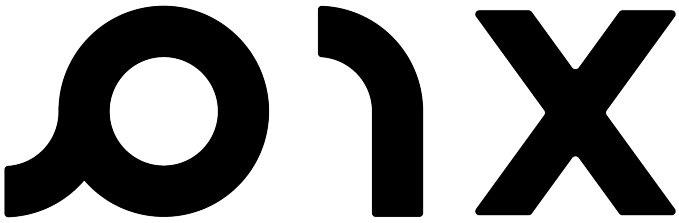
February 27, 2024

Autonomous Driving Department

Cristian Gariboldi

Comparison of Black Box and Grey Box Models for Longitudinal Vehicle's Dynamic



Pix Moving

Year 2023/24

Supervisor: Mark Jin

Index

List of Figures

Abstract

Modeling the dynamic of a vehicle is essential for two reasons:

1. When designing a control system that can accomplish autonomous task like steering or velocity control, we rely on Prediction Models, which are central for modern control applications, where the predicted output is controlled;
2. If instead we want to simulate the behavior of the system for new inputs, we rely on Simulation Models, which for example are useful to test what happens in new situations, to design systems and controllers and to mimic physical systems.

In literature, there exist several model's architectures.
In this paper I will compare Black Box models and Gray Box models of the longitudinal dynamic of a 4-wheels vehicle called Robobus, produced by Pix Moving.
First, I will explain the methodology of data collection and data pre-processing, followed with the analysis of the training stage and the parameters estimation methods.
Afterwards, I will analyze the structure of the black box and grey box models, dividing the analysis between linear and nonlinear systems.
Then, I will evaluate the models on the same validation dataset, which includes only longitudinal maneuvers, comparing their performances.
Finally, I will test the robustness of the models on a dataset which includes also steering maneuvers, and I will show how to significantly improve the black box model's performance by adding the steering angle as input to the model during the training stage.
It will be shown that the gray box models are more robust and accurate than the black box ones even when applying steering maneuvers, but the updated black box model trained on the steering angle as additional input outperforms the gray box one.

1 Introduction

Autonomous cars will play an important role in the next future mobility because of their safety and higher utilization of roads.
In order to guarantee these benefits, one of the main first steps is to design an accurate model of the vehicle, which should be able to precisely describe the dynamical behaviour of the car in the environment.
Regarding the longitudinal dynamic, the most popular approach is to design a first principle based model and then estimate the unknown parameters using optimizers such as the least square method. A limitation of this approach could be the modeling of the power-train system. Indeed, the knowledge of engine maps sometimes is available just for the manufacturer, and not for the end users.
To go beyond this limitation, it is possible to approximate the power-train system with a first-order linear system with a time delay, under certain assumptions which I will show later.
More recently, with the evolution of machine learning and deep learning architectures, other approaches have been developed which do not require any knowledge of the system. Even though they have been criticized for their low interpretability compared to the physic based models, learning based approaches have showed to have high performance and to be able to take into account the non-linearities of the system that may be difficut to model.
In Pix Moving, a Chinese company which builds level 4 autonomous vehicles called Robobus, I have had the opportunity to design a model of the longitudinal dynamic.
I designed both black box and grey box models, and compared them on the same validations datasets in order to highlight their differences of performance.

Figure 1.1: Robobus — Pix Moving

2 Data Collection and Preprocessing

Data collection and preprocessing are necessary when designing black box and gray box models. In black box models design, data are needed to build the model in the training stage, while in the gray box ones, data are essential to estimate the parameters of the model.
In the final stage, we also need a dataset to validate their prediction accuracy.
That's why it is important to acquire a high quality dataset, properly distributed and filtered.
In my case study, I collected the following data:

- Throttle and brake levels;
- Velocity, measured by the VCU (vehicle control unit);
- Acceleration, measured by IMU sensor (inertial measurement unit);
- Pitch angle, measured by IMU sensor;
- Steering angle.

2.1 Data Acquisition

During the data collection process, our goal is to obtain a dataset which should be uniformly distributed in its domain.
In order to achieve that, every data is classified according to its speed and throttling/braking information. This way, we can check if we have collected enough data for every case scenario.
This stage is very important because this way we can make sure that we have a balanced dataset and that we have enough data to train our models or estimate their parameters for every conditions.
The scenarios are divided as follow:

- LOW SPEED SCENARIO (0 - 10km/h):
    1. Brake: 0 - deadzone
    2. Brake: deadzone - 15%
    3. Brake: 15% - 25%
    4. Brake: 25% - 100%
    5. Throttle: 0 - deadzone
    6. Throttle: deadzone - 30%
    7. Throttle: 30% - 55%
    8. Throttle: 55% - 100%

- HIGH SPEED SCENARIO (10km/h - 40km/h):
  1. Brake: 0 - deadzone
  2. Brake: deadzone - 15%
  3. Brake: 15% - 25%
  4. Brake: 25% - 100%
  5. Throttle: 0 - deadzone
  6. Throttle: deadzone - 30%
  7. Throttle: 30% - 55%
  8. Throttle: 55% - 100%

We have two main scenarios in which we collect data: low speed and high speed.
In each scenario, we have eight different conditions, depending on the brake or throttle cases and their levels. Notice that deadzone is the value needed for the vehicle to effectively start braking or accelerating, respectively. It can be found experimentally.
Once each case has reached the preset threshold value of the maximum number of data to collect, we can stop our collection process and start the preprocessing step.

2.2 Data Filtering

Preprocessing data is essential to ensure a good quality dataset, and consequently, accurate models.
Indeed, data may be affected by noise and could present several outliers, which will affect the quality of our models.
For this reason, the following filtering procedure has been applied:

1. Since our goal is to build models of the longitudinal dynamics of the vehicle, data with steering angle values grater than two degrees are filtered out;
2. Data with velocity equal to zero or greater than the upper threshold (40km/h) have been removed from the dataset;
3. The longitudinal acceleration measured by the IMU is not the real longitudinal acceleration. Even if the collection process has been conducted on a flat road, it's almost impossible to avoid some ground irregularities which are going to create some bumps in the vehicle. These bumps introduce a gravitational acceleration component which disturbs the longitudinal one.

   By measuring the pitch angle, we can remove this disturbance according to the following formula:

$$Acc_{real} = Acc_{meas} - g\sin(\alpha)$$

Where:

- Acc_{real} is the real longitudinal acceleration of the vehicle;
- Acc_{meas} is the longitudinal acceleration measured by the IMU sensor;
- g is the gravitational acceleration;
- α is the pitch angle, in radians.

4. In order to ensure data consistency, we need to check the values of two consecutive throttle/brake commands. If their difference is greater than a certain threshold, we filter out those data, according to the following formula:

$$|cmd(t) - cmd(t+1)| < threshold$$

Where:

- $cmd(t)$ and $cmd(t+1)$ are the values of two consecutive brake or throttle commands;
- $threshold$ is the maximum command perturbation allowed.

5. Afterwords, the following mean filter is applied in order to smooth data:

$$y = \frac{(x_{t-1} + x_{t-2} + ... + x_{t-N})}{N}$$

Where:

- N is the mean filter window size (equal to 20 in our case);
- $x_{t-1}, x_{t-2}, ..., x_{t-N}$ are data from time $t-1$ to $t-N$.

6. Finally, we remove the outliers according to the following formula:

$$\frac{x - x_{mean}}{x_{std}} > 1 : outlier$$

Where:

- x is the data to be processed;
- x_{mean} is the mean of the data;
- x_{std} is the standard deviation of the data.

3 Linear Models Structure

3.1 Gray Box Linear Model

3.1.1 Forces Acting on the Vehicle

First principle based models are very common and used among scientific community, especially when dealing with simple dynamics such as the longitudinal one of a vehicle. In order to design such a model, a knowledge of the system should be available.
We can obtain the longitudinal model of the vehicle relying on Newton's second law [1], which states that the inertial force applied to the car is equal to the sum of:

1. F_{roll}: rolling resistance force;
2. F_{drag}: aerodynamic drag force;
3. F_{brake}: braking force;
4. F_{gravity}: downgrade force;
5. F_{engine}: engine drive force transmitted to the wheels.

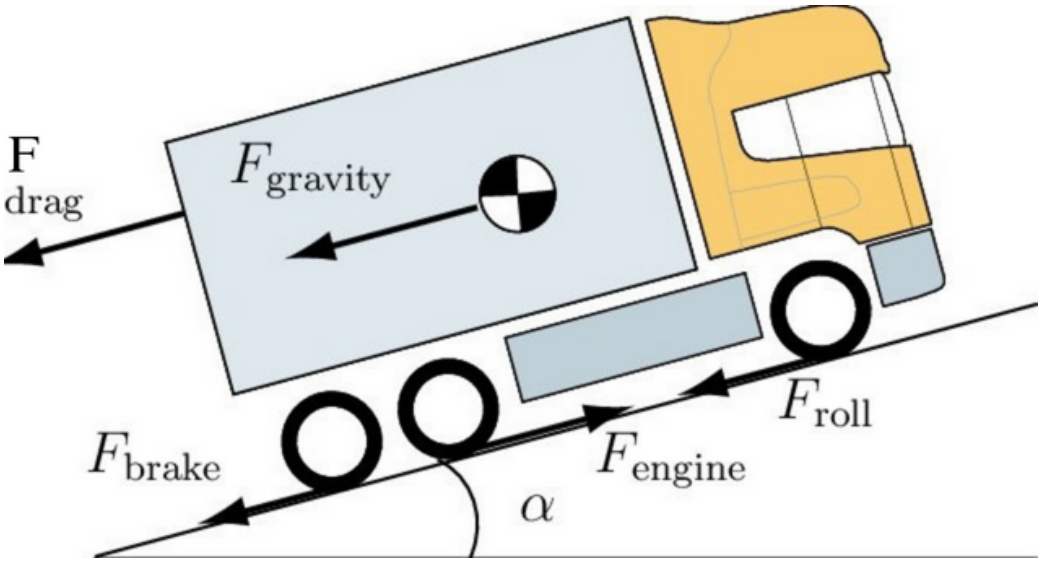


Figure 3.1: Longitudinal Forces Acting on the Vehicle [2]

Let's now define the forces acting on our system [3]:

- The rolling resistance force is generated by the friction between the tyre and the roadway contact surface during rolling. It can depend on several factors such as vehicle speed, tyre pressure and road surface characteristic, that's why it is difficult to estimate analytically. It can be defined as:

$$F_{\text{roll}} = (Cr_0 + Cr_1 v) Mg \cos \alpha$$

  Where:

  1. Cr_0 and Cr_1 are tyre rolling resistance coefficients;
  2. M is the mass of the vehicel;
  3. g is the gravitational acceleration constant;
  4. α is the slope of the road.

