KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Learning comfort objective from lane change demonstrations for optimal control

Stijn Staring

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: elektrotechniek, optie Elektronica en geïntegreerde schakelingen

Promotor:
Prof. dr. ir. Jan Swevers

Assessoren:
Prof. dr. ir. Bert Pluymers
Prof. dr. ir. Herman Bruyninckx

Begeleider:
Dr. ir. Son Tong

Academiejaar 2019 – 2020

Preface

I would like to thank my family in the first place. During the history of my studies they always have been my biggest fans and I want to show my gratitude for the opportunities they have given me. I also want to thank my promoter Professor Swevers at the KU Leuven and Dr. Tong my mentor at Siemens for the professional discussions and tips they have given me in order to improve results. I also want to thank Flavia Acerbo, an employee at Siemens, who answered my questions on several occasions. Finally I also want to thank my aunt and sister for proofreading this thesis.

Stijn Staring

Contents

Abstract

The `abstract` environment contains a more extensive overview of the work. But it should be limited to one page.
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Samenvatting

In dit `abstract` environment wordt een al dan niet uitgebreide Nederlandse samenvatting van het werk gegeven. Wanneer de tekst voor een Nederlandstalige master in het Engels wordt geschreven, wordt hier normaal een uitgebreide samenvatting verwacht, bijvoorbeeld een tiental bladzijden.
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

List of Figures and Tables

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations

MS	Multiple shooting
Gs	Gsteeringfactor
OCP	Optimal control problem
MPC	Model predictive control
ODE	Ordinary differential equation

Symbols

	Vectors are printed in bold.
N	The number of control actions performed during the optimization 4.8.
N_{MPC}	amount of control points in the control horizon of 5.1.
V_{start}	Longitudinal speed at the start of the lane change.
V_{des}	Desired longitudinal speed during a lane change. Taken as V_{start} during this thesis.
y_{des}	Desired lateral distance travelled at the end of the lane change.
$\boldsymbol{\theta}$	Weights that have to be learned in the objective function 4.12.
$\boldsymbol{f}_{obs}$	Feature vector calculated from the observed data according to 4.2.2.
$\boldsymbol{f}_{learned}$	Feature vector as calculated according to 4.2.2 from the lane change following out solving 4.8.
$\boldsymbol{f}_{rel}$	Relative feature vector defined as $\frac{\boldsymbol{f}_{learned}}{\boldsymbol{f}_{obs}}$.
$\boldsymbol{r}$	The 2D path followed during the lane change maneuver.
t_r	The amount of throttle given.
δ	The angle of the front wheel of the bicycle model.
δ_s	The steerwheelangle, related to δ by G_s according to $\delta = \frac{\delta_s}{G_s}$.

Chapter 1

Introduction

1.0.1 Importance of topic

"Society expects autonomous vehicles to be held to a higher standard than human drivers." [16] This quote is setting the tone of the technology in autonomous driving. In order to be accepted to the public, autonomous vehicles should perform as least as good as the conventional human driver on parameters as for example safety. Despite widespread research on self-driving vehicles the acceptance by the user stays only limited.[2] The purchase behaviour of customers can be directly linked with comfort. Also in order to gain more trust by the public it is clear that the challenge of making autonomous vehicles as comfortable as possible, should be tackled. This immediately leads to the questions what comfort during driving exactly is and how to measure it. A survey was conducted by researchers of the university of Warwick with as research question: Do passengers prefer autonomous vehicles be driven as full efficiency machines or in a way that emulates averaged human behaviour? [20] The result suggested that a blend of both is appealing the most confidence by user of a autonomous vehicle.
Driving comfort is a personal experience and also depend on the current emotional state of the driver. This means that more than one driving style for autonomous vehicle-driving should be identified for a certain vehicle. [22] The state of the driver can be communicated with the vehicle at the start of each ride and different driving styles can be obtained by changing the parameters in the path planning algorithm.

Figure 1.1: Concept visualization of autonomous driving. (source: [6])

1.0.2 Link with previous studies and problem formulation

In order to identify the specific comfort preferences of the driver that are quantized by these parameters, the vehicle should be able to learn them by demonstration. [11] Despite that each driver has its own preferences, they are based on a common notion of comfort where only different trade-offs are made. For example some drivers prefer more aggressive driving behaviour than others which will manifests itself in a different trade-offs of different comfort criteria than for example a defensive drivers. This will later in this thesis be translated into a comfort objective where different weights are used in order to quantify different comfort trade-offs made. This approach is in literature called inverse optimal control because it is learning the objective function of the comfort optimization. The lower the outputted value of this comfort objective, the higher the measurement of attained comfort will be in the later path planning algorithm.

In order to find comfort criteria which can be used to distinguish different drivers, research about the common notion of comfort is necessary. Passenger surveys in public road transport about carsickness [19] have identified lateral acceleration as the primarily responsible for motion sickness. It is explained that drive style is a main factor to influence the amount of sickness and it was found that sickness is higher when drivers drove with a higher average magnitude of fore-and-aft and lateral motion. These effect were found far more significant than the effect of vertical vibrations. There is also a consensus reached about the contribution of continuous trajectories to the prevention of motion sickness and the natural feel of paths.[7] This means that higher order kinematic variables like accelerations and jerks also should be considered when measuring comfort.

1.0.3 Thesis objective

The goal of this thesis is to build further on the research of learning by demonstration [11] and to refine this idea in a good working practical application. The thesis is more concretely focussed in the ability to explain driver data and the practical implementation and validation of a learning algorithm that is able to capture user specific driving preferences in weights of a comfort objective function. The learning process is to be done offline and is based on an inverse optimal control approach. A comfort objective will be derived from literature to describe the common notion of comfort and this will be fitted on individual driver data to produce driver specific parameters. In the next step this objective function will be used in a path planning MPC formulation which will be calculating online feasible and comfortable paths whereafter an tracking MPC algorithm will follow it.

To conduct the inverse learning control there is first look at data generated by simulations where it is assumed that the vehicle is driving on an straight road and high way speed when executing manoeuvrers as lane changes and longitudinal accelerations.

An example lane change can be seen in Figure 1.2. Assumptions made during this manoeuvre To be able to make the generated data of high quality an MPC approach with a 15 degree of freedom vehicle model is used. Also in the learning algorithm itself a three degree of freedom non-linear bicycle model is used in order to adequately capture the different kinematic signals e.g. jerks and accelerations. Further there were comparisons made of different methods to learn from multiple datasets.

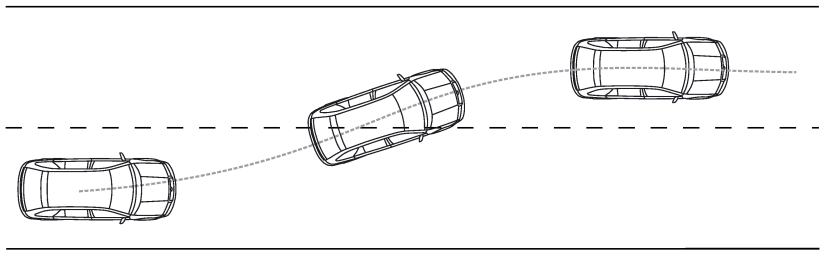


Figure 1.2: Example lane change as used as input in the inverse optimal control algorithm.

The execution of this research is conducted with the support of "Siemens Digital Industries Software - NVH R&D engineering department" located in Leuven which made it possible to preserve the direct link with reality. Software was made available e.g. Simcenter Amesim and the possibility to validate the obtained algorithms with real driver data made it possible to make the results that could be obtained more significant.

Chapter 2

Background in optimal control problems

This chapter gives some background information of the theory behind optimal control (OCP). After a global introduction and the discussion about the time discretization and shooting option, the chapter is giving an introduction into model predictive control (MPC).

2.1 Optimal control problem (OCP)

"An optimal control problem determines the desired inputs and corresponding state trajectories to change the system from an initial state to a desired final state in an optimal way while satisfying some input and state constraints." [12].

$$\begin{aligned} \min_{\boldsymbol{q},\boldsymbol{u}} \quad & \int_0^T l(\boldsymbol{q}(t),\boldsymbol{u}(t))dt + E(\boldsymbol{q}(T)) \\ \text{s.t.} \quad & \dot{\boldsymbol{q}}(t) = \boldsymbol{f}(\boldsymbol{q}(t),\boldsymbol{u}(t)) \\ & \boldsymbol{q}(0) = \boldsymbol{q}_0, \;\; \boldsymbol{q}(T) = \boldsymbol{q}_T \\ & \boldsymbol{q}(t) \in Q, \quad \boldsymbol{u}(t) \in U, \quad t \in [0,T] \end{aligned} \tag{2.1}$$

$\boldsymbol{q}$ is called the state vector and contains all the states of the system. $\boldsymbol{u}$ is the control vector containing the controls. In theory the amount of states of an system can be infinite, but in order to sufficiently model a system only a good chosen set of states is needed to sufficiently describe the system. In vehicle control in order to describe the driving behaviour often kinematic variables like positions and velocities of the centre of gravity of the vehicle are chosen together with the yaw angle. This fully describes the position of a vehicle in a 2D plane. Typical controls $\boldsymbol{u}$ are steerwheelangle and the amount of throttle which can be directly linked the amount of propulsion force. Also higher order controls are possible.
The objective function l of the optimization problem 2.1 is integrating over the desired control horizon T. In the objective function it is indicated what should be minimized and it is in function of the different states and controls. $E(\boldsymbol{q}(T))$ describes

the terminal cost and the value of the integrated objective is reduced when the system comes closer to a predefined final state at the end of the control horizon. The dynamics of the system are modelled by an explicit ordinary differential equation $\dot{\boldsymbol{q}}(t) = \boldsymbol{f}(\boldsymbol{q}(t), \boldsymbol{u}(t))$ and the evolution of the vehicle states during the control horizon are retrieved by integration of this ODE. Q and U represent the constraints that can be put on respectively the optimization variables. It is worth noting that the control horizon time T by itself can also be an optimization variable. What comes out of an OCP indicates what states will be visited by the system and which controls have to be applied in order to optimize the associated objective function with respect to predefined constraints. [14]

There is a difference between soft and hard constraints. A hard constraint directly demarcates the feasible solution areas $\boldsymbol{q}(t) \in Q, \quad \boldsymbol{u}(t) \in U$. A soft constraint is added in the objective function l and will contribute to a more optimal solution when it is better fulfilled. Also when a soft constraint is not met this can be an optimal solution which is not the case for a hard constraint. [21]

2.1.1 Time discretization

The optimal control problem 2.1 is continuous in time. This means that it has infinite dimensions, but to be able to run the optimization problem on digital systems there is need for discretization. There are several ways to do this which are summarized in Figure 2.1.[9]

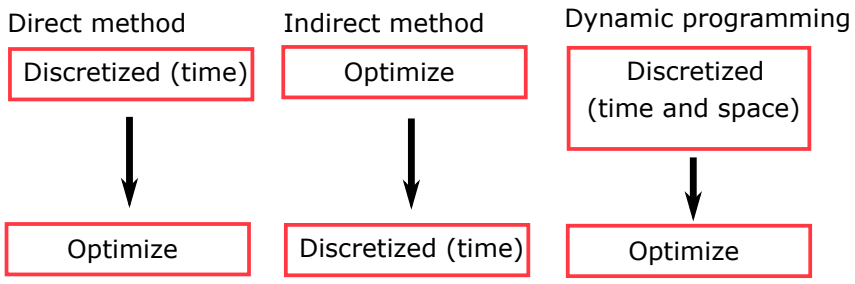


Figure 2.1: Overview of different discretization methods.

"Since direct methods are best suited to solve practically relevant OCPs" [12], this thesis is following the direct method. To implement the discretization of time when using the Direct method, a time shooting approach is used.

