing in Java Programming,* 2003.

[11] Barlow, "Unsupervised Learning," *Neural Computation,* 1989.

[12] Z. Ghahramani, "Unsupervised Learning," *Advanced Lectures on Machine Learning,* 2003.

[13] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks,* 2005.

[14] I. Syarif, A. P. Bennett and G. Wills, "Unsupervised Clustering Approach for Network Anomaly Detection," in *Networked Digital Technologies*, 2012.

[15] T. Szandała, "Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks," in *Bio-inspired Neurocomputing*, Springer, 2020.

[16] D. Svozil, V. Kvasnicka and J. Pospichal, "Introduction to multi-layer feed-forward neural networks," *Chemometrics and Intelligent Laboratory Systems,* 1997.

[17] Q. Wang, Y. Ma, K. Zhao and Y. Tian, "A Comprehensive Survey of Loss Functions in Machine Learning," *Annals of Data Science,* 2022.

[18] R. H. Nielsen, "Theory of the Backpropagation Neural Network," *Neural Networks for Perception,* 2014.

[19] S.-i. Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing,* 1993.

[20] N. Bjorck, C. Gomes, B. Selman and K. Weinberger, "Understanding Batch Normalization," *Advances in Neural Information Processing Systems,* 2018.

[21] S. Santurkar, D. Tsipras, A. Ilyas and A. Madry, "How Does Batch Normalization Help Optimization?," *Advances in Neural Information Processing Systems,* 2018.

[22] "Difference between Model Parameter and Hyperparameter," [Online]. Available: https://www.javatpoint.com/model-parameter-vs-hyperparameter.

[23] F. Hutter, L. Kotthoff and J. Vanschoren, "Hyperparameter Optimization," in *Automated Machine Learning*, Springer, pp. 3-34.

[24] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing,* 2020.

[25] J. Brownlee , "What is the Difference Between a Batch and an Epoch in a Neural Network?," *Machine learning mastery,* 2018.

[26] Y. Li, C. Wei and T. Ma, "Towards Explaining the Regularization Effect of Initial Large Learning Rate in Training Neural Networks," in *Advances in Neural Information Processing Systems 32*, 2019.

[27] S. Du, J. Lee, H. Li, L. Wang and X. Zhai, "Gradient Descent Finds Global Minima of Deep Neural Networks," in *36th International Conference on Machine Learning*, 2019.

[28] K. You, M. Long, J. Wang and M. I. Jordan, "How Does Learning Rate Decay Help Modern Neural Networks?," 2019.

[29] J. Brownlee, "What is the Difference Between a Batch and an Epoch in a Neural Network," Machine Learning Mastery, 2018.

[30] N. Murata, S. Yoshizawa and S. Amari, "Network information criterion-determining the number of hidden units for an artificial neural network model," *IEEE Transactions on Neural Networks,* 1994.

[31] D. Vidaurre, C. Bielza and P. Larrañaga, "L1 regularization coefficient," *International Statistical Review,* 2013.

[32] C. Cortes, M. Mohri and A. Rostamizadeh, "L2 Regularization for Learning Kernels," in *Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 2012.

[33] P. Baldi and P. J. Sadowski, "Understanding dropout," in *Advances in Neural Information Processing Systems 26*, 2013.

[34] M. Li, Z. Tong, C. Yuqiang and S. J. Alexander, "Efficient mini-batch training for stochastic optimization," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014.

[35] F. J. Pontes, G. F. Amorim, P. Balestrassi, A. P. de Paiva and J. R. Ferreira, "Design of experiments and focused grid search for neural network parameter optimization," *Neurocomputing,* 2016.

[36] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *Journal of Machine Learning Research,* 2012.

[37] J. Wu, X. Y. Chen, H. Zhang, L. D. Xiong, H. Lei and S. H. Deng, "Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization," *Journal of Electronic Science and Technology,* 2019.