- The aerodynamic drag force is a resistance force that occurs when air flows over the vehicle. It can be defined as:

$$F_{\text{drag}} = \frac{1}{2} \rho A C v^2$$

  Where:

  1. ρ is the air density;
  2. A is the vehicle frontal area;
  3. C is the aerodynamic drag coefficient.

- The grading resistance force occurs when the road is not flat but has a slope. It is the gravitational force acting on the vehicle and can be defined as:

$$F_{\text{gravity}} = Mg \sin \alpha$$

We can notice that $\vec{F}_{\text{roll}}$, $\vec{F}_{\text{drag}}$ and $\vec{F}_{\text{brake}}$ are always opposite to the movement of the vehicle, but $\vec{F}_{\text{gravity}}$ and $\vec{F}_{\text{engine}}$, depending on the slope and the operation point of the motor respectively, can have a positive or negative sign.

In vector form, the equation that represents our system can be written as follow:

$$\frac{d}{dt}(M\vec{v}) = \vec{F}_{\text{roll}} + \vec{F}_{\text{drag}} + \vec{F}_{\text{brake}} + \vec{F}_{\text{gravity}} + \vec{F}_{\text{engine}}$$

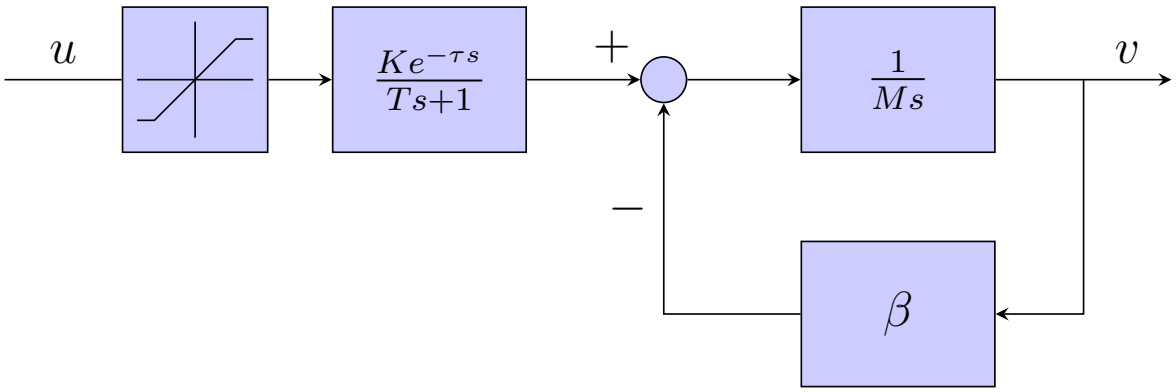
Where:

- M is the mass of the vehicle;
- v is its longitudinal velocity.

For the sake of simplicity, let's assume that the vehicle is operating on a flat road, at low speeds and without braking. Under these conditions, we can neglect $\vec{F}_{\text{gravity}}$, $\vec{F}_{\text{drag}}$ and $\vec{F}_{\text{brake}}$. $\vec{F}_{\text{engine}}$ represents now the main force that determines the movement of the vehicle, and representing this force mathematically can be quite complex.

3.1.2 System Model — Frequency Domain

The Robobus has 4 in-wheels electric DC motors and operates at speeds less than 40km/h. At such low speeds, the dynamics of the powertrain dominates the dynamics of the vehicle and finding an accurate model is crucial.
Since we have no information about the DC motors, it is reasonable to initially adopt a simplification which consists in representing the engine transmission system by a first-order linear system with constant T, static gain K and time delay τ. We can also assume to model the rolling resistance force just proportional to the longitudinal speed of the vehicle with constant β [4].
If we write our model in frequency domain, we can represent the longitudinal dynamics of the car with the following block diagram:



If we now compute the resulting transfer function of the block diagram, where the input is the throttle level u and the output is the velocity v, we obtain:

$$G(s) = \frac{V(s)}{U(s)} = \frac{\frac{K}{MT}e^{-\tau s}}{(s + \frac{1}{T})(s + \frac{\beta}{M})}$$

Approximating the engine transmission as a first order linear system could be not so accurate since it may not represent the motor's behavior in all operating conditions, but since since we limit the model's input (which is the throttle signal sent to the ECU) between a minimum value (idle speed) and a maximum value (chosen for security reasons), we can assume that the motor will operate in the linear operating condition (constant torque region, within its base speed).
We can make this assumption also because the Robobus, due to regulations, can operate just at low speeds (not more than 40km/h), and for the same reason, as already stated, we can assume the aerodynamic drag force to be negligible.

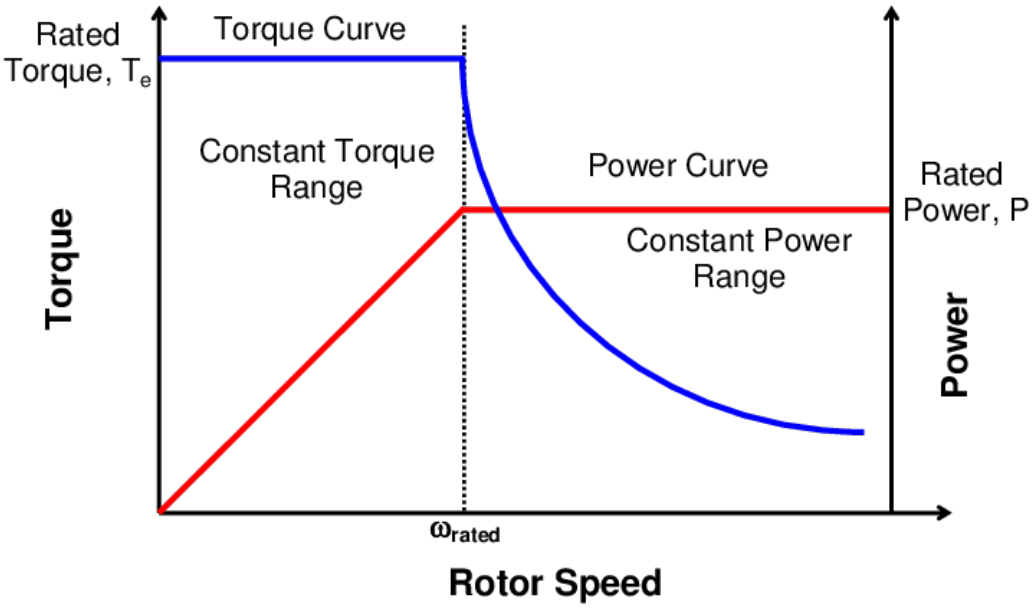


Figure 3.2: DC Motor Torque Curve [5]

As you can see from the graph of the torque curve of a DC motor, if we keep the rotor speed smaller than the rated speed, we can expect a linear behaviour.
That's why approximating the powertrain system as a first-order linear system with a delay can be quite accurate for such low speeds.

3.1.3 Discretization

Now we want to obtain a discrete version of $G(s)$ in order to use this model for computational applications. Before doing that, we can approximate the time delay using the Taylor series approximation:

$$e^{-\tau s} = 1 - \tau s = \tau(\frac{1}{\tau} - s)$$

So, our transfer function becomes:

$$G(s) = \frac{V(s)}{U(s)} = \frac{\frac{K\tau}{MT}(\frac{1}{\tau} - s)}{(s + \frac{1}{T})(s + \frac{\beta}{M})}$$

We are now ready to apply the bilinear transorm in order to obtain a discrete-time model. I chose to discretize the system with Tustin method:

$$s = \frac{2}{T_a}\frac{z-1}{z+1}$$

where T_a is the sampling period.