2.1.2 Time shooting

A shooting approach makes use of a time grid. Time will be sampled and on every time instant the optimal control problem is assessed. On these discrete points constraints will not violated but there are no limits on the amount of violation between the different time samples. In order to reduce the amount of constraint violation

the sampling rate should be taken high enough, while bearing in mind the extra optimization variables introduced and therefore calculation load. An other approach to take constraint violation between different sample points into account, is using spline based optimization formulations. [12]

Two different shooting approaches exist:

1. Multiple shooting (MS)
   During multiple shooting every new time sample $ns \in \mathbb{N}$ new states and controls are introduced and taken as optimization variables. Input changes are only allowed on the different time discrete time instances which leads to a piece wise control input signal. The blue bars indicate in Figure 2.2 (left) indicate the control value U_i that is applied to the system. The red dots are system states that are defined as optimization variables and are not constant during a time interval ΔT. In order to make the connection between states at time t_i and t_{i+1}, discrete time integration is used according to $\boldsymbol{q}(k+1) = \boldsymbol{f}(\boldsymbol{q}(k), \boldsymbol{u}(k))$. These connections are put as constraints in the discretized optimization formulation of 2.1 and are called in literature 'path closing constraints' [9].
   Equation 2.2 shows 2.1 when discretized with the multiple shooting approach.

$$
\begin{aligned}
\min_{\boldsymbol{q}(.),\boldsymbol{u}(.)} \quad & \sum_{k=0}^{N-1} l_k(\boldsymbol{q}_k, \boldsymbol{u}_k) + E(\boldsymbol{q}_N) \\
\text{s.t.} \quad & \boldsymbol{q}_{k+1} = \boldsymbol{f}(\boldsymbol{q}_k, \boldsymbol{u}_k) && k = [0, \cdots, N-1] \\
& \boldsymbol{q}_0 = \boldsymbol{q}_{measured} \\
& \boldsymbol{q}_k \in Q, && k = [0, \cdots, N] \\
& \boldsymbol{u}_k \in U && k = [0, \cdots, N-1] \\
& \boldsymbol{q}_N \in Q_f, \quad N \in \mathbb{N}
\end{aligned}
\tag{2.2}
$$

2. Single shooting (SS)
   In the single shooting or sequential approach which is visualized in Figure 2.2(right), only the first state and the controls are taken as optimization variables. Other states during the time horizon are derived from the initial state and the applied control. This is achieved by substituting the states during the control horizon by the integration results from the initial state and the applied controls. The mathematical formulation of this is shown by 2.3. In Figure 2.2 (right) are the states that result from integration indicated in green. [9]

$$
\begin{aligned}
\boldsymbol{q}^1 &= \boldsymbol{f}(\boldsymbol{q}^0, \boldsymbol{u}^0) \\
\boldsymbol{q}^2 &= \boldsymbol{f}(\boldsymbol{f}(\boldsymbol{q}^0, \boldsymbol{u}^0), \boldsymbol{u}^1) \\
\boldsymbol{q}^3 &= \boldsymbol{f}(\boldsymbol{f}(\boldsymbol{f}(\boldsymbol{q}^0, \boldsymbol{u}^0), \boldsymbol{u}^1), \boldsymbol{u}^2) \\
&\ldots
\end{aligned}
\tag{2.3}
$$

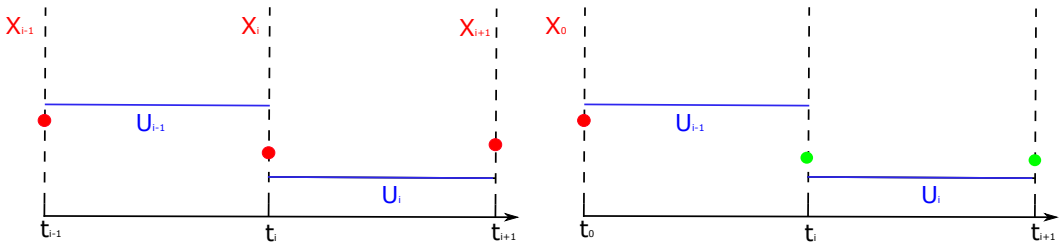


Figure 2.2: Schematic view of the time shooting approaches (left: multiple shooting; right: single shooting).

In this thesis a multiple shooting approach is used together with the use of a Runge-Kutta integration scheme. Runge-Kutta is an explicit integration scheme which has a higher calculation cost than a standard Euler scheme but is more reliable for non-linear systems and has a higher stability with respect to the chosen time-step. [12]

The difference between the two shooting approaches is that 'Multiple shooting'(MS) will lead to a larger Hessian of the objective and a larger Jacobian of the constraints. The reason for this is the introduction of more optimization variables. The advantage of MS is then again that these matrices are more sparse because a certain state is only dependent on the previous one and a control input which means they can be used in calculations more efficiently. In single shooting every state depends on the begin state and different controls which gives smaller, more populated matrices which are more calculation expensive. [9]

2.2 Model predictive control

In slow changing environments as for example a chemical plant, MPC is already a mature approach. More recently this technology also made its introduction in controlling systems with higher dynamics e.g. vehicle control due to an increase of computational power and the use of more efficiently algorithms. [12]. In the next section a short introduction of the formulation is made.

MPC is an approach were optimizations are solved in a loop in order to be able to notice disturbances, changing environments and model-plant mismatches. Therefore it makes use of a moving time horizon[1]. Every iteration an OCP is solved over the prediction horizon and as result control signals are given as output. Only the first control is applied for one time interval of the finite prediction horizon $N \cdot T_s$ as is visualized in Figures 2.3and 2.4.
The decision on the amount of samples N that are used in the control horizon is based on a trade off between higher accuracy and calculation effort. [18, 12]. In Figure 2.3 the solving of an OCP during one MPC iteration on time sample $t + 1$ is depicted. Figure 2.4 shows than only the first control of the obtained control signal

[1]There are also other implementations e.g. shrinking time horizon, where the prediction time gets smaller every step.

will be applied, which induces the system to come in a different state which will be the new start state and a new OCP will be solved. This will go on until the final desired conditions are met. A single OCP iteration only takes the model of the system into account as is done in a feedforward controller but through the iterative way of solving the OCPs, the MPC can still deal with unexpected acquired states due to feedback that is given by each initial state. As stated earlier the downside of this approach is a bigger computational load, which makes efficiently written software a necessity. [13]

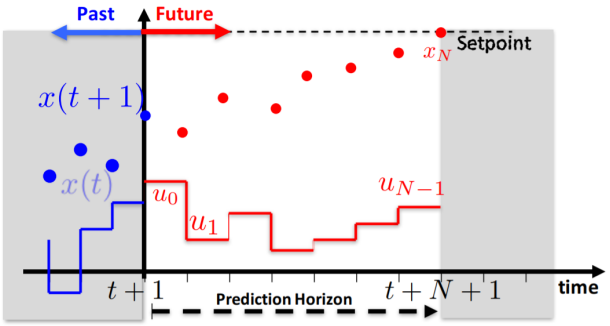


Figure 2.3: Visualization of the optimal control problem solved in one iteration of the MPC (Source: [13]).

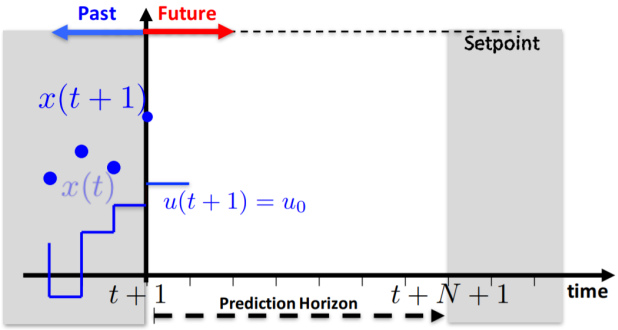


Figure 2.4: Visualization of the application of the first step of the calculated control signal during one iteration of the MPC (Source:[13]).

Chapter 3

State of the art modelling of comfort

As discussed in the introduction the goal of the thesis is to learn and implement a method to capture personal experience of comfort in autonomous driving. This will be done by using an inverse optimal control approach where the weights are learned from demonstration. To be able to do this it is necessary that a literature study is done about how to define comfort in a vehicle and to gain information about inverse optimal control.
This chapter will give an overview of the literature that is available and will show how the thesis fits in earlier conducted research.

3.1 What are the parameters that define comfort during driving?

In the following US patent [5] the idea is to assess the amount of comfort by calculating a value for carsickness. This value is calculated by a weighted sum of the sway motion, surge motion and heave motion of the vehicle. These motions are being directly calculated from the lateral acceleration, fore-aft acceleration and the vertical acceleration of the vehicle.

In the paper 'Investigating ride comfort measures in autonomous cars' [7], it is explained that due to the introduction of autonomous vehicles there will be an other perception of comfort. Figure 3.1 indicates in blue the claimed traditional comfort factors and in red the new ones that also have to be taken into account in when driving in autonomous vehicles. Concretely this can be translated into the preference of smooth trajectories and low lateral motions when the roads are assumed to be sufficiently smooth. A hypotheses is that motion sickness will be more prominent in autonomous driving due to the loss of control. Is is also argued that the amount of travel time and the distance to an obstacle are naturally parameters that contribute to a comfortable feeling.

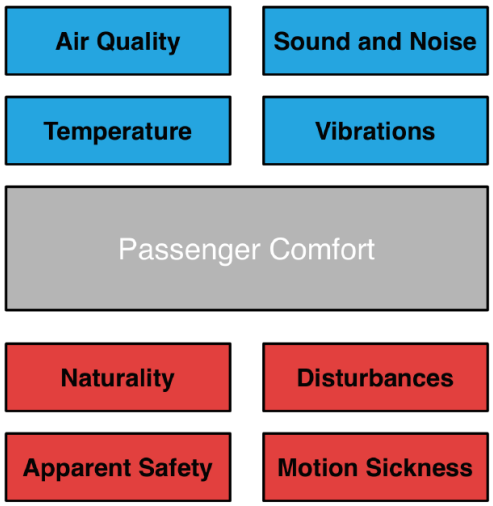


Figure 3.1: Overview of comfort parameters in autonomous vehicle with old parameters (blue) and new ones (red).(Source: [7])

In 'Analysis of Driving Style Preference in Automated Driving' [3] three studies are conducted in order to capture the definition of comfortable driving in autonomous vehicles. In the first study drivers drove manually with their own driving styles and this data was used in order to look for relevant metrics that could be used in defining distinct driver styles. It was found that these metrics are varying across the maneuver. This is a logical result because not in all maneuvers are all the comfort matrices equally dominant.
The results of this research is that accelerations play a key role in comfort but are not the only factor. [3]

The different comfort metrics found:

- lateral and longitudinal acceleration;
- lateral and longitudinal jerk;
- quickness of maneuver;
- headway distance;

Quickness of completing a lane change or an acceleration maneuver could respectively be modelled as: $q_L = \frac{\frac{\int velocity \cdot dt}{Time}}{\Delta distance}$ and $q_A = \frac{\frac{\int acceleration \cdot dt}{Time}}{\Delta velocity}$.

Comfortable driving as being assessed in [3] can be summarized as driving with small accelerations and jerk to obtain sufficient smooth trajectories and keeping sufficient headway distance in order to have a feeling of control and safety. These results suggest that when an algorithm to assess comfort is suggested for autonomous

vehicles, a notion of the surrounding traffic should be inherently present. It was found that in order of the traffic density the driver is more tolerant towards less smooth driving behaviour e.g. to be able to insert in a busy lane. In this case higher comfort can be attained if the driver has a feeling of a fast responds of the vehicle translating in early peaks of acceleration. Vertical vibrations come not into scope when roads are assumed sufficiently smooth.