[38] J. Snoek, H. Larochelle and R. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms," *Advances in Neural Information Processing Systems,* 2012.

[39] M. Narkhede, P. Bartakke and M. Sutaone, "A review on weight initialization strategies for neural networks," *Artificial Intelligence Review,* 2022.

[40] Thimm and Fiesler, "Neural network initialization," *From Natural to Artificial Neural Computation,* 2005.

[41] S. K. Kumar, "On weight initialization in deep neural networks," *arXiv preprint,* 2017.

[42] R. Clark, Control system dynamics, Cambridge University Press, 1996.

[43] G. Goodwin, S. Graebe and M. Salgado, Control system design, 2000.

[44] N. Norman, Control systems engineering, John Wiley & Sons, 2020.

[45] C. Knospe, "PID control," *IEEE Control Systems Magazine,* 2006.

[46] M. Johnson and M. Mohammad, PID control, UK: Springer-Verlag London Limited, 2005.

[47] J. Qin and T. Badgwell, "An overview of industrial model predictive control technology," in *AIche symposium series*, American Institute of Chemical Engineers, 1997.

[48] Holkar, "An overview of model predictive control," *International Journal of control and automation,* 2010.

[49] B. Kouvaritakis and M. Cannon, "Model predictive control," in *Advanced Textbooks in Control and Signal Processing*, Springer, 2016.

[50] Hou and Xu, "On data-driven control theory: the state of the art and perspective," in *Acta Automatica Sinica*, 2009.

[51] Z. S. Hou and Z. Wang, "From model-based control to data-driven control: Survey, classification and perspective," *Information Sciences,* 2013.

[52] Kaelbling, Littman and Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research,* 1996.

[53] M. Wiering and M. Otterlo, "Reinforcement Learning," in *Adaptation, Learning, and Optimization*, Springer, 2012.

[54] J. Eschmann, "Reward Function Design in Reinforcement Learning," in *Reinforcement Learning Algorithms: Analysis and Applications*, Springer, 2021.

[55] M. Coggan, "Exploration and Exploitation in Reinforcement Learning," in *CRA-W DMP Project at McGill University*, 2004.

[56] O. Nachum, M. Norouzi, K. Xu and D. Schuurmans, "Bridging the Gap Between Value and Policy Based Reinforcement Learning," in *Advances in Neural Information Processing Systems*, 2017.

[57] I. Grondman, L. Busoniu, G. Lopes and R. Babuska, "A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C,* 2012.

[58] M. Igl, K. Ciosek, Y. Li, S. Tschiatschek, C. Zhang, S. Devlin and K. Hofmann, "Generalization in Reinforcement Learning with Selective Noise Injection and Information Bottleneck," in *Advances in Neural Information Processing Systems 32*, 2019.

[59] O. Barndorff-Nielsen and N. Shephard, "Non-Gaussian Ornstein–Uhlenbeck-based models and some of their uses in financial economics," *Journal of the Royal Statistical Society: Series B (Statistical Methodology),* 2002.

[60] R. Marotta, S. Strano, M. Terzo and C. Tordela, "Multi-Output Physically Analyzed Neural Network for the Prediction of Tire–Road Interaction Forces," *SAE International Journal of Vehicle Dynamics, Stability, and NVH,* 2024.

[61] R. Marotta, V. Ivanov, S. Strano, M. Terzo and C. Tordela, "Estimation of the Tire-Road Interaction Forces by using Pacejka's Formulas with Combined Slips and Camber Angles," in *WCX SAE World Congress Experience*, 2023.

[62] R. Marotta, S. Strano, M. Terzo and C. Tordela, "Improvement of Traction Force Estimation in Cornering through Neural Network," *SAE International Journal of Connected and Automated Vehicles,* 2024.

[63] R. Marotta, S. Strano, M. Terzo and C. Tordela, "On the Prediction of the Sideslip Angle Using Dynamic Neural Networks," *IEEE Open Journal of Intelligent Transportation Systems,* 2024.