After some computations, we obtain the following discrete-time system:

$$G(z) = \frac{H_1 + H_2 z^{-1} + H_3 z^{-2}}{H_4 + H_5 z^{-1} + H_6 z^{-2}}$$

Where:

$$\begin{cases} H_1 = T_a(\frac{K}{MT}T_a - 2) \\ H_2 = \frac{2K}{MT}T_a^2 \\ H_3 = T_a(\frac{K}{MT}T_a + 2) \\ H_4 = \frac{\beta}{MT}T_a^2 + 2(\frac{\beta}{M} + \frac{1}{T})T_a + 4 \\ H_5 = \frac{2}{T}\frac{\beta}{M}T_a^2 - 8 \\ H_6 = \frac{\beta}{MT}T_a^2 - 2(\frac{\beta}{M} + \frac{1}{T})T_a + 4 \end{cases}$$

3.1.4 Parameters Estimation

This model has some unknown parameters ($H_1, H_2, H_3, H_4, H_5, H_6$) that we could estimate using data-based parameter identification techniques.
I chose to represent the model as an Auto Regressive eXogeneous inputs model, as known as ARX model.
The ARX model corresponding to our $G(z)$ can be written as:

$$y(k) = -\frac{H_5}{H_4}y(k-1) - \frac{H_6}{H_4}y(k-2) + \frac{H_1}{H_4}u(k) + \frac{H_2}{H_4}u(k-1) + \frac{H_3}{H_4}u(k-2) + \frac{1}{H_4}e(k)$$

We can do the estimation of the parameters by using a least squares method over pairs of input and output data, resulting in a minimization of the error of the model for one-step-ahead prediction.
So, we can rewrite the model as:

$$y(k) = \psi^T(k-1)\hat{\theta} + \xi(k)$$

Where:

- $\psi^T(k-1)$ is a vector that contains input and output data collected until (k–1);
- $\hat{\theta}$ is a vector containing the estimated parameters;
- $\xi(k)$ represents the residuals between estimated and measured output.

We can rewrite it in matrix form, since we will apply it to a lot of data:

$$Y = \Psi\hat{\Theta} + \Xi$$

Where:

- Y is the measured output vector;
- Ψ is the regressor matrix;
- Ξ is the residual vector.

Finally, the parameter vector $\hat{\Theta}$ can be obtained with the minimum norm solution:

$$\hat{\Theta} = [\Psi^T\Psi]^{-1}\Psi^T Y$$

3.2 Black Box Linear Model

3.2.1 Structure

The second model I designed is a simple black box linear model.
In order to estimate the linear relationship between data, I used a multivariable linear regression with the following structure:

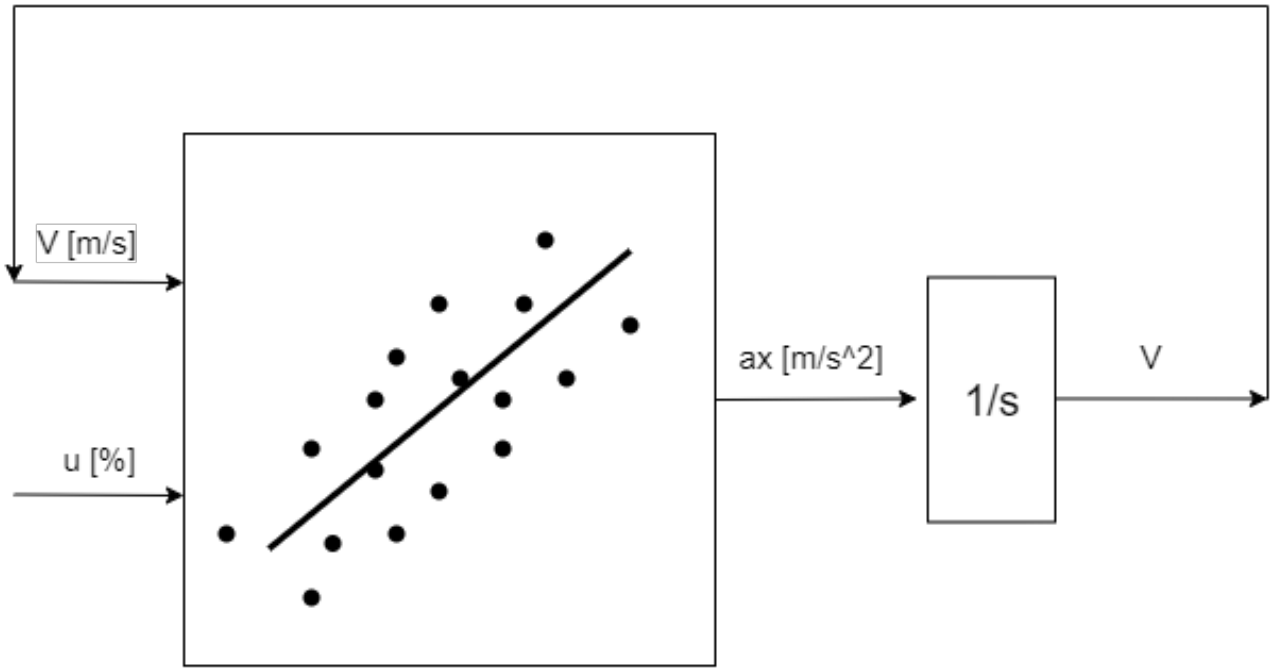


Figure 3.3: Linear Regression Structure

This model has been designed to have the following inputs and output:

- Inputs:
  1. Velocity $[m/s]$
  2. Throttle level [%]
- Output:
  1. Acceleration $[m/s^2]$

Moreover, since in the validation phase we are going to plot the velocity error in order to compare its prediction accuracy with the gray box model, from our acceleration output we obtain the speed as shown in the diagram above (Figure 3.3).

3.2.2 Formulation

Mathematically speaking, we want to minimize the vertical distance between data points and our predictor $\hat{y}$.
If we assume that the relationship between dependent and independent variables is linear, we can define:

$$y = X\beta + \epsilon$$

Where:

- y is the output variable (the one we try to predict with $\hat{y}$);
- X represents the input information provided to the model (independent variables or regressors);
- β represents the regression coefficients;
- ϵ is the error term (or disturbance term).

In our case, being the regressor composed by two independent variables, the multiple linear regression model can be defined as:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon$$

being β_0 the intercept term.

The principle behind the simple linear regression, that we can extend to the multiple one, is to minimize the following error:

$$E_{rr} = \frac{1}{N}\sum_{i=1}^{n}(y_i - (\beta x_i + \epsilon))^2$$

Where:

- N is the total number of data points;
- y_i is the actual value of an observation;
- $\beta x_i + \epsilon$ is our prediction ($\hat{y}$).

3.2.3 Visualization

Let's now visualize our predictor in a three-dimensional space.
The independent variable x_1 is the velocity, and the independent variable x_2 is the throttle level.
The dependent variable y is the acceleration.

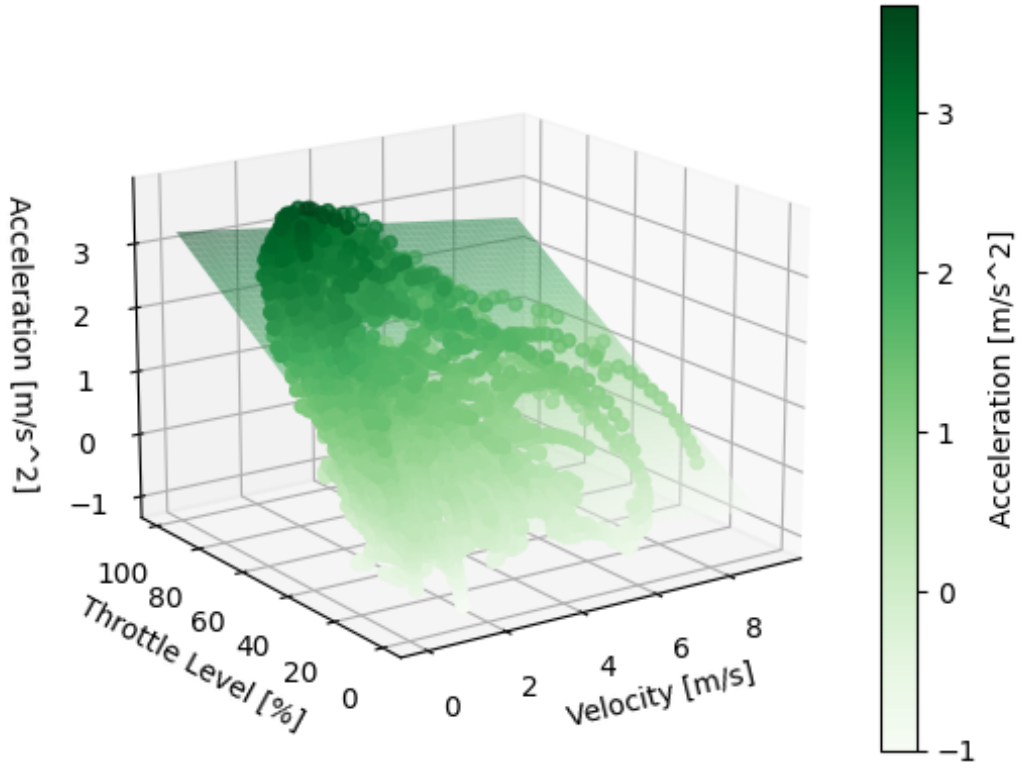


Figure 3.4: Linear Regression Visualization

The plot shows the predictor (green plane) together with the data points (green dots), in order to have a better understanding about the model's predictions capabilities.
However, we will analyze its performance in the "Results" chapter.