In a second study the main metrics that were found from study one are varied and combinations are rated by the use of a survey about the amount of comfort. "Out of this it followed that accelerations are playing a key factor." [3] For lane changes it could be concluded that the maneuvers with a small lateral acceleration and early perceivable onset were more comfortable. It is further also hypothesised that the location of the acceleration peaks and the amount of symmetry of the acceleration signal also plays a role when the amount of comfort is rated.

The relation between jerks and comfort is also confirmed by [8] where it is stated that: "Jerk has been shown to elicit a stronger influence on comfort than acceleration".

3.1.1 Conclusion

In order to find parameters of driving comfort that will further on in the thesis will be used as comfort criteria, a literature study is conducted which is mainly questionnaire based. The results are that in order to be able to assess comfort higher order kinematic variables as accelerations and jerks should be taken into account. These are important variables in order to obtain a smooth vehicle path and to give a continuous and natural feeling when driving. Also should the quickness of the maneuver and the feeling of safety of the driver be taken into account. As quoted by [19] states that low frequency lateral acceleration are the mean responsible for carsickness. In addition comfort assessment is influenced by the environment because in busier traffic there is a higher tolerance for less smooth trajectories. It was also found that the effect of a fast reaction of vehicle at the start of a maneuver is influencing the amount of comfort perceived in a positive way.

3.2 Inverse reinforcement learning

Because every human has its own driving style it is a cumbersome task to tune these parameters for each individual in order to model a personal perception of comfort. In [15] it is showed that manual tuned parameters will lead besides also to sub-optimal solutions in comparison when the parameters are learned. That is why it is chosen in this thesis to derive these parameters by demonstration of a human driver.

The goal of the learning algorithm explained in this thesis is to derive the parameters of different linearly combined comfort criteria combined in the costfuntion J that when optimized as best as possible explain the observed data. As the match with the observed data gets better, the model as suggested by equation 3.1 is getting

closer to the inner comfort model of the individual driver. However it should be noted that the driver is taken unknowingly a lot of comfort criteria into account and they are not always linearly relating. Therefore the suggested comfort cost function consisting of a linear combination of comfort criteria will always be an approximation of the reality. Yet as discussed in [11] the results suggest that the magnitudes of the quantities that contribute to the comfort of the user are obtained.

$$J = \sum_{j=1}^{N} \theta_j \cdot f_j \tag{3.1}$$

In order to create a generative model that create vehicle paths r_i with equivalent kinematic characteristics as the path that was observed $\tilde{r_i}$, a feature-based inverse reinforcement learning is applied. [11, 1] With $i \in [1...m]$ and m the amount of observed trajectories. A feature is encoding relevant kinematic properties and the difference between the demonstrated and calculated features says something about the similarity of the kinematic vehicle signals.

3.2.1 Background on learning algorithm

A feature values maps a trajectory onto a scalar and the higher the scalar value the more discomfort is experienced. An example of a that measures the amount of accelerations in a maneuver is given by equation 3.2.

$$f : \boldsymbol{r} \rightarrow f(\boldsymbol{r}) = \int_0^T \ddot{x}(t)^2 + \ddot{y}(t)^2 dt \tag{3.2}$$

The path that that the centre of gravity of the vehicle is following can be represented by: 3.3.

$$\boldsymbol{r} : t \rightarrow \boldsymbol{r}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} \tag{3.3}$$

The driver is not a deterministic agent and is modelled by a stochastic distribution: $p(\boldsymbol{r}|\boldsymbol{\theta})$ For certain weights $\boldsymbol{\theta} \in \mathbb{R}^N$ a path $\boldsymbol{r}$ is produced as being a sample of a stochastic distribution. The distribution that is chosen for this is the distribution of maximum entropy (equation: 3.4). [4, 10]. This describes the data best because it is the distribution that is least biased. [11] The distribution with the highest entropy represents the given information best since it does not favour any particular outcome besides the observed constraints. [1]

$$p(\boldsymbol{r}|\boldsymbol{\theta}) = exp(-\boldsymbol{\theta}^T \cdot \boldsymbol{F}(\boldsymbol{r})) \tag{3.4}$$

Equation 3.4 can be interpreted as a linear costfunction $\boldsymbol{\theta}^T \boldsymbol{F}(\boldsymbol{r})$ where agents are exponentially more likely to select trajectories with lower cost. [11] The observed feature vector $\tilde{\boldsymbol{F}} \in \mathbb{R}^N$ has on its entries the different feature values $\tilde{f}_i$ with $i \in [1...m]$. The averaged observed feature vector is $\tilde{\boldsymbol{F}}_{av} = \frac{1}{m} \sum_{j=1}^{m} \tilde{\boldsymbol{F}}_j$.

In order to be able to explain the averaged observed feature vector, the weights $\boldsymbol{\theta}$ need to be found that match the expected features vector obtained from the distribution $\boldsymbol{F}(\boldsymbol{r}_{expected})$ with the averaged observed features vector $\boldsymbol{F}_{obs}$. $\boldsymbol{r}_{expected}$ defined as $E(p(\boldsymbol{r}|\boldsymbol{\theta}))$. To make the expected features vector match the averaged observed features vector, gradient descent method can be used with as gradient $\boldsymbol{F}_{obs} - \boldsymbol{F}(r_{expected})$. Equation 3.5 summarizes the gradient descent method with α the step size taken in the direction of descent.

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha \frac{\partial \boldsymbol{F}^k}{\partial \boldsymbol{\theta}} \tag{3.5}$$

When $\boldsymbol{\theta}_{optimal}$ is found the gradient is minimized and the features vectors will match as closely as possible. "When the feature expectations match, guaranteeing that the learner performs equivalently to the agent's demonstrated behaviour regardless of the actual reward weights the agent is attempting to optimize" [1]

In order to calculate $\boldsymbol{F}(\boldsymbol{r}_{expected})$ a Hamiltonian Markov chain Monte Carlo stochastic distribution sampling approached is used in [10]. In [11] a simplified method and less calculation demanding approach is proposed where it is assumed that the expected path is the one that is been assessed as the most comfortable by the driver and is thus the path that is minimizing 3.1 for certain weights. In order to be able to match with the observed data it is hereby assumed that the demonstrations not only are samples of a stochastic distribution but are also generated by optimizing an inner cost function which is similar to 3.1. Equation 3.6 summarizes the approximation proposed by [11].

$$\boldsymbol{r}_{expected} = \underset{\boldsymbol{r}}{argmax}\ p(\boldsymbol{r}|\boldsymbol{\theta}) = \underset{\boldsymbol{r}}{argmin}\ \boldsymbol{\theta}^T \cdot \boldsymbol{F}(\boldsymbol{r}) \tag{3.6}$$

$$\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\theta}} = \boldsymbol{F}_{obs} - \boldsymbol{F}(\boldsymbol{r}_{expected}) \tag{3.7}$$

It is checked in chapter 4 if this approach that is assuming that demonstrations are generated by minimizing a cost function also gives acceptable results when the assumption is violated.

3.2.2 RPROP algorithm

From [14] it is known that not every step size in the opposite direction of the gradient is leading to the convergence towards a minimum. When the step size is too small it will take a long time to convergence. However when it is chosen too big, cycling behaviour between limit points can occur. In order to avoid this kind of unwanted behaviour a line-search method is needed in order to change the step size taken during the course of the algorithm. For this the Resilient backpropagation algorithm (**RPROP**) [17] is used as it was first proposed by Martin Riedmiller and Heinrich Braun in 1993.

The main advantage when using RPROP is that the size of the gradient is not blurring the update value of the weights for the start of the next iteration in the gradient descent optimization method. The update value Δu is solely dependent of the sign of the current gradient and the sign of the gradient in the previous iteration. The main equations of the algorithm are given for each by:

$$\Delta u_i^t = \begin{cases} \eta^+ \cdot \Delta u_i^{t-1}, & if \ \frac{\partial f_i}{\partial \theta_i}^t \cdot \frac{\partial f_i}{\partial \theta_i}^{t-1} > 0 \\ \eta^- \cdot \Delta u_i^{t-1}, & if \ \frac{\partial f_i}{\partial \theta_i}^t \cdot \frac{\partial f_i}{\partial \theta_i}^{t-1} < 0 \\ \Delta u_i^{t-1}, & if \ \frac{\partial f_i}{\partial \theta_i}^t \cdot \frac{\partial f_i}{\partial \theta_i}^{t-1} = 0 \end{cases} \tag{3.8}$$

When the update value of the weight is determined it is applied in the direction of steepest descent which equals the opposite direction of the current gradient.

$$\Delta \theta_i^t = \begin{cases} -\Delta u_i^t, & if \ \frac{\partial f_i}{\partial \theta_i}^t > 0 \\ +\Delta u_i^t, & if \ \frac{\partial f_i}{\partial \theta_i}^t < 0 \\ 0, & else \end{cases} \tag{3.9}$$

Exception on 3.9:

$$\Delta \theta_i^t = -\Delta \theta_i^{t-1}, \ if \ \frac{\partial f_i}{\partial \theta_i}^t \cdot \frac{\partial f_i}{\partial \theta_i}^{t-1} < 0 \tag{3.10}$$

$$0 < \eta^- < 1 < \eta^+ \tag{3.11}$$

and with $\theta_i^{t+1} = \theta_i^t + \Delta \theta_i^t$, $i \in \mathbb{N}_{[1 \cdots N]}$ and $t \in \mathbb{N}_[1 \cdots \tau]$ the amount of iterations. Every time the partial derivative $\frac{\partial f_i}{\partial \theta_i}^t$ of the corresponding weight θ_i^t changes its sign, it is assumed that the last update was too big and the local minimum was passed. In 3.8 the step size is reduced and in order to go back the previous situation the update of the weight is done as 3.10. In order to not again decrease the update value when going back to the previous situation, in the next iteration $\frac{\partial f_i}{\partial \theta_i}^{t-1}$ is set equal to zero. If the derivative retains its sign, the update-value is slightly increased in order to accelerate convergence in shallow regions." [17] Parameters chosen bij the user are Δ_0, η^+ and η^-. In this thesis following values were chosen: $\Delta_0 = 0.1$, $\eta^+ = 1.2$ and $\eta^- = 0.5$.

3.2.3 Optimization principles

The optimization tool used is the CasADi software. This section discusses the main optimization principles used under the hood and discusses the IPOPT solver and SQP method. (See Handbook opti and slides CasADi 3 lecture vooral)

Conclusion

A background from literature on the inverse reinforcement learning algorithm used in this thesis was given. First inverse reinforcement learning was explained and how it could help to solve the research question how to learn comfort from observations.

Afterwards a more detailed background on the formulation of the specific learning algorithm was given, which was complemented with a discussion on the algorithm used to update the weights being part of the gradient descent method to match expected feature values with observed ones.

Chapter 4

Learning from ideal data

This chapter is focussing on the implementation of the inverse reinforcement learning idea that is used to learn the different weights in the comfort cost function $\boldsymbol{\theta}^T \boldsymbol{F}(\boldsymbol{r})$ explained in chapter 3. The use of ideal data is assumed which means that vehicle model mismatch is avoided by using the same model for learning the weights as generating the data. Thereby is the approximation that comfort of a driver can be modelled by a simple linear relation of features exactly for-filled. Data is generated by choosing a set of weights and generating kinematic vehicle signals by the minimization seen in equation 3.6. Perfect learning occurs when the chosen weights used in generating the data are found back.

First the non-linear bicycle model with the used parameters is explained. There is chosen for a bicycle model because this can model the dynamics sufficiently during a comfortable lane change maneuver and the calculation cost stays affordable. The model makes abstraction of suspension dynamics i.d. neglects rolling and pitching and the distribution of mass is fixed during driving. [21] Further on, the concrete used algorithm is discussed. After that a detailed discussion is given about the simulations done. It is worth noting that the learning process discussed, concerns an offline optimization which allows for higher calculations loads because of the absence of binding real time constraints.