[64] R. Marotta and L. De Matteis, "Neural Network-Based Virtual Measurement of Road Vehicle Wheel Displacements," in *The International Conference of IFToMM ITALY*, 2024.

[65] R. Marotta, S. van Aalst, K. Praet, M. Dhaens, V. Ivanov, S. Strano, M. Terzo and C. Tordela, "Enhancing Wheel Vertical Displacement Estimation in Road Vehicles Through Integration of Model-Based Estimator with Artificial Intelligence," *IEEE Open Journal of Vehicular Technology,* 2024.

[66] L. De Matteis and R. Marotta, "Addressing Integration Challenges in Vehicle Roll and Pitch Estimation Using Neural Networks," in *International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)*, 2024.

[67] "ISO 8855:2011," International Organization for Standardization, [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:8855:ed-2:v1:en.

[68] M. Guiggiani, The Science of Vehicle Dynamics: Handling, Braking, and Ride of Road and Race Cars, 2014.

[69] "IG-500A," SBG, [Online]. Available: https://zenmicrosystems.co.in/wp-content/uploads/2017/09/IG-500A-Leaflet.pdf.

[70] "Wheel force transducer RoaDyn S635nsp / 9267A2 RoaDyn S635 nsp," Kistler, [Online]. Available: https://www.kistler.com/IT/it/p/wheel-force-transducer-roadyn-s635nsp-roadyn-s635-nsp/000000000018027002.

[71] D. Stathakis, "How many hidden layers and nodes?," *International Journal of Remote Sensing,* 2009.

[72] T. Fine, Feedforward Neural Network Methodology, Springer, 2006.

[73] J. Sola and J. Sevilla, "Importance of input data normalization for the application of neural networks to complex industrial problems," in *IEEE Transactions on Nuclear Science*, 1997.

[74] J. Snoek, H. Larochelle and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms," in *Advances in Neural Information Processing Systems 25*, 2012.

[75] M. V. Shcherbakov, A. Brebels, N. L. Shcherbakova, A. P. Tyukov, T. A. Janovsky and V. A. Kamaev, "A Survey of Forecast Error Measures," *World Applied Sciences Journal 24 (Information Technologies in Modern Industry, Education & Society),* 2013.

[76] H. B. Pacejka, Typle and Vehicle Dynamics, Elsevier, 2006.

[77] M. Guiggiani, The Science of Vehicle Dynamics, Spinger, 2014.

[78] S. Ruder, "An overview of gradient descent optimization algorithms," 2016. [Online]. Available: http://arxiv.org/abs/1609.04747.

[79] D. T. Greenwood, Principles of Dynamics, 1988.

[80] M. Short, M. Pont and Q. Huang, Safety and Reliability of Distributed Embedded Systems, 2004.

[81] I. Automotive, User's Guide Version 10.0, CarMaker.

[82] I. Automotive, Reference Manual Version 10.0, CarMaker.

[83] Y. Fukada, "Slip-Angle Estimation for Vehicle Stability Control," *Vehicle System Dynamics,* 2010.

[84] M. Guiggiani, The Science of Vehicle Dynamics, Springer, 2014.

[85] "Why Random Shuffling improves Generalizability of Neural Nets," 2021. [Online]. Available: https://www.deepwizai.com/simply-deep/why-random-shuffling-improves-generalizability-of-neural-nets.

[86] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *Journal of Machine Learning Research,* 2012.

[87] Y. Li and S. Abdallah, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, 2020.

[88] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference for Learning Representations*, 2015.

[89] H. Ren, T. Shim, J. Ryu and S. Chen, "Development of effective bicycle model for wide ranges of vehicle operations," in *SAE World Congress & Exhibition*, 2014.

[90] "Lommel Proving Ground (LPG)," [Online]. Available: https://www.fordlpg.ford.com/pages/dtc/Testbanen.