4 Non-Linear Models Structure

4.1 Gray Box Non-Linear Model

4.1.1 Formulation

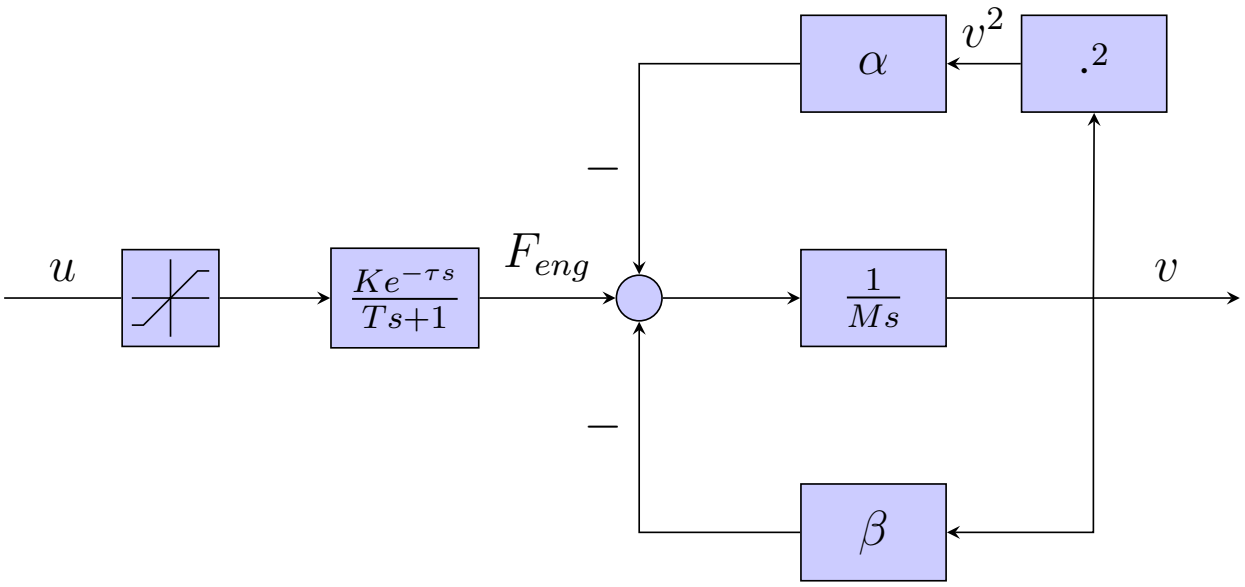
Even though the linear models designed in the previous chapter could be accurate, they don't take into account the non-linearities of the system.
Indeed, in order to obtain a linear representation of our system, we took into considerations some assumptions, which in reality could lead to a simplistic model.
In this section, we will tear down the assumption of the previous chapter and we will build a nonlinear model.

The assumption I would like to take into consideration is the drag force F_{drag}.
Indeed, even if the maximum speed of our vehicle is about 40km/h and the aerodynamic resistance force is very low, it is not absent. So, if we consider it in our system, we will obtain a more accurate model.

If we assume the road to be flat, our continuous-time model then becomes:

$$M\dot{v}(t) = -\beta v(t) - \alpha v^2(t) + F_{engine}(t)$$

This way, we end up with a nonlinear function. $F_{engine}(t)$ is the output force of the engine transmission system, which, as before, is represented like a first-order linear system with time delay.
Our block diagram then becomes:



4.1.2 Discretization

Let's now just focus on the car body dynamics, since the engine has been modeled exactly like in the gray box linear model.
Let's define:

1. $\beta' = \frac{\beta}{M}$
2. $\alpha' = \frac{\alpha}{M}$
3. $k(t) = \frac{F_{engine}(t)}{M}$
4. $y(t) = v(t)$

Our system becomes:

$$\dot{y}(t) = -\beta' y(t) - \alpha' y^2(t) + k(t)$$

Now we need to find the Discrete time system. We can use Eulero-forward method, which states that:

$$\dot{y}(t) = \frac{y(t+1) - y(t)}{\Delta t}$$

where Δt is the sampling time.

Our Discrete time system then becomes:

$$y(t) = (1 - \Delta t\beta')y(t-1) - \Delta t\alpha' y^2(t-1) + \Delta t k(t-1)$$

In our regressor we have a nonlinear component caused by $y^2(t-1)$, that's why I chose to represent the model as a Non-Linear Auto Regressive eXogeneous inputs model, as known as NARX model.
Then, we can proceed with the estimation of the parameters using a least squares method over pairs of input and output data, resulting in a minimization of the error of the model for one-step-ahead prediction, exactly as we did for the gray box linear case.

4.2 Black Box Non-Linear Models

4.2.1 Two Inputs Neural Network

The first Black Box model I have developed is a standard three-layers feedforward neural network with:

- Inputs:
  1. Velocity;
  2. Throttle level.
- Output:
  1. Acceleration.

The model was designed using ReLU activation functions, mean square error as the cost function and Adam optimizer.
As you can see in Figure 4.1, the model is able to capture the nonlinearities of the system.

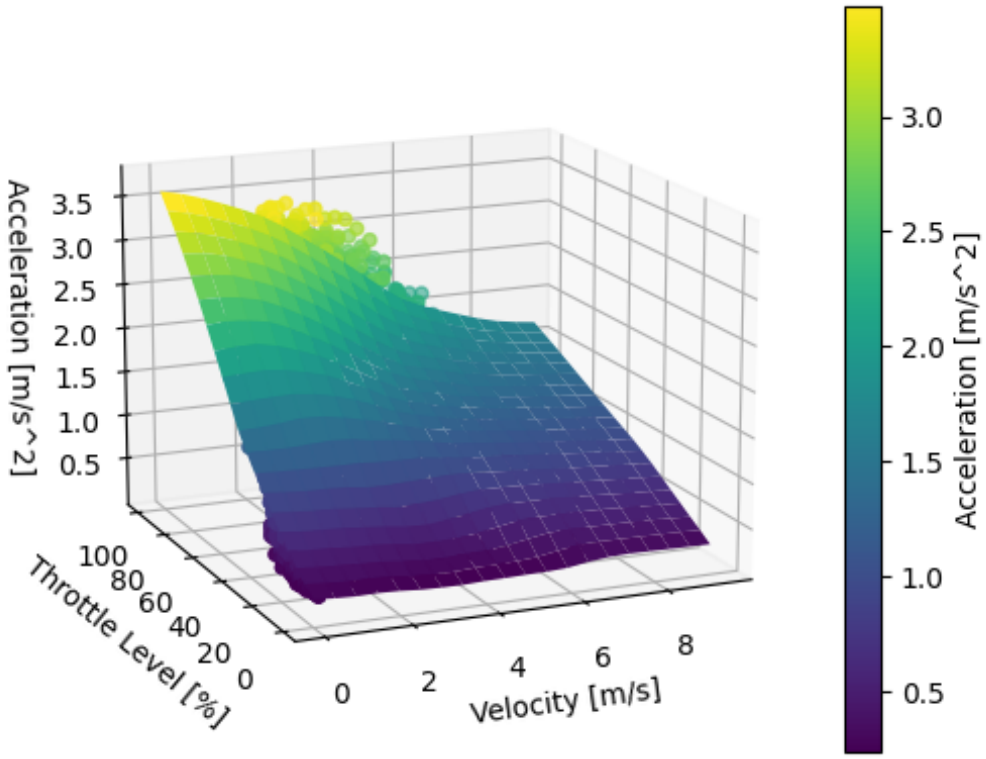


Figure 4.1: 2-Inputs Neural Network Model Visualization

As already stated in the Linear Regression chapter, we are going to evaluate the model based on the velocity prediction error. Since the output of our model is an acceleration, the velocity has been obtained as follow:

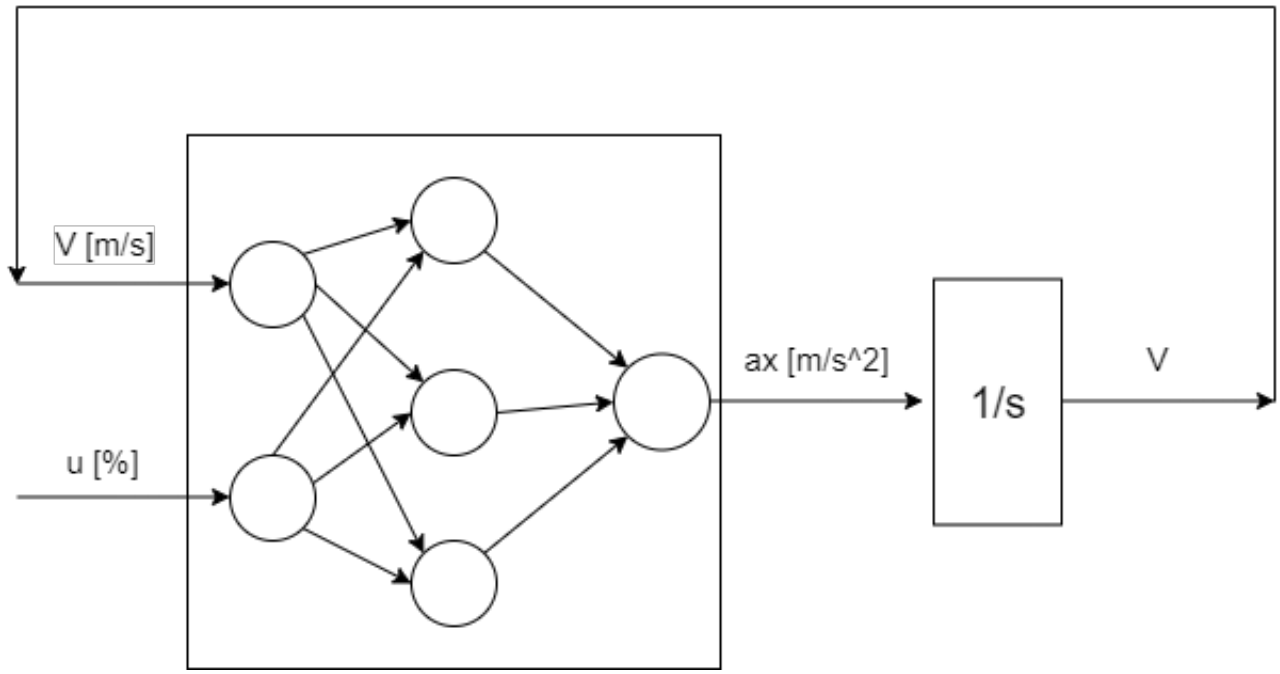


Figure 4.2: 2-Inputs Neural Network Structure

The metrics of this model will be carefully analyzed in the Results chapter, evaluating its prediction capabilities on datasets with and without steering perturbations.