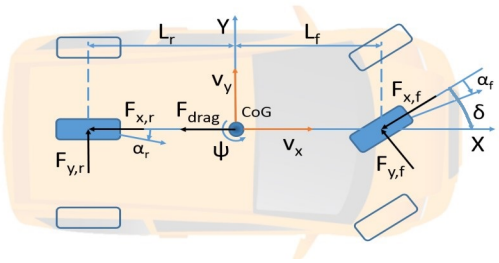


Figure 4.1: Non-linear vehicle bicycle model (Source: [18]).

4.1 Non-linear bicycle model

The free body diagram of the non-linear bicycle model can be seen in figure 4.1. Two variants of this model are discussed which are differentiated by the smoothness of the controls that can be received.

1. Model has 6 states and 2 inputs:

$$\boldsymbol{X} = \begin{bmatrix} x & y & v_x & v_y & \psi & \dot{\psi} \end{bmatrix}^T \; and \; \boldsymbol{U} = \begin{bmatrix} t_r & \delta \end{bmatrix}^T \tag{4.1}$$

2. Model has 10 states and 2 inputs:

$$\boldsymbol{X} = \begin{bmatrix} x & y & v_x & v_y & \psi & \dot{\psi} & t_r & \delta & a_x & a_y \end{bmatrix}^T \; and \; \boldsymbol{U} = \begin{bmatrix} \dot{t_r} & \dot{\delta} \end{bmatrix}^T \tag{4.2}$$

x and y in the above formulation are the position of the centre of gravity of the car in the global coordinate system. v_x and v_y are the vehicle velocities in the local vehicle frame.ψ is the vehicle yaw angle and $\dot{\psi}$ the yaw rate. The input vector of 4.1 consists of the throttle control input t_r and the angle of the front wheel δ. In the more extended bicycle model of 4.2 throttle and frontwheelangle serve as states. a_x and a_y are the total accelerations of the centre of gravity in the local vehicle frame. The inputs in this second formulation are the first order derivatives of throttle and frontwheelangle.

The equations of motion derived and checked in literature [18] are[1] :

$$\begin{aligned}
\dot{x} &= v_x cos(\psi) - v_y sin(\psi) \\
\dot{y} &= v_x sin(\psi) + v_y cos(\psi) \\
m\dot{v}_x &= F_{x,f} cos(\delta) - F_{y,f} sin(\delta) + F_{x,r} - F_{drag} + m v_y \dot{\psi} \\
m\dot{v}_y &= F_{x,f} sin(\delta) + F_{y,f} cos(\delta) + F_{y,r} - m v_x \dot{\psi} \\
\dot{\psi} &= \dot{\psi} \\
I_z \ddot{\psi} &= L_f (F_{y,f} cos(\delta) + F_{x,f} sin(\delta)) - L_r F_{y,r} \\
\dot{t_r} &= \dot{t_r} \\
\dot{\delta} &= \dot{\delta} \\
a_{tx} &= \dot{v}_x \\
a_{nx} &= -v_y \dot{\psi} \\
a_{ty} &= \dot{v}_y \\
a_{ny} &= v_x \dot{\psi} \\
j_x &= \dot{a}_{tx} + \dot{a}_{nx} \\
j_y &= \dot{a}_{ty} + \dot{a}_{ny}
\end{aligned} \tag{4.3}$$

[1] Appendix A shows the fully worked out jerk equations j_x and j_y.

The drag force is calculated as:

$$F_{drag} = C_{r0} + C_{r1} v_x^2 \tag{4.4}$$

, with C_{r0} the roll resistance and C_{r1} the air drag contributions.

To calculate the tyre forces, a linear tyre model is used instead of a more complex non-linear model e.g. Pacejka tire model. The longitudinal tyre forces are calculated as:

$$\begin{aligned} F_{x,f} &= \frac{t_r T_{max}}{2R_w} \\ F_{x,r} &= F_{x,f} \end{aligned} \tag{4.5}$$

R_w is the wheel radius and T_{max} a measure for the maximum torque the engine is able to supply. Because $F_{x,r} = F_{x,f}$ the longitudinal forces that are induced by the engine is equally distributed between front and rear axle (division by 2 in above equations). The coefficient t_r is the normalised amount of throttle that can be applied and has a value between -1 and 1 (negative for braking). In the bicycle model it is assumed that braking behaves the same as giving a negative amount of throttle. The lateral tyre forces are calculated based on the tyre slipangles α_f and α_r:

$$\begin{aligned} \alpha_f &= -atan(\frac{\dot{\psi} L_f + v_y}{v_x}) + \delta \\ \alpha_r &= atan(\frac{\dot{\psi} L_r - v_y}{v_x}) \end{aligned} \tag{4.6}$$

, resulting in:

$$\begin{aligned} F_{y,f} &= 2K_f \alpha_f \\ F_{y,r} &= 2K_r \alpha_r \end{aligned} \tag{4.7}$$

The use of this linearised lateral tyre model is valid for small lateral accelerations ($a_y <= 4m/s^2$) and slip angles ($\alpha <= 5^o$) [18]. It is acceptable to use this approximation model in this thesis as the goal is learn a comfortable and thus smooth lane change manoeuvre. These constraints will also be checked during the section 4.3.2.

The vehicle model parameters used are given in table 4.1. These correspond to common used vehicle parameters as provide by Siemens Digital Industries Software - NVH R&D engineering department. The $Gsteerfactor$ approximates linearly the relation between the frontwheelangle and the steeringwheelangle turned by the driver by the relation $\delta = \frac{\delta_s}{Gs}$.

Parameter	Value
Vehicle mass m [kg]	1430
Moment of inertia I_z $[kgm^2]$	1300
Front axle distance L_f [m]	1.056
Rear axle distance L_r [m]	1.344
Roll resistance coefficient C_{r0} [N]	0.6
Air drag coefficient C_{r1} $[\frac{Ns^2}{m^2}]$	0.1
Engine torque limit T_{max} [Nm]	584
Wheel radius R_w [m]	0.292
Lateral front tyre stiffness K_f [N]	41850.85
Lateral rear tyre stiffness K_r [N]	51175.78
Gsteeringfactor Gs [-]	16.96

Table 4.1: Used vehicle model parameters.

4.2 Formulation of the algorithm

The goal of the learning algorithm is to learn the weights $\boldsymbol{\theta}$ in the pre-defined comfort objective function: $\boldsymbol{\theta}^T \boldsymbol{F}(\boldsymbol{r})$. The features that are the entries of the feature vector $\boldsymbol{F}(\boldsymbol{r})$ capture a notion of comfort felled by the driver. Based on the literature study conducted in Chapter 3 and paper [11], the amount of discomfort can be modelled by the features discussed in 4.2.2 during timespan T of the maneuver. The scenario of a lane change on a highway is chosen. The time horizon where over the minimization of the comfort objective is itself also an optimization variable T. The simulations done in this thesis were performed on a notebook provided by Siemens with Intel Core i7-7920HQ CPU @ 3.10GHz and 32 GB of RAM memory.

4.2.1 Flow of the algorithm

The goal of the learning algorithm is to output weights that when applied in the objective $\boldsymbol{\theta}^T \boldsymbol{F}(\boldsymbol{r})$, generate feature vector $\boldsymbol{F}(\boldsymbol{r})$ that are the best possible fit with the observed feature vector. Without a vehicle mismatch a match of feature values will directly induce a good match of the kinematic signals as is extensively discussed in section 4.4. This means that similarity between the observed path and the one that follows from minimizing the objective $\boldsymbol{\theta}^T \boldsymbol{F}(\boldsymbol{r})$ for chosen weights, is quantified by the difference between the feature values of the observed path and the obtained one. The flow of a single data set learning algorithm can be seen in Figure 4.2. The path that is expected to be produced by the driver is the path that is felt the most comfortable and equals $\boldsymbol{r}_{expected} = \underset{\boldsymbol{r}}{argmin}\ \boldsymbol{\theta}^T \cdot \boldsymbol{F}(\boldsymbol{r})$

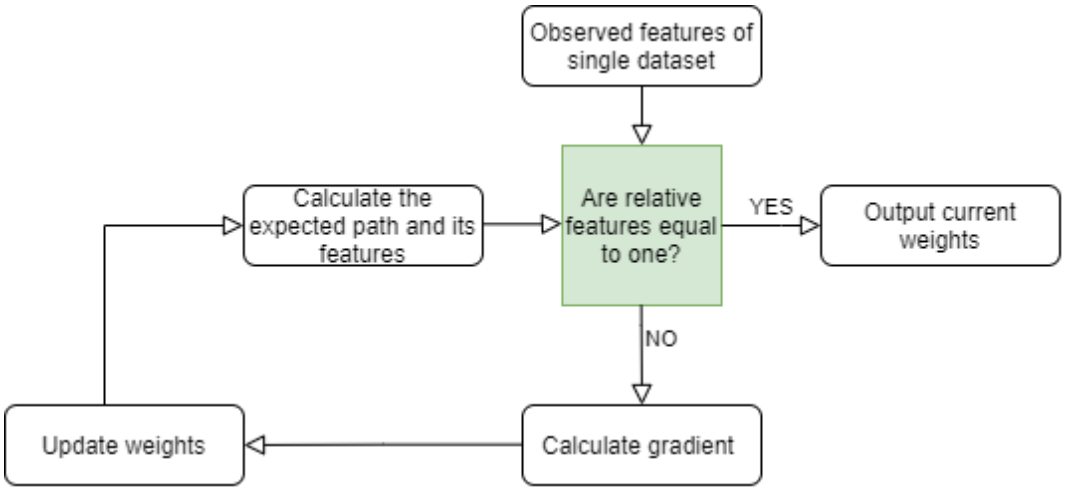


Figure 4.2: Basic flow of the reinforced learning algorithm.

The learning is started by guessing a set of weights e.g. all equal to one. Equation 3.6 is minimized in order to generate an expected path for these weights. From this calculated path, features for the observation and the calculated path from optimization 3.6,further addressed as learned features, can be retrieved by using the definition of the features discussed in 4.2.2.From this the relative features $f_{rel,i}$ are calculated by dividing the learned features by the observed features. A perfect match is acquired when the division equals one and the learning algorithm is terminated. The tolerance on convergence towards one, is chosen during this chapter equal to 10^{-3}.