[91] "Speed limits by country," [Online]. Available: https://en.wikipedia.org/wiki/Speed_limits_by_country.

[92] A. S. Kilinç and T. Baybura, Determination of Minimum Horizontal Curve Radius Used in the Design of Transportation Structures, Depending on the Limit Value of Comfort Criterion Lateral Jerk, Engineering Surveying, Machine Control and Guidance.

[93] N. Stojanovic, O. I. Abdullah, I. Grujic, J. Glisovic and A. Belhocine, "The influence of spoiler on the aerodynamic performances and longitudinal stability of the passenger car under high speed condition," *Journal of Visualization.*

[94] I. A. Faisal, T. W. Purboyo and A. S. R. Ansori, "A Review of Accelerometer Sensor and Gyroscope Sensor in IMU Sensors on Motion Capture," *Journal of Engineering and Applied Sciences,* 2020.

[95] N. Ahmad, R. A. R. Ghazilla and N. M. Khairi, "Reviews on Various Inertial Measurement Unit (IMU) Sensor Applications," *International Journal of Signal Processing Systems,* 2013.

[96] T. W. Hestermeyer, T. Frese and A. Thoene, "Steering angle detection by means of ESC and EPAS". Patent US20080119987A1, 2007.

[97] A. Markel, "Brake & Front End," 2017. [Online]. Available: https://www.brakeandfrontend.com/steering-angle-frequently-asked-questions/#:~:text=How%20is%20the%20angle%20measured,the%20steering%20wheel%20in%20degrees..

[98] N. Roveri, G. Pepe and A. Carcaterra, "OPTYRE – A new technology for tire monitoring: Evidence of contact patch phenomena," *Mechanical Systems and Signal Processing,* 2016.

[99] "Levers and Pulleys in Translating Mechanical Systems," [Online]. Available: https://lpsa.swarthmore.edu/Systems/MechTranslating/TransMechSysLeversPulleys.html.

[100] F. Chang and Z. L. H, "Dynamic model of an air spring and integration into a vehicle dynamics model," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering,* 2008.

[101] Bebis and Georgiopoulos, "Feed-forward neural networks," *IEEE Potentials*, 1994.

[102] Y. Bai, "RELU-Function and Derived Function Review," in *International Conference on Science and Technology Ethics and Human Future*, 2022.

[103] F. Liu, F. Xue, W. Wang, W. Su and Y. Liu, "Real-time comprehensive driving ability evaluation algorithm for intelligent assisted driving," *Green Energy and Intelligent Transportation,* 2023.

[104] Y. Wang, J. Gong, B. Wang, P. Jia and T. Kyo, "Off-road testing scenario design and library generation for intelligent vehicles," *Green Energy and Intelligent Transportation,* 2022.

[105] Lillicrap, P. Timothy , J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver and D. Wierstra, "Continuous Control with Deep Reinforcement Learning," *ArXiv,* 2015.

[106] M. Kaur, S. Kakar and D. Mandal, "Electromagnetic interference," in *3rd International Conference on Electronics Computer Technology*, 2011.

[107] D. C. DeGroot, D. K. Walker and R. Marks, "Impedance Mismatch Effects on Propagation Constant Measurements," in *5th Topical Mtg. Electrical Performance of Electronic Packaging*, 1996.

[108] P. R. Saulson, "Thermal noise in mechanical experiments," *Physical Review D,* 1990.

[109] W. H. Press and S. A. Teukolsky, "Savitzky-Golay Smoothing Filters," *Computers in Physics.*

[110] J. F. Kennedy, Mathematics of Statistics.

[111] R. A. Roberts and C. T. Mullis, Digital signal processing.

[112] M. V. Shcherbakov, A. Brebels, N. L. Shcherbakova, A. P. Tyukov, T. A. Janovsky and V. A. Kamaev, "A Survey of Forecast Error Measures," *World Applied Sciences Journal 24,* 2013.