4.2.2 Three Inputs Neural Network

The previous model, proposed by Baidu Apollo [6], [7], was trained to map the inputs (velocity and throttle level) to the output (acceleration).
I want to propose a solution that can improve the model by augmenting its architecture. Indeed, if we add one more input to the network, it will be able to learn the relationship between the enlarged regressor and the output, leading to a more sophisticated model.
Let's consider the following structure adding one more input:

- Inputs:
  1. Velocity;
  2. Throttle level;
  3. Steering angle.
- Output:
  1. Acceleration.

For visualization purposes, let's set the velocity to a constant value and let's plot the map between steering angle, throttle level and acceleration:

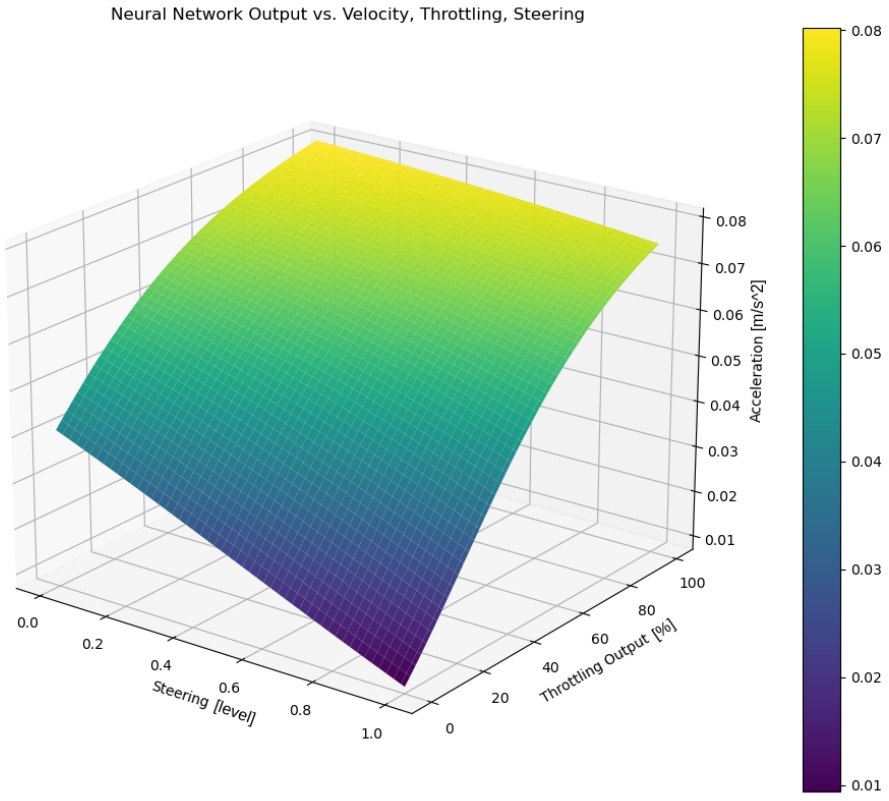


Figure 4.3: Steering — Throttle — Acceleration Map

In Figure 4.3, the map between steering angle, throttle level and acceleration was obtained by training a neural network, with the three inputs and one output already defined, on real data. It is clear that if we apply a constant throttle level, the acceleration decreases with the increase of the steering angle.
This is caused by the friction force between the wheels and the ground, which is higher when we apply large steering angles.

So, if we don't consider the steering angle as input to our network, the model we obtain would not take into consideration this friction force and will have poor performances in predicting the output if we apply steering commands to our vehicle.

Therefore, the final structure of our model is the following:

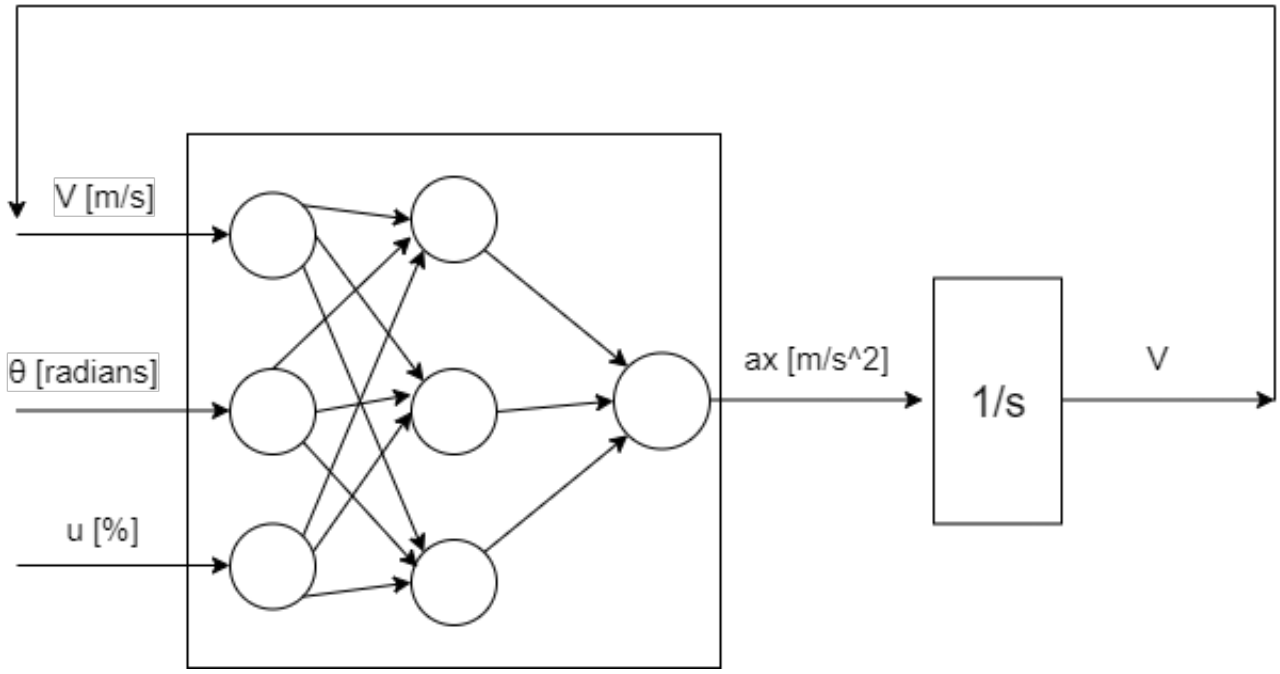


Figure 4.4: 3-Inputs Neural Network Structure

As stated before, for evaluation purposes, the velocity is obtained from the acceleration as shown in Figure 4.4.

In the next page, I will show two examples of new maps I have obtained from this enlarged structure.