While no convergence takes place the weights are updated, making use of the RPROP algorithm explained in Chapter 3. In order to apply the RPROP method only the sign of the gradient is used which means that the size of the gradient is decoupled from the update value of the weights. The estimate of the gradient is given by $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\theta}} = \boldsymbol{F}_{obs} - \boldsymbol{F}(\boldsymbol{r}_{expected})$ and the update of the weight is achieved by applying the gradient method according to $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - |\Delta\theta^k| sign(\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\theta}}^k)$ where $\Delta\theta^k$ is calculated according section 3.2.2. If follows that the weight θ_i is decreased if $f_{obs,i} > f_i(\boldsymbol{r}_{\boldsymbol{exptected}})$ and increased when $f_{obs,i} < f_i(\boldsymbol{r}_{exptected})$. the new weights can be used in the minimization of the comfort objective for a set of weights given by 3.6. Next a more detailed description of how this is done is given by 4.8 which uses as vehicle model 4.2.

$$
\begin{aligned}
\min_{\boldsymbol{X}(.),\boldsymbol{U}(.),T} \quad & \boldsymbol{\theta}^T \boldsymbol{F}(\boldsymbol{X},\boldsymbol{U},T) & \\
\text{s.t.} \quad & \boldsymbol{X}^{k+1} = I(\boldsymbol{X}^k,\boldsymbol{U}^k) & k=[0,\cdots,N-1] \\
& \boldsymbol{X}^0[1:8] = \boldsymbol{X}_{intitial} & \\
& T \leq T_{limit} & \\
& \boldsymbol{F}(\boldsymbol{X}^k) \geq 0 & k=[0,\cdots,N] \\
& \boldsymbol{H}(\boldsymbol{X}^k) = 0 & k=[0,\cdots,N] \\
& \boldsymbol{X}^k \in \mathbb{R}^{10\times 1} & k=[0,\cdots,N] \\
& \boldsymbol{U}^k \in \mathbb{R}^{2\times 1} & k=[0,\cdots,N-1] \\
& T \in \mathbb{R}, \quad N \in \mathbb{N} &
\end{aligned}
\tag{4.8}
$$

Where $\boldsymbol{X} \in \mathbb{R}^{10\times N+1}$ and $\boldsymbol{U} \in \mathbb{R}^{2\times N}$ contain respectively the states and controls 4.2 during the maneuver, complemented with the time of the maneuver T. In order to discretize time, a multiple shooting approach is adopted as is explained in section 2.1.1. The amount of integration intervals N is chosen equal to 1000 and is the same as the amount of control intervals in order to go from the initial state towards the end state. With this a sampling time of 0.025 s is obtained as is discussed in section 4.3.2. Inside the function I the Runge-Kutta time integration method is embedded in order to connect different states over time when a certain control is applied for ΔT. Here the equations of motion 4.3 are inputted in the optimization because to perform the integration, derivatives of the vehicle states are used. The time discretization used is can be categorized as a direct method and a multiple shooting approach is used which means that at every time instance a new set of state optimization variables is introduced as is explained in section 2.1.1. The path constraint vectors $\boldsymbol{F}$ demarcates together with the equality constraint vector $\boldsymbol{H}$ the feasible area of the solutions for $\boldsymbol{X}$, $\boldsymbol{U}$. An overview of these constraints is given by equations 4.9 and 4.10.

$$
\boldsymbol{F} = \left\{ \begin{array}{cc}
-\frac{Width\ Lane}{2} \leq y^k \leq \frac{3 \cdot Width\ Lane}{2}, & k=[0,\cdots,N] \\
0 \leq x^k, & k=[0,\cdots,N] \\
-\frac{\pi \cdot 150}{180 Gs} \leq \delta^k \leq \frac{\pi \cdot 150}{180 Gs}, & k=[0,\cdots,N] \\
-1 \leq t_r^k \leq 1, & k=[0,\cdots,N]
\end{array} \right\}
\tag{4.9}
$$

It is not necessary to introduce physical vehicle limits because these are present as soft constraints in the comfort objective $\boldsymbol{\theta}^T \boldsymbol{F}(\boldsymbol{X},\boldsymbol{U},T)$.

$$
\boldsymbol{H} = \left\{ \begin{array}{c}
y^N = Width\ Lane \\
vy^N = 0 \\
\psi^N = 0 \\
\dot{\psi}^N = 0 \\
\delta^N = 0
\end{array} \right\}
\tag{4.10}
$$

The constraints displayed in $\boldsymbol{H}$ make sure that at the end of the lane change the slipangles in the tires and the steerwheelangle are zero. From 4.3 this induces that the lateral velocity and acceleration and yaw acceleration also becomes zero and the lane change is completed. y^N makes sure that the wanted lateral distance is covered. This distance can be calculated from the start position of the vehicle in its lane and the width of the lane in order to end up at the centreline of the desired lane.
At the start of the lane change straight driving at constant longitudinal speed is assumed. To obtain this, the constraints of 4.11 are used. No constraints for accelerations are needed because this would give a redundancy due to the other initial states in combination with the motion equations 4.3. In 4.11 all initial states are zero excepts for the initial speed $v_{x,start}$ and $t_{r,start}$. The amount of throttle at the start of the lane change is chosen to overcome the aerodynamic drag without accelerating. This is given by $t_r^0 = \frac{(C_{r0}+C_{r1}v_{start}^2)r_w}{T_{max}}$. Therefore it can be concluded that the parameters that distinguish different lane changes are $v_{x,start}$ and $Width\ Lane$. This is exploited when generating different ideal lane change datasets.

$$\boldsymbol{X}_{initial} = \begin{bmatrix} x_{start} \\ y_{start} \\ v_{x,start} \\ v_{y,start} \\ \psi_{start} \\ \dot{\psi}_{start} \\ t_{r,start} \\ \delta_{start} \end{bmatrix} \tag{4.11}$$

The time limit constraint in 4.8 is needed in order to demarcate the optimization solution space. When set it has to take two conflicting criteria. It has to be chosen large enough in order to have a minor influence introduced by this constraint. Secondly it has to be taken small enough in order to preserve good conditions for the numerical integration performed in $\boldsymbol{F}(\boldsymbol{X}, \boldsymbol{U}, T)$. This is because the number of optimization points $N+1$ is fixed which means that a larger time limit will give a coarser time discretization. In order to serve these conditions time limit is set in this thesis on $25s$. This choice is validated in section 4.3.2. It is worth noting that with the removal of the time limit constraint the optimized comfortable lane change takes around 160 . This is not an realistic results because the objective will, as previously explained, not have good numerical properties.

In order to solve 4.8 an initial guess is needed for the longitudinal velocity in order to avoid the emergence of an invalid number. The default initial guess used in the CasADi software is an all zero vector. As can be seen in 4.6 this would give a division by zero.
To further enhance the solving speed of 4.8 also initial guesses are given for the other

vehicle states and additionally the controls. To do this, a feasible solution of the non-linear bicycle model for a lane change is needed. Therefore the initial guesses for $\boldsymbol{X}, \boldsymbol{U}$ *and* T are taken from the observed ideal data.

Another initial guess that speeds up the solving time of the IPOPT solver, is setting the initial guess of the lambda multipliers internally equal to the ones found during the previous call of 4.8 during the loop visualized by figure 4.2.

The solver used to calculate the states and control signals in 4.8 is the interior point solver IPOPT which is an open source solver. The idea behind it is to smoothing the KKT conditions and transform it to a smooth root finding problem. [14] Because IPOPT is a interior boundary method, a feasible initial guess is needed. Every time that the expected path has to be calculated as is visualised in flow diagram 4.2, the optimization 4.8 is performed with as initial guess the observed maneuver.

The time needed by the CPU in order to calculate the expected path for a certain set of weights and thus solving 4.8 takes around 5 s (python implementation) when a timelimit of 30 seconds and N equal to 1000 is chosen.

4.2.2 Objective function

The objective function used in 4.8 has to contain comfort felt by the driver and is based on the literature study displayed in section 3.1. The choice made in how to define the different features is important because it sets the fixed framework where the weights serve as parameters that steer the learning process towards a better match with the observations. It can be expected that the linear relation of features given by $\boldsymbol{F}(\boldsymbol{r})$ will serve as an approximation for the real, more complex comfort objective of a human drive. As is showed in [11] this linear approximation of features can already capture the main trends that contribute to an comfortable maneuver experience. As is suggested in [11] the feature framework that is further discussed in this section, can be validated and adjusted based on an user study in order to better match real driver observations.

When ideal data is used the observations are generated based on a known underlying comfort objective with known weights and feature framework. This permits the validation of the objective learning method discussed in this thesis. The remaining part of this section will discus the feature framework used in this thesis.

$$discomfort = \theta_1 \cdot f_1 + \theta_2 \cdot f_2 + \theta_3 \cdot f_3 + \theta_4 \cdot f_4 + \theta_5 \cdot f_5 + \theta_6 \cdot f_6 \tag{4.12}$$

$$f_i, \theta_i \in \mathbb{R} \quad i \in \mathbb{N}$$

Feature 1: longitudinal acceleration

$$f_1 : \boldsymbol{r} \rightarrow f_1(\boldsymbol{r}) = \int_0^T a_{x,total}^2(t)dt \tag{4.13}$$

Feature one is assessing the amount of discomfort by integrating the total longitudinal acceleration in the local axis. The local axis are fixed to the centre of gravity of the vehicle as can be seen in Figure 4.1. The total longitudinal acceleration $a_{x,total}$ is the sum of $a_{x,tangential}$ and $a_{x,normal}$ as described in 4.3.

Feature 2: lateral acceleration

$$f_2 : \boldsymbol{r} \rightarrow f_2(\boldsymbol{r}) = \int_0^T a_{y,total}^2(t)dt \tag{4.14}$$

Feature two is assessing the amount of discomfort by integrating the total lateral acceleration in the local axis. The total lateral acceleration $a_{y,total}$ is the sum of $a_{x,tangential}$ and $a_{x,normal}$ as described in 4.3.

Feature 3: longitudinal jerk

$$f_3 : \boldsymbol{r} \rightarrow f_3(\boldsymbol{r}) = \int_0^T j_x^2(t)dt \tag{4.15}$$

Feature three is assessing the amount of comfort by integrating the total change of longitudinal acceleration during the followed path.

Feature 4: lateral jerk

$$f_4 : \boldsymbol{r} \rightarrow f_4(\boldsymbol{r}) = \int_0^T j_y^2(t)dt \tag{4.16}$$

Feature four is assessing the amount of comfort by integrating the total change of lateral acceleration during the followed path.

Feature 5: desired speed

$$f_5 : \boldsymbol{r} \rightarrow f_5(\boldsymbol{r}) = \int_0^T (v_{des} - v_x)^2 dt \tag{4.17}$$

v_{des} is assumed to be a constant value and set equal to the start velocity just before the lane change.

Feature 6: desired lane change

$$f_6 : \boldsymbol{r} \rightarrow f_6(\boldsymbol{r}) = \int_0^T (y_{lane\_change} - y)^2 dt \tag{4.18}$$

y_{des} is a constant and is the desired lateral distance completed after the lane change. If the vehicle reaches faster its desired lateral displacement this is perceived as a

good responds and is interpreted as comfort of the driver as is discussed in section 3.1.

In order to implement the above defined integrals in the objective function of 4.8 discretization is needed. For this the Crank-Nicolson numerical integration is used as is shown in 4.19 for feature f_i.

$$\int_{f_i(t^n)}^{f_i(t^{n+1})} df_i = \int_{t^n}^{t^{n+1}} I(t) \cdot dt \tag{4.19a}$$

$$f_i^{n+1} - f_i^n = \frac{1}{2} \frac{I(t^{n+1}) + I(t^n)}{\Delta T} \tag{4.19b}$$

To summarize the objectives described by equation 4.12 consists out of a set of comfort features that model the amount of discomfort experienced during a maneuver. This is achieved by mapping kinematic signals onto scalar feature values through integration. By finding the driver specific weights $\boldsymbol{\theta}$ in 4.12, it is possible to model driver preferences between different comfort features. With this information an autonomous vehicle can perform path planning of the most comfortable path to do a lane change for a specific driver. As is shortly discussed in Chapter 3, the perception of save driving of an autonomous vehicle contributes to the amount of comfort that is experienced during driving. A perception of save driving comprises next to smooth behaviour also distances between other road agents. However features that consider the environment are not taken into account in 4.12 but this can be done if data of the position of other vehicles during the maneuver is available. Paper [11] gives some suggestions showed by equations 4.20 and 4.21.

$$f_d = \sum_{k=1}^{L} \int_0^T \frac{1}{(x_{o,k}(t) - x)^2 + (y_{o,k}(t) - y)^2} \cdot dt \tag{4.20}$$

$$L \in \mathbb{N}$$

With $[x_o, y_o]_k$ the position of the closest point of a different vehicle and L the total amount of vehicles in the nearby area.

Not only the bird's eye view distance between two different vehicles plays a role, but also the following distance of vehicle in the same lane is important. This can be modelled as suggested by [11]:

$$f_d = \int_0^T max(0, \hat{d} - d(t)) \cdot dt \tag{4.21}$$

The desired following distance $\hat{d}$ can be calculated based on the stopping distance of the vehicle when driving on a certain longitudinal velocity.

An other assumption that is not discussed in this thesis is the time limit to finish a maneuver. In a real life application however this limit often influences the maneuver.

After the weights are identified, the most comfortable path in a limited time span can be planned for a specific driver.

Normalization factors

In order to reduce the effect of order of size given by the units in the objective discussed in section 4.2.2, a normalization of the features is used. The kinematic signals of an example lane change are produced and the same features that are used in the objective function are calculated from it. These will be the normalization factors. Then the objective function is divided by the corresponding normalization factor which means that a feature that inherently gives a small feature value, will be divided by a small value and the other way around, an inherently large feature value will be divided by a large value. Table **??** gives an overview of the lane change normalization factors used in this chapter. The longitudinal jerk normalization factor is taken equal to the longitudinal one.

Normalization factor	Value
Nr.1	0.0073
Nr.2	2.64
Nr.3	0.0073
Nr.4	11.28
Nr.5	0.047
Nr.6	17.14

Table 4.2: Overview of normalization factors.

4.3 Ideal data

4.3.1 Generation

As mentioned above the term 'ideal data' concerns data that is generated with a non-linear bicycle model and exactly fulfils the assumption that the observations are minimizing an underlying comfort objective. More over the discomfort objective is the same as the one used in the learning algorithm which means that the generated path is a solution of the objective function discussed in section 4.2.2 for weights $\boldsymbol{\theta}$ known in advance. Because that the observations are using the same feature frame as the learning algorithm and the absence of vehicle model mismatch, weights can be learned that accurately explain the observations. This is translated in a accurate matching of the feature values discussed in 4.2.2. Because beforehand it is know what the weights should be in 4.12, the ideal data can serve as a validation of the learning algorithm discussed in section 4.2.

The relative weights $\theta_{r,i} = \frac{\theta_{abs,i}}{norm_i}$ chosen are $\left[4,\ 5,\ 1,\ 6,\ 1,\ 2\right]$, which gives as absolute weights $\left[549.75,\ 1.90,\ 137.44,\ 0.53,\ 21.43,\ 0.12\right]$ when the normalization of 4.2.2 is taken into

account. The objective that is used in 4.8 and outputs the ideal data is given by 4.22.

$$\frac{4}{0.0073} \cdot f_1 + \frac{5}{2.64} \cdot f_2 + \frac{1}{0.0073} \cdot f_3 + \frac{6}{11.28} \cdot f_4 + \frac{1}{0.047} \cdot f_5 + \frac{2}{17.14} \cdot f_6 \tag{4.22}$$
$$f_i \in \mathbb{R},\ i \in \mathbb{N}$$

Because of the normalization the relative weights will quantify the trade-offs between different comfort features without the disturbance of units used. Aside of this it allows to learn weights faster, because of the absence of big size differences that are present in the absolute weights. During the learning loop (Figure 4.2) the weights are set on a maximum of 1. When the absolute weights are learned instead of the relative ones, this would cause a substantially higher amount of iterations. When multiple ideal data sets have to be generated in order to serve as observations, the initial speed and width of the lateral distance are varied.

4.3.2 Validation

In this section the results of the generated ideal data are discussed. There is being look at an numerical vs analytical formulation, influence of the initial guess, choice of time limit, amount of optimization points, use of linear tire model and an discussion of the resulting feature values of the generated data.

Numerical vs Analytical formulation

In section 4.1 about the non-linear bicycle model, two different variants were described by equations 4.1 and 4.2. Variant one has only 6 states and the accelerations and jerks in the objective function 4.12 are calculated by making use of numerical differentiation described by equations 4.23. Variant two on the other hand uses the total accelerations as direct states in the vehicle model and uses an formulation of the jerk described in appendix A.

$$\frac{\partial \phi}{\partial t} = \frac{\phi(i+1) - \phi(i-1)}{2\Delta t} \tag{4.23a}$$

$$\frac{\partial^2 \phi}{\partial t^2} = \frac{\phi(i+1)\phi(i) + \phi(i-1)}{\Delta t^2} \tag{4.23b}$$

In order to validate the two approaches the generated lateral jerk signals are compared. It is clear that the analytical formulation (right figure) of the jerk gives more smooth results and less high peaks which is also seen in the more complex vehicle that is discussed in chapter 5. Therefore the analytical formulation approaches reality better. An other way to see this is that the analytical formulation contains more information about the vehicle, because the numerical equation of 4.23 are approximations of the equations shown in appendix A.

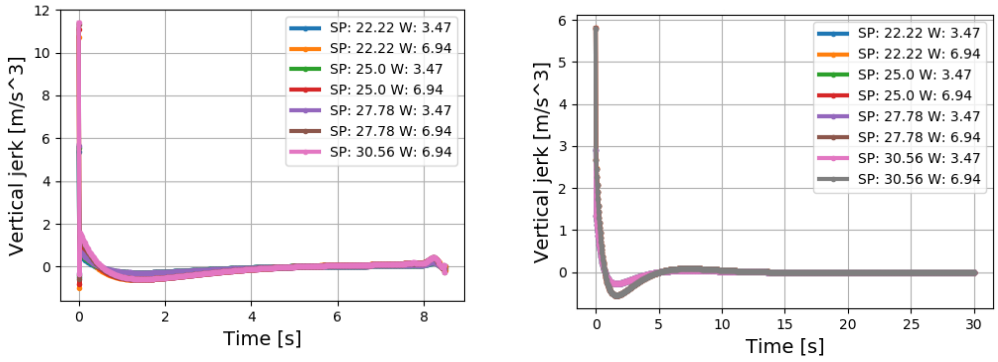


Figure 4.3: A comparison between the numerical jerk (left) based on 4.23 and 4.1 and the analytical jerk (right) based on appendix A and 4.2.

Initial guess

To determine if the generated data is a local solution of the optimization of 4.22, two different initial guesses are used whereof the obtained data feature values are summarized in table 4.3. In order to calculate the features, time limit is set on 30 s, N on 1000 and the parameters of the constraints of 4.8 to $v_{start} = 22.22\ \frac{m}{s}$ and $y_{des} = 3.47\ m$. From the results it can be concluded that the optimized features are not a local solution. In table 4.3 V_0 is the initial speed and L the lateral distance of the lane change used as initial guess. The initial guesses are generated with the numerical formulation and the observation is generated making use of the analytical approach. During the learning of the weights as described by Figure 4.2 the initial guess is set equal to the observed data.

Feature Value	V0:22.22 - L3.47	V0:25.00 - L6.94
Nr.1	6.83e-8	6.83e-8
Nr.2	0.37	0.37
Nr.3	1.77e-7	1.77e-7
Nr.4	0.57	0.57
Nr.5	1.98e-6	1.98e-6
Nr.6	30.94	30.94

Table 4.3: This table shows the retrieved feature values using different initial guesses in 4.8.

Time limit

In order to check the dependency of the generated data on the chosen T_{limit} constraint in 4.8, data is generated for an lane change with N, the amount of control points equal to 1000, initial velocity equal to 80 $\frac{km}{h}$ (V_{start}), a desired lateral displacement of 3.47 m (L) and a varying T_{limit} constraint as indicated in table 4.4. Figures that

show what the difference in the feature means for the different kinematic signals of the vehicle during a lane change, can be consulted in Appendix B

Feature Value	20 s	50 s	100 s
Nr.1	3.66e-8	1.13e-7	2.04e-7
Nr.2	0.37	0.38	0.38
Nr.3	1.13e-7	3.98e-7	1.56e-6
Nr.4	0.58	0.57	0.54
Nr.5	1.72e-6	2.27e-6	3.16e-6
Nr.6	31.05	30.74	30.42

Table 4.4: This table shows the retrieved feature values using different time limits in 4.8.

Looking at even greater time limits is not desirable because beyond a time limit of 100 s, the time discretization gets larger than 0.1 s for N equal to 1000 and thus more unreliable.

From the result of table 4.4 it can be concluded that the influence of the manually setting of the time limit in 4.8 can be neglected.

Amount of optimization points

The test carried out to investigate the dependence of resulting features values uses the same parameters as described in the previous section but fixes the time limit on 30 s and varies N over 500, 1000 and 1500 points. The results are shown in table 4.5 whereof it follows that a choice of N equal to 1000 is justified considering the small difference of the obtained features when N is equal to 1500 and a lower calculation load.

Feature Value	500	1000	1500 s
Nr.1	9.42e-8	6.63e-8	6.45e-8
Nr.2	0.38	0.37	0.37
Nr.3	5.61e-7	1.77e-7	1.11e-7
Nr.4	0.56	0.57	0.58
Nr.5	2.50e-6	1.98e-6	1.85e-6
Nr.6	30.66	30.94	31.05

Table 4.5: This table shows the retrieved feature values using different amount of control point N in 4.8.

Linear tire model

In this section it is checked if the conditions to use a linearised lateral tire model is valid. In literature [18] it was found that this is the case when there are small lateral accelerations ($a_y <= 4m/s^2$) and slip angles ($\alpha <= 5^o$) during the maneuver. Figure ?? gives the total lateral acceleration during a lane change maneuver that moves around two lanes or an estimated lateral distance of $6.94m$. Figure 4.5 shows the slip angle during this maneuver. From the graphs it can be concluded that the linearisation of the lateral tire forces is valid and there is no need for a more complex tyre model in 4.8. Both resulting figures are generated with the complex vehicle model discussed in chapter 5.

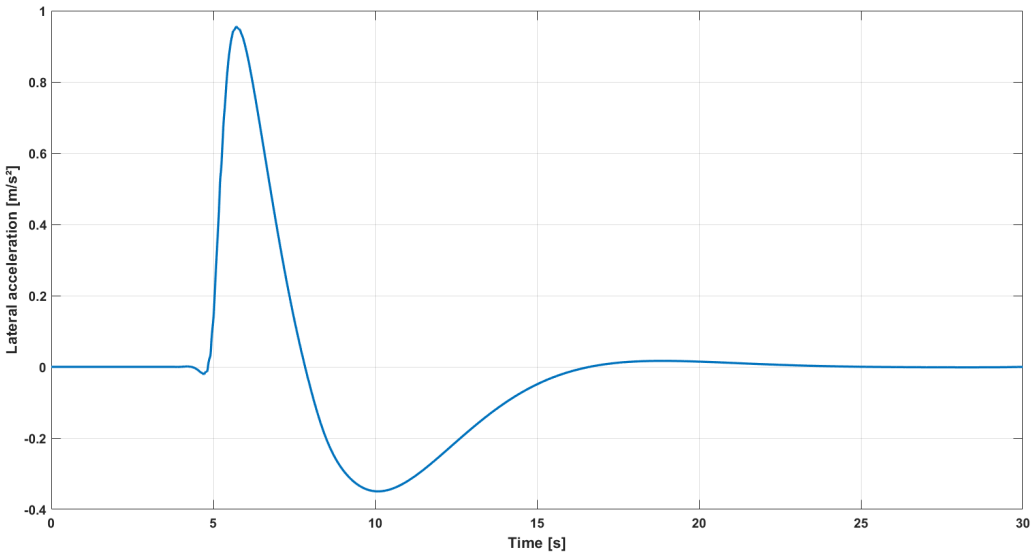


Figure 4.4: Lateral acceleration during a lane change $V_0 : 25.00\frac{m}{s}$ and $L : 6.94m$ generated with the amesim model.