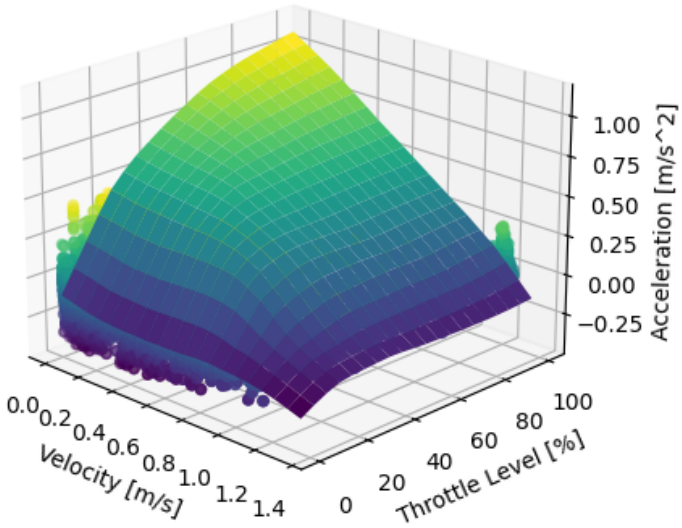


Figure 4.5: Neural Network Map — Steering level: 5% - 20%

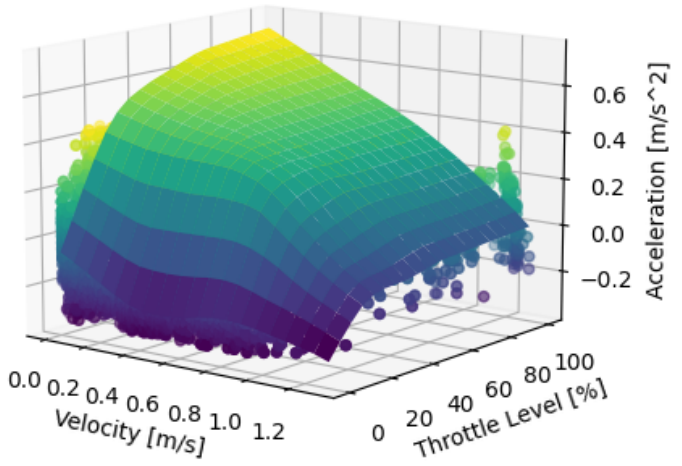


Figure 4.6: Neural Network Map — Steering level: 40% - 60%

5 Results

5.1 Linear Models

5.1.1 Longitudinal Dynamics without Steering Maneuvers

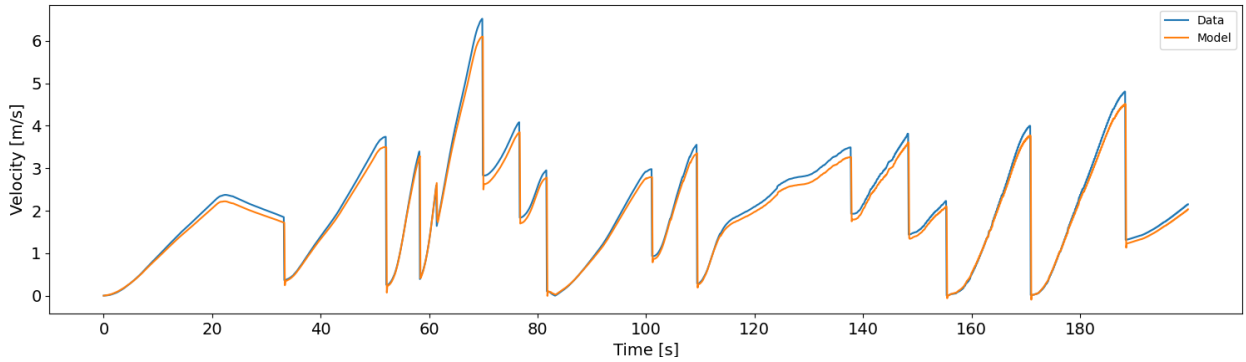


Figure 5.1: Gray Box — Linear Model

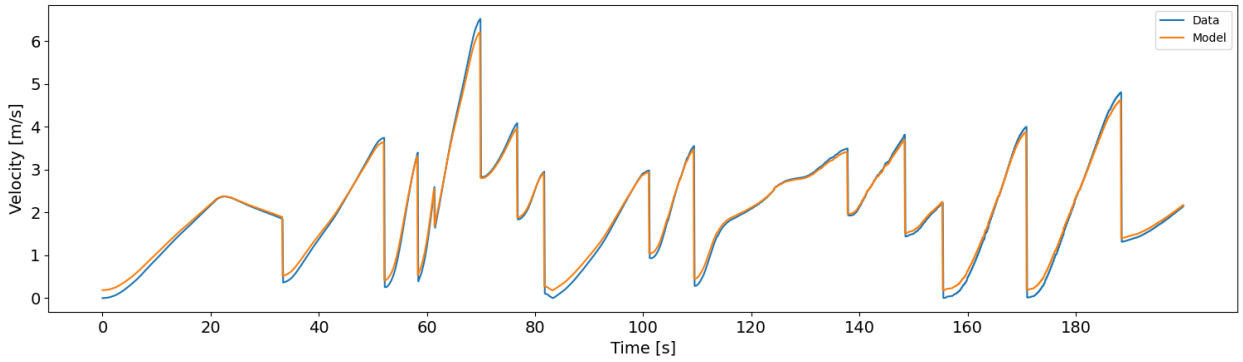


Figure 5.2: Black Box — Linear Model

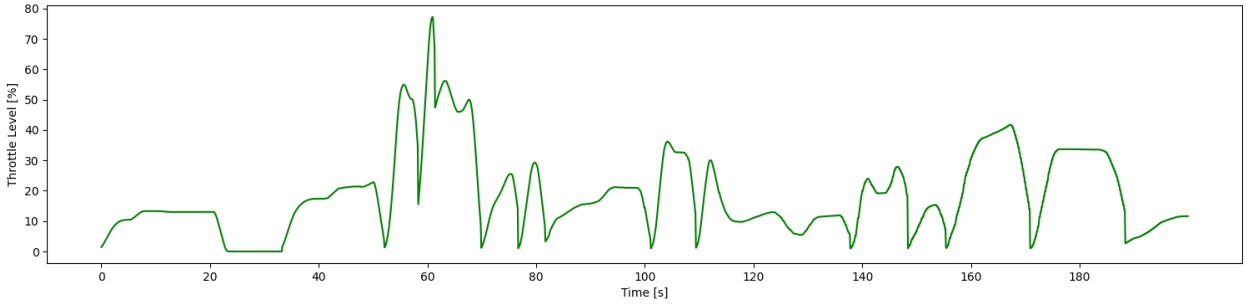


Figure 5.3: Throttle Level

5.1.2 Longitudinal Dynamics with Steering Maneuvers

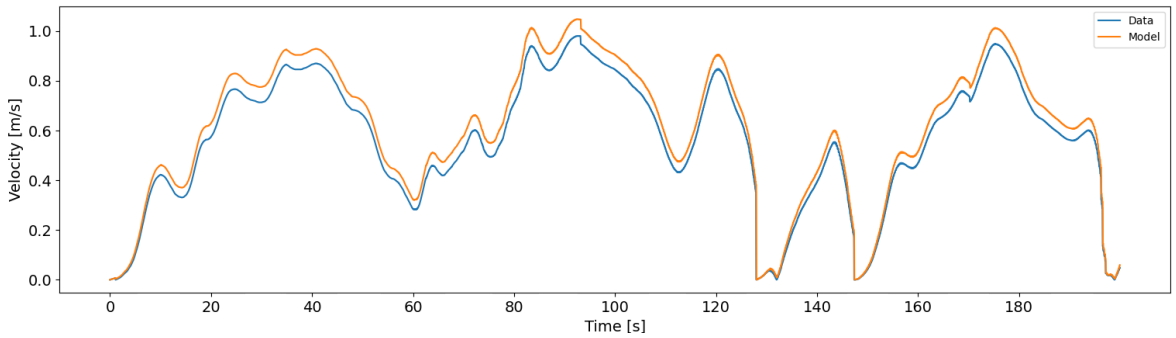


Figure 5.4: Gray Box — Linear Model — Steering Maneuvers

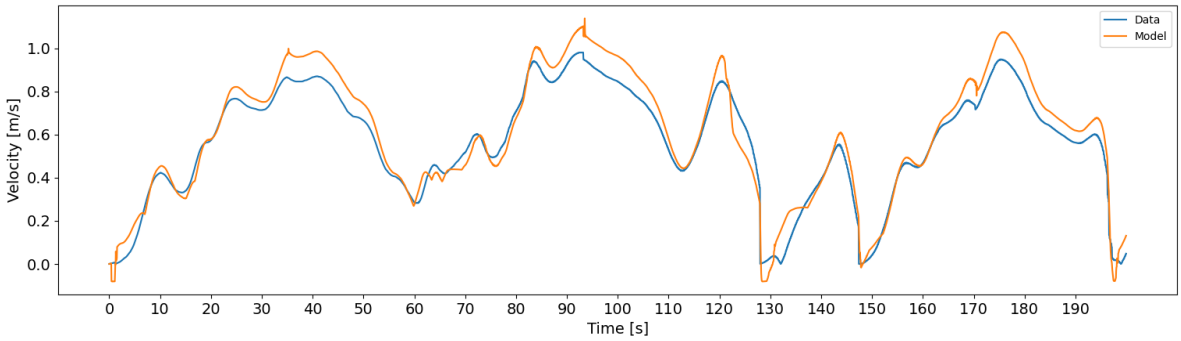


Figure 5.5: Black Box — Linear Model — Steering Maneuvers

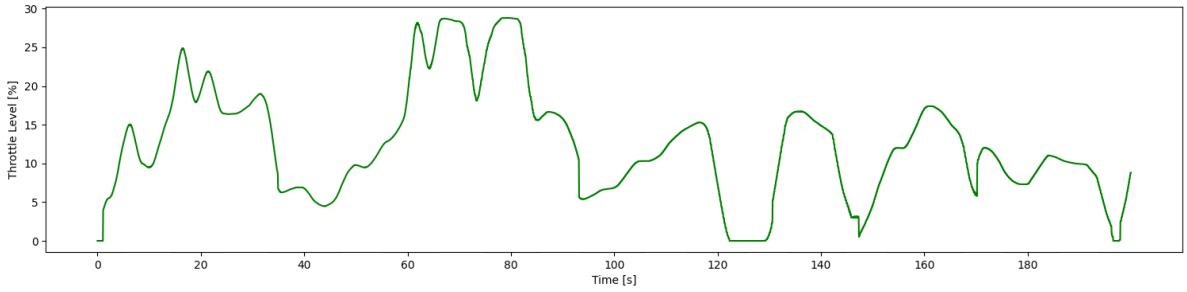


Figure 5.6: Throttle Level — Steering Maneuvers

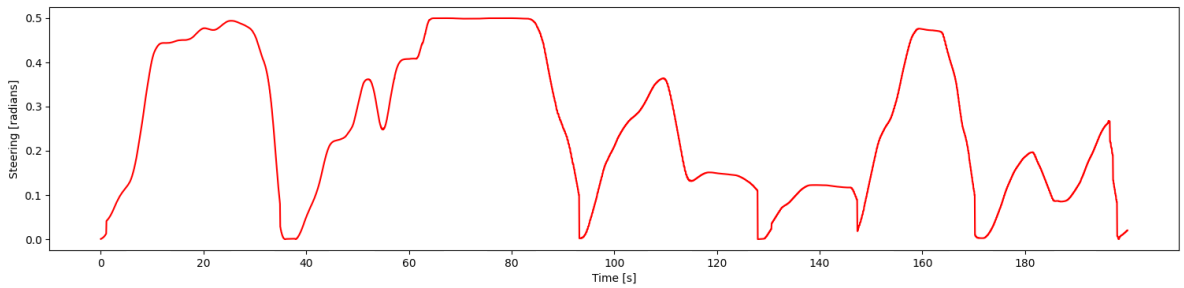


Figure 5.7: Steering Inputs

5.1.3 Metrics

Let's first inspect the metrics of our linear models on the dataset without steering inputs:

Linear Models Metrics — No Steering Maneuvers								
Linear Models	MSE	MAE	MSLE	MdAE	nRMSE	R2	RMSE	RRSE
Grey Box Model	0.06291	0.12506	**0.00755**	0.10913	0.03847	0.95681	0.25083	0.20782
Black Box Model	**0.05929**	**0.09831**	0.00916	**0.06751**	**0.03735**	**0.95935**	**0.24351**	**0.20162**

Overall, we can notice that the Linear Regression model performs slightly better than the Linear Grey Box model in predicting the velocity output for the longitudinal dynamic without applying steering inputs to our vehicle.