Feature values

In the above tables it can be seen that the first, third and fifth features concerning longitudinal behaviour of the vehicle are very small. This means that during a lane change these features have a low influence. It can be expected that for the values with an order of size in computer accuracy, it becomes hard to accurately learn weights that generate feature values that match with these small values. This will be further discussed in 4.4.1.

An other interesting thing to note about the ideal data generated features is that when the initial speed of the lane change is varied and the desired lateral displacement stays the same ($3.47m$ or $6.97m$), because of the small longitudinal features values the data features found are quasi identical.

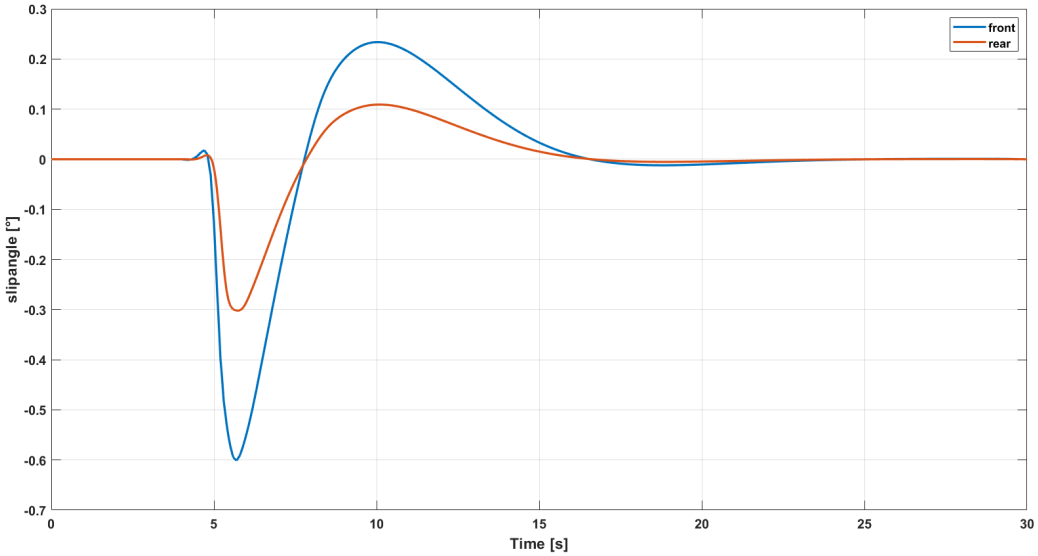


Figure 4.5: Slipangle during lane change $V_0 : 25.00\frac{m}{s}$ and $L : 6.94m$ generated with the amesim model.(bleu:front,red:rear)

Conclusion

From the above results it the influence of the choice of taking the time limit constraint equal to 30 s and the amount of control points equal to 1000 further on in this thesis, has been validated. Next it is shown that it is justified to use a linearised tire model. Afterwards an numerical and analytical formulation to model the total accelerations and jerk is discussed. Finally the resulting feature values of the ideal data are shortly look into.
write something about small longitudinal features and resulting bad weight learning.

4.4 Ideal lane change data learning results

In this section the results of the learning algorithm which makes use of ideal data is discussed. The first result discussed concerns the learning of a single dataset with $V_0 : 22.22\ \frac{m}{s}$, $L : 3.47\ m$. It is known beforehand that the weights used to generate the data are $\left[4,\ 5,\ 1,\ 6,\ 1,\ 2\right]$[2] and the initial guess of the weights is chosen as an all one vector. The convergence criteria to stop the learning algorithm displayed in Figure 4.2 is when the maximum amount of iterations equal to 300 is reached or if the feature values which use the learned weight match accurately the ones of the observed data according to following formula $f_{rel,i} = \frac{f_{calc,i}}{f_{obs,i}} \leq 10^{-3}$. The generated data features that are supposed to be matched in section 4.4.1, can be seen in table 4.3. Convergence is reached when the three lateral features, that dominantly define

[2]The features complying with the chosen weights are $\left[f_{ax},\ f_{ay},\ f_{jx},\ f_{jy},\ f_{diffvx},\ f_{diffy}\right]$.

the lane change, are accurately fitted for a certain set of weights.

In reality one observation demonstrated doesn't capture perfectly the preference of the human driver. In order to do so, learning of multiple datasets is needed. Therefore the conflict and averaging method are discussed. Afterwards a comparison is made and follows a conclusion.

4.4.1 Single dataset learning

The resulting weights found after 28 iterations are $\left[14.469, 5.000, 3.045, 5.977, 3.689, 1.998\right]$ in which the weight concerning the lateral acceleration is taken as reference in order to compare the learned weights with the chosen ones. $\boldsymbol{f}_{rel}$ at the end of the learning process is given by $\left[0.9907, 0.9995, 1.0037, 1.0006, 0.9941, 1.0001\right]$. When the algorithm is manually set to run for 121 iterations following weights are learned $\left[10.021, 5.000, 2.179, 5.998, 2.538, 2.002\right]$ which lay closer to the chosen ones. $\boldsymbol{f}_{rel}$ is given by $\left[1.0004460, 1.0000014, 0.99762746, 1.00000585, 0.98924983, 0.99999986\right]$ and a clear difference between the lateral weights that have an increased learning accuracy and the longitudinal weights that seem to be constraint in their accuracy. As already suggested in section 4.3.2 this is because of the small size of the feature values of the longitudinal direction which come in the range of computer accuracy during the optimization of 4.8. Not very accurate learned longitudinal feature will have a negligible influence on the overall behaviour of the planned path of the lane change and as can be seen, a range of weights will give a match of the longitudinal feature values around 10^{-2}. It is good to notice that just scaling of the feature values in the calculation of the gradient will not help because the RPROP algorithm operates only based on the sign of the gradient. In order to learn the longitudinal weights of the human driver accurately, a maneuver should be considered were these features will be more prominent e.g. an acceleration maneuver or different features can be chosen e.g. $a_tot^2 = a_x^2 + a_y^2$. Because the interest in the longitudinal weights during a lane change is marginal this is not further discussed during this thesis.

It could be argued that if the longitudinal feature weights are not so important that they could be removed from the objective 4.12. This is however not an correct assumption. The longitudinal features are less determinative during a lane change and therefore different weights give rise to the same lane changes. However they still play a roll in comfortable path planning in order to avoid nervous throttle and consequently longitudinal jerk and acceleration behaviour.

4.4.2 Averaging method

In order to simultaneously learn from multiple datasets the averaging method is proposed. [11] The flow of the algorithm is similar as shown in Figure 4.2 and the convergence criteria stays the same as for single dataset learning, but instead of inputting an observed feature vector based on a single dataset, an averaged one over multiple datasets is used. To calculate the gradient, the difference between

the averaged observed feature vector and the averaged calculated feature vector has is taken. In order to obtain the averaged calculated feature vector, m times the optimization 4.8 is called for each observed maneuver with varying initial speeds and desired lateral displacement. Afterwards the individual found feature vectors are averaged. m is the amount of combined datasets.

The datasets chosen to perform the learning algorithm on are $V_0 : 22.22\ \frac{m}{s},\ L : 3.47\ m$, $V_0 : 25.00\ \frac{m}{s},\ L : 3.47\ m$ and $V_0 : 22.22\ \frac{m}{s},\ L : 6.94\ m$. The resulting weighs and average $\boldsymbol{f}_{rel}$ found after 25 iterations are respectively $\left[\,14.459,\ 5.000,\ 3.204,\ 5.984,\ 3.720,\ 2.000\,\right]$ and $\left[\,0.9981,\ 0.9996,\ 0.9829,\ 1.0000,\ 0.9946,\ 1.0001\,\right]$. The $\boldsymbol{f}_{rel}$ of the individual datasets can be seen in table 4.6.

Feature	V022.22 - L3.47	V022.22 - L6.94	V025.00 - L3.47
Nr.1	1.0000	0.9973	1.0064
Nr.2	1.0002	0.9992	1.0001
Nr.3	0.9860	0.9816	0.9983
Nr.4	1.0012	0.9995	1.0010
Nr.5	0.9972	0.9948	0.9902
Nr.6	0.9999	1.0002	0.9999

Table 4.6: This table shows the $\boldsymbol{f}_{rel}$ for each individual dataset at convergence using the average method.

Figure 4.6 shows the three initial guesses of the paths that make use of an all one weight vector in red, purple and brawn. The finally learned paths can be seen in pink, grey and yellow. These paths lay exactly on the observed ones. From this it can be concluded that matching of feature values which are scalars, give a good representation for also a good matching of the 2D kinematic vehicle signals. The rest of the kinematic signals from the non-linear bicycle model can be seen in Appendix C.

The progress towards convergence over the iterations is showed by Figure 4.7.

Figure 4.8 gives a view of how the learned weights change over the iterations for the different features. The weights in these graphs likewise a solution of 4.8 as the ones discussed earlier, but differ in a scaling factor $\frac{5.0}{\theta_2}$. During the learning process the degree of freedom of the just appointed scaling factor can be removed by fixing the second weight equal to 5.000. Convergence is reached after 59 iterations and following weights are outputted $\left[\,15.143,\ 5.000,\ 3.222,\ 6.016,\ 3.880,\ 2.007\,\right]$. It takes the algorithm longer to find equivalent weights because of the lost of the ability of the RPROP algorithm to vary all the weights. Therefore the scaling degree of freedom is retained during the rest of this thesis.

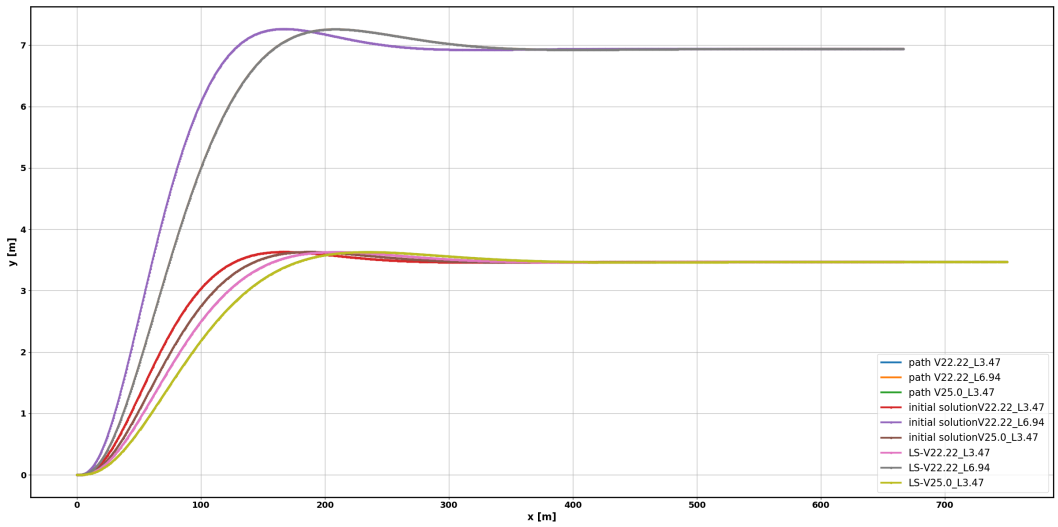


Figure 4.6: Observed, initial and learned paths for 3 different observed lane changes generated with common underlying weights.