Let's now check the metrics of the same models, but evaluating them on the dataset with steering perturbations:

Linear Models Metrics — Steering Maneuvers								
Linear Models	MSE	MAE	MSLE	MdAE	nRMSE	R2	RMSE	RRSE
Grey Box Model	**0.10448**	**0.13207**	**0.04271**	**0.22141**	**0.05296**	**0.92427**	**0.32323**	**0.32539**
Black Box Model	0.18617	0.14682	0.09514	0.42529	0.11025	0.81614	0.43147	0.42879

As expected, both models shows a degradation compared with the previous metrics.
The Linear Regression presents a huge decline, indeed it is outperformed by the Linear Grey Box model, which, even though is slightly deteriorated, demonstrates to be robust to steering perturbations.

Let's now check the Non-Linear models performances and make a final comparison.

5.2 Non-Linear Models

5.2.1 Longitudinal Dynamics without Steering Maneuvers

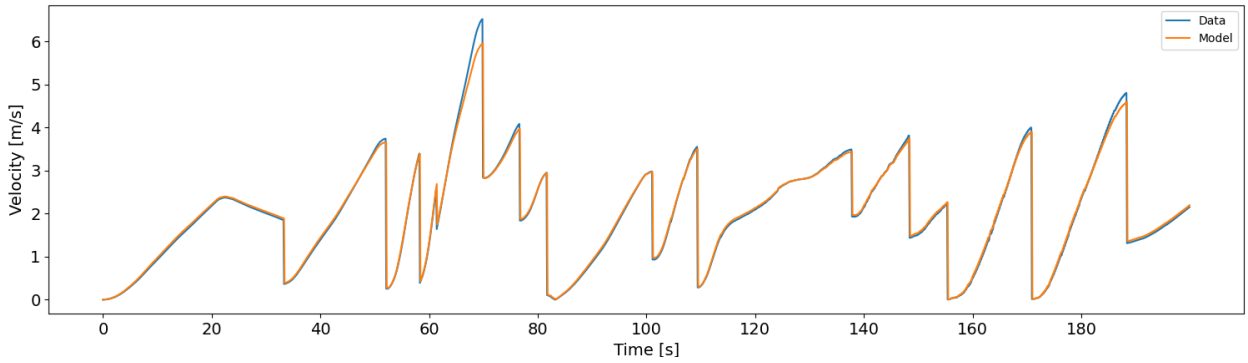


Figure 5.8: Gray Box — Nonlinear Model

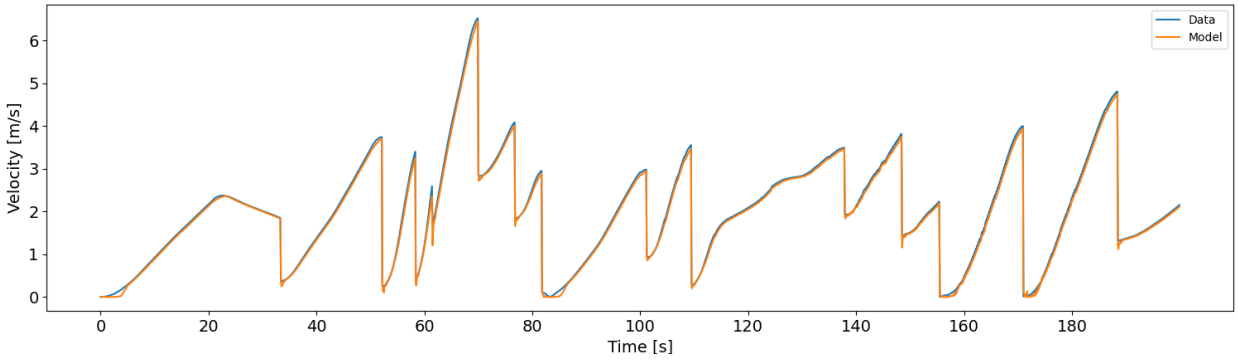


Figure 5.9: Black Box — Nonlinear Model

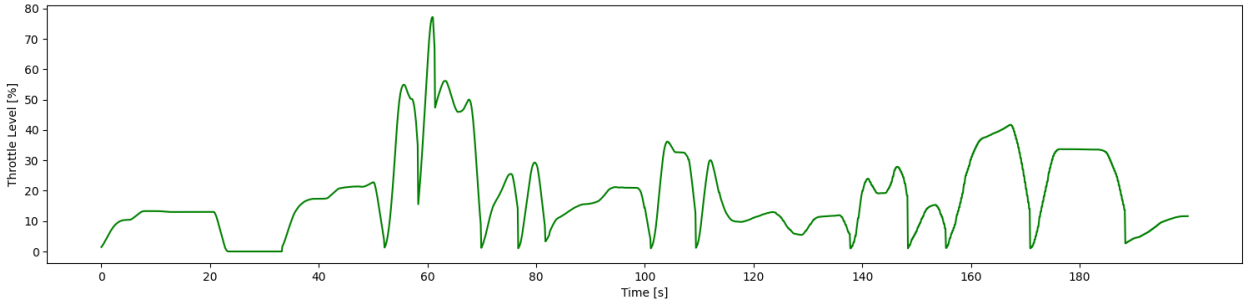


Figure 5.10: Throttle Level

5.2.2 Longitudinal Dynamics with Steering Maneuvers

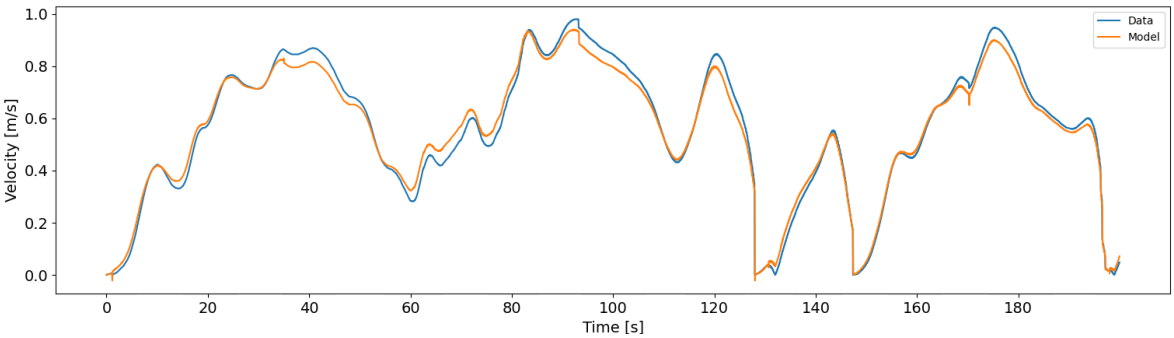


Figure 5.11: Gray Box — Non-Linear Model — Steering Maneuvers

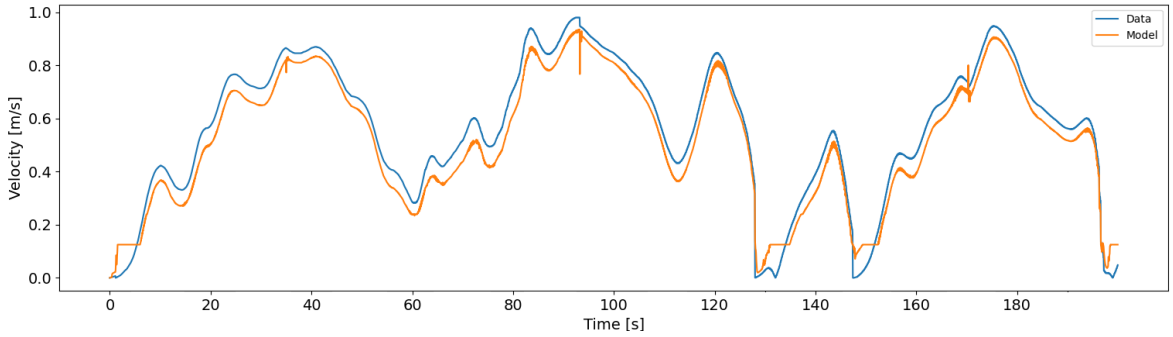


Figure 5.12: Black Box (2 inputs) — Non-Linear Model — Steering Maneuvers

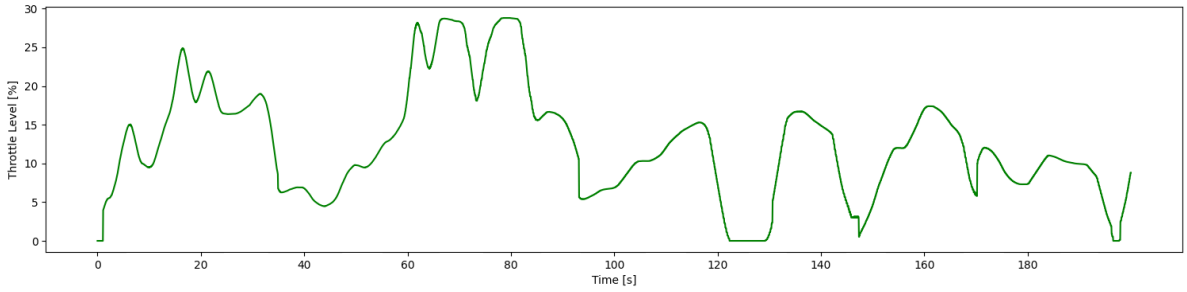


Figure 5.13: Throttle Level — Steering Maneuvers

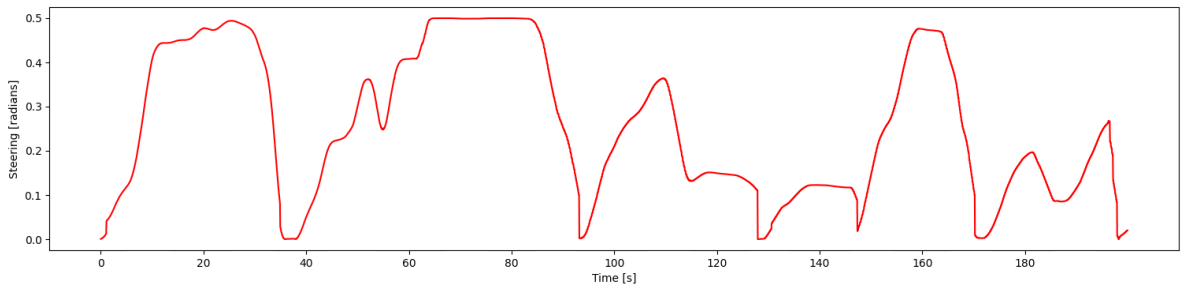


Figure 5.14: Steering Inputs

As already discussed in the previous chapters, in addition to the 2-inputs 1-output neural network, I have enlarged the regressor, designing a 3-inputs 1-output neural network.
By taking into consideration the steering angle as input to the model, its prediction performances should improve.
Let's see its plot:

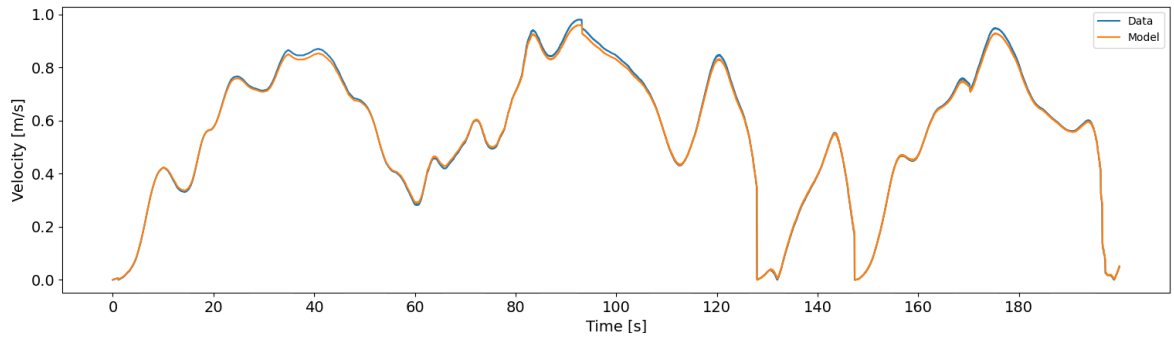


Figure 5.15: Black Box (3 inputs) — Non-Linear Model — Steering Maneuvers

Just looking at the plot, we can already realize that the performances are improved. But before jumping to any conclusions, let's analyze the metrics in the next chapter.

5.2.3 Metrics

As before, let's first inspect the metrics of the Non-linear models on the dataset without steering inputs:

Non-Linear Models Metrics — No Steering Maneuvers								
Non-Linear Models	MSE	MAE	MSLE	MdAE	nRMSE	R2	RMSE	RRSE
Grey Box Model	0.05508	**0.06118**	0.00696	**0.03460**	**0.03599**	**0.96218**	0.23470	**0.19446**
Black Box Model (2 inputs)	**0.00645**	0.06482	**0.00285**	0.05598	0.06488	0.93632	**0.08029**	0.25232

The models seem to have very similar performances. The grey box nonlinear model is slightly better, though.

Now let's check the metrics on the dataset with steering inputs, considering both neural network structures:

Non-Linear Models Metrics — Steering Maneuvers								
Non-Linear Models	MSE	MAE	MSLE	MdAE	nRMSE	R2	RMSE	RRSE
Grey Box Model	0.07014	0.08262	0.03423	0.04221	0.04329	0.94427	0.26483	0.22539
Black Box Model (2 inputs)	0.08644	0.11482	0.08285	0.15986	0.09174	0.87598	0.29401	0.25232
Black Box Model (3 inputs)	**0.00134**	**0.07342**	**0.00619**	**0.02557**	**0.01162**	**0.98804**	**0.03660**	**0.04423**

It is not surprising to see that the neural network trained with 3 inputs outperforms the other models.
Despite that, the grey box model shows a good robustness even when stimulating the system with steering maneuvers. We cannot say the same thing for the neural network trained on 2 inputs, which shows a degradation of performances.

5.3 Final Comparison

5.3.1 Longitudinal Dynamics without Steering Maneuvers

Linear Models Metrics — No Steering Maneuvers								
Linear Models	MSE	MAE	MSLE	MdAE	nRMSE	R2	RMSE	RRSE
Grey Box Model	0.06291	0.12506	0.00755	0.10913	0.03847	0.95681	0.25083	0.20782
Black Box Model	0.05929	0.09831	0.00916	0.06751	0.03735	0.95935	0.24351	0.20162

Non-Linear Models Metrics — No Steering Maneuvers								
Non-Linear Models	MSE	MAE	MSLE	MdAE	nRMSE	R2	RMSE	RRSE
Grey Box Model	0.05508	**0.06118**	0.00696	**0.03460**	**0.03599**	**0.96218**	0.23470	**0.19446**
Black Box Model (2 inputs)	**0.00645**	0.06482	**0.00285**	0.05598	0.06488	0.93632	**0.08029**	0.25232

In this case study, the grey box nonlinear model outperforms the other ones proving better performances in 5 out of 8 metrics. The neural network still remains a good candidate, performing better than the other models in 3 out of 8 metrics.

5.3.2 Longitudinal Dynamics with Steering Maneuvers

Linear Models Metrics — Steering Maneuvers								
Linear Models	MSE	MAE	MSLE	MdAE	nRMSE	R2	RMSE	RRSE
Grey Box Model	0.10448	0.13207	0.04271	0.22141	0.05296	0.92427	0.32323	0.32539
Black Box Model	0.18617	0.14682	0.09514	0.42529	0.11025	0.81614	0.43147	0.42879

Non-Linear Models Metrics — Steering Maneuvers								
Non-Linear Models	MSE	MAE	MSLE	MdAE	nRMSE	R2	RMSE	RRSE
Grey Box Model	0.07014	0.08262	0.03423	0.04221	0.04329	0.94427	0.26483	0.22539
Black Box Model (2 inputs)	0.08644	0.11482	0.08285	0.15986	0.09174	0.87598	0.29401	0.25232
Black Box Model (3 inputs)	**0.00134**	**0.07342**	**0.00619**	**0.02557**	**0.01162**	**0.98804**	**0.03660**	**0.04423**

Here, instead, we clearly have the best performances with the 3-inputs neural network model, which outperforms all the other linear and nonlinear models.

References

[1] R Rajamani. *Vehicle dynamics and control*. Springer, 2011.

[2] KH Johansson P Sahlholm H Jansson. *Road grade estimation results using sensor and data fusion*. 14th World Congress on Intelligent Transport Systems, 2007.

[3] N. Misron S. M. E. Fadul I. B. Aris. *Modelling and Simulation of Powertrain System for Electric Car*. Journal of the Society of Automotive Engineers Malaysia, 2018.

[4] Reinaldo Martinez Palhares Jullierme Emiliano Alves Dias Guilherme Augusto Silva Pereira. *Longitudinal Model Identification and Velocity Control of an Autonomous Car*. IEEE Transactions on Intelligent Transportation Systems, 2014.

[5] M Metwaly M Zaky E Elattar. *Decoupled Speed and Torque Control of IPMSM Drives Using a Novel Load Torque Estimator*. Advances in Electrical Computer Engineering, 2017.

[6] Xin Xu Fan Zhu Lin Ma. *Baidu apollo auto-calibration system-an industry-level data-driven and learning based vehicle longitude dynamic calibrating algorithm*. arXiv, 2018.

[7] Kecheng Xu Jiaxuan Xu Qi Luo. *An Automated Learning-Based Procedure for Large-scale Vehicle Dynamics Modeling on Baidu Apollo Platform*. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2019.