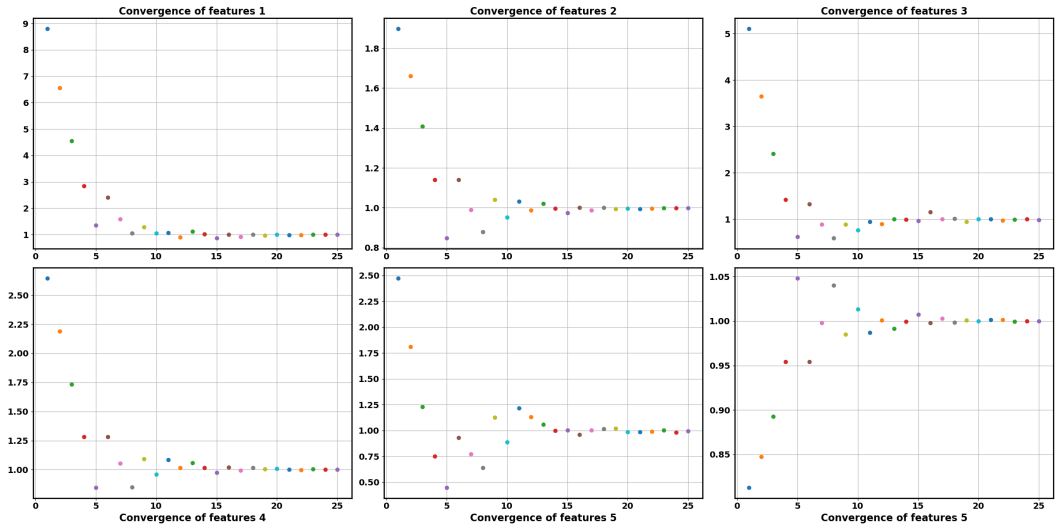


Figure 4.7: The evolvement of the average $\boldsymbol{f}_{rel}$ over the learning iterations.

4.4.3 Conflict method

The second method proposed is the conflict method. Its main idea is to increase or decrease the weights if the gradient, as given by 3.7, of the different individual datasets point all in the same direction. If there are individual gradients with conflicting signs, the update direction is ambiguity and the concerning weight is not updated. The updating will be resumed when the conflict is resolved by updating the other weights. Resolving conflicts is possible because the features are not independent of each other. Figure 4.9 shows how the conflict check is integrated together with the

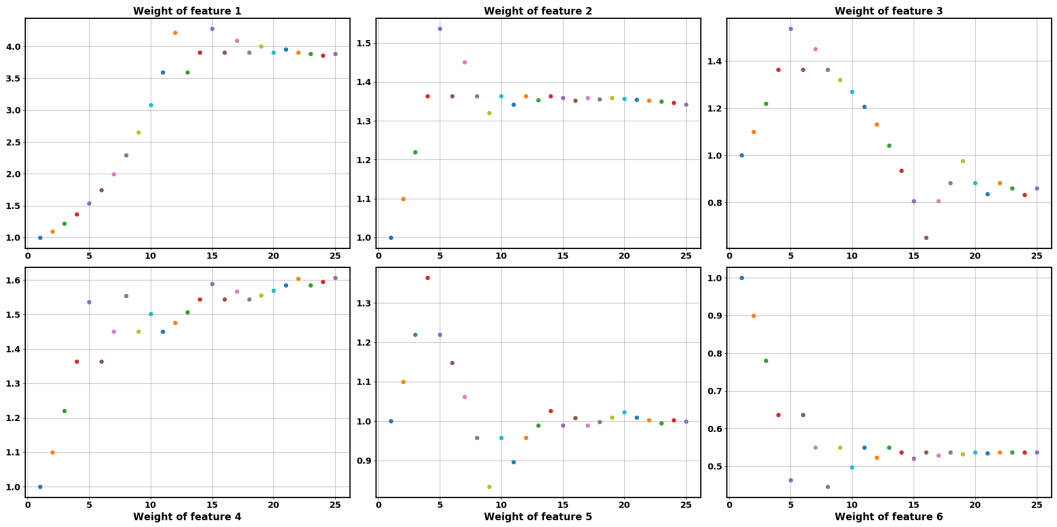


Figure 4.8: The different outputted weights over the learning iterations.

RPROP algorithm in the 'update weights' box in the basic learning flow of Figure 4.2. The bleu boxes serve as output ports.

Depending if the previous case was RPROP case 2 and the sign difference between the current and previous gradient, three distinct RPROP cases can be identified which outputs the new update value, delta weights and the exception value as discussed in section 3.2.2. Inside the conflict test, it is checked if all the signs of the current gradients are consistent. This boils down to verifying on which entry in the different gradients there is a sign difference. If there is a sign difference, this means that for one dataset the learned feature value is higher than the observed one and the corresponding weight should be increased (negative gradient) and for at least one other dataset, the corresponding weight should be decreased. (positive gradient) according to 3.5 and 3.7. For such a case the conflict test will give rise to a positive conflict value. If there is no conflict there will also be unity in the decision of the RPROP case. After the conflict is solved, the next case will be automatically equal to case 3 because no decision can be made about the increase or decrease of the update value. The convergence criteria used stays the same as discussed in 4.4.2 but now an additional criteria is added that there should still be improvement possible. This means that it should be possible to update a weight in a direction that benefits every dataset.

The learning algorithm is applied on the same three datasets as discussed in section 4.4.2 and uses as initial guess of the weights an all one vector. The resulting weighs found after 32 iterations are $\left[\,14.283,\ 5.000,\ 3.114,\ 5.979,\ 3.645,\ 2.000\,\right]$. The learning algorithm stopped because no update of the weights could be done without ambiguity. The $\boldsymbol{f}_{rel}$ of the individual datasets can be seen in table 4.7. Figure 4.10 shows the convergence of the $\boldsymbol{f}_{rel}$ vector of dataset $V_0 : 22.22\ \frac{m}{s}$, $L : 3.47\ m$. The individual convergence plots are similar for the different datasets.

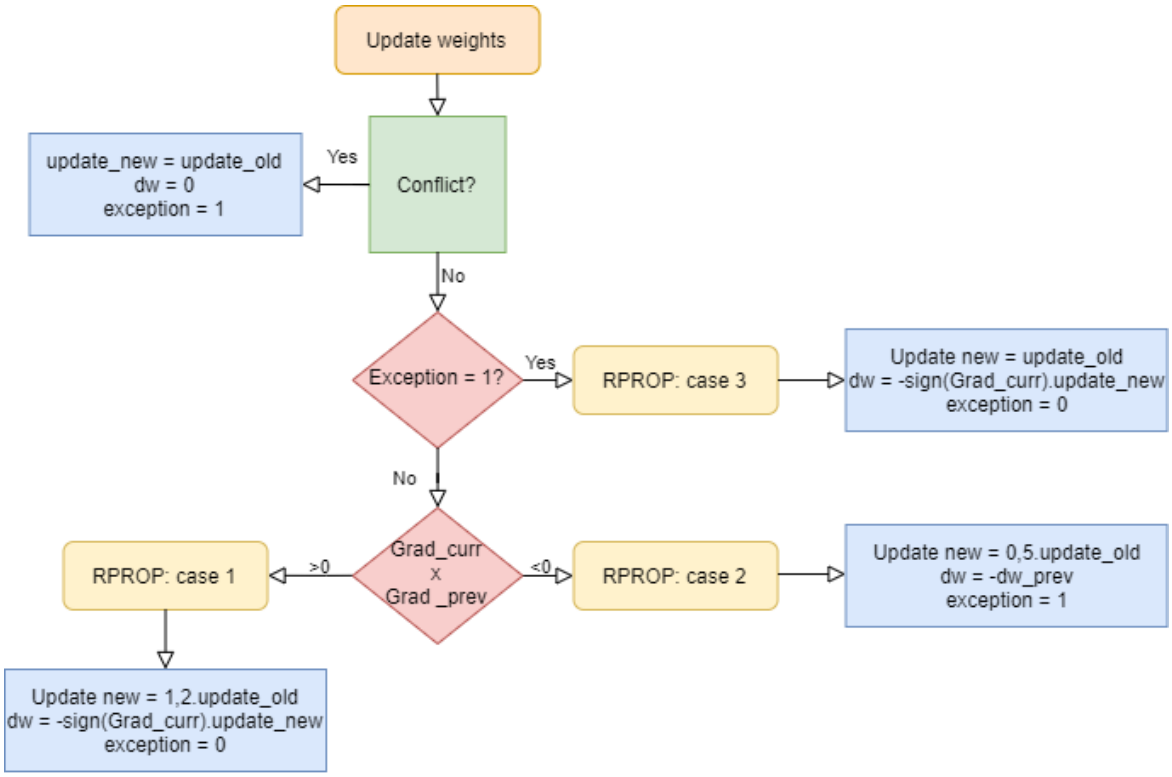


Figure 4.9: Flow of the conflict method as part of the basic flow diagram of Figure 4.2

f_{rel}	V022.22 - L3.47	V022.22 - L6.94	V025.00 - L3.47
Nr.1	0.9939	0.9913	1.0003
Nr.2	1.0003	0.9993	1.0002
Nr.3	0.9911	0.9868	1.0032
Nr.4	1.0016	0.9999	1.0014
Nr.5	1.0009	0.9985	0.9941
Nr.6	0.9998	1.0001	0.9998

Table 4.7: This table shows the f_{rel} for each individual dataset at convergence using the conflict method.

As can be seen in table 4.7 and Figure 4.10 the conflict method is capable to give an adequate convergence towards the observed features. As is demonstrated in section 4.4.2 when the feature values match, there is also a good match between the learned and demonstrated kinematic signals.
The algorithm was stopped because no improvement could be realized anymore. It could be hypothesized that if no ideal data is used where the human driver is less consistent in producing observations that resemble its underlying weights, the algorithm is stopped even earlier. This will naturally constrict the algorithm in

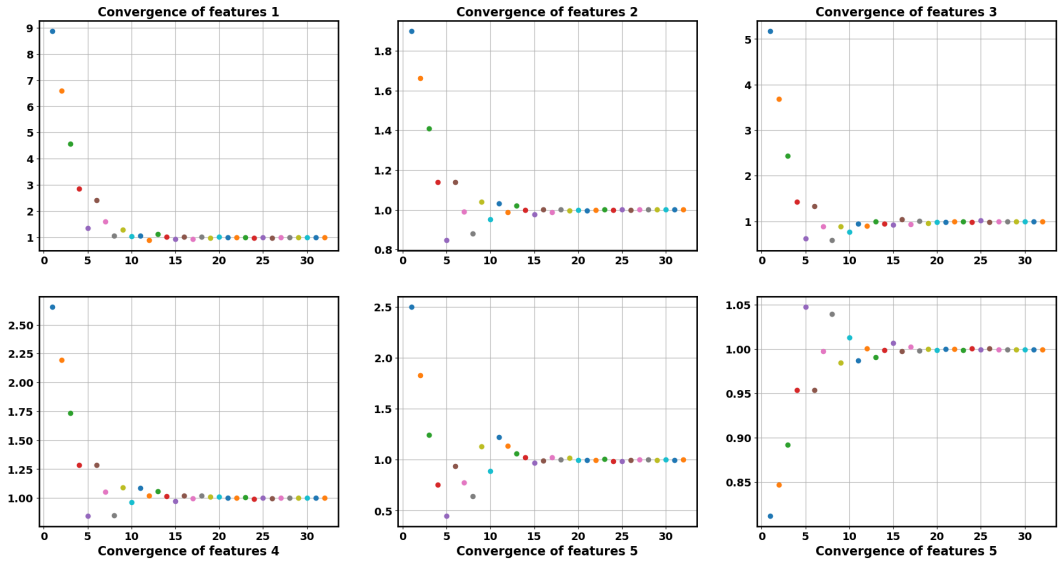


Figure 4.10: The evolvement of $\boldsymbol{f}_{rel}$ of dataset V_0 : 22.22 $\frac{m}{s}$, L : 3.47 m over the learning iterations.

its feature matching accuracy due to no available direction of improvement. This can already be seen when more datasets are used in the conflict method e.g. 7 datasets. The algorithm stops then at 25 iterations because it reached earlier a point of no possible improvement anymore. The learned weights for 7 datasets are $\left[14.785,\ 5.000,\ 3.223,\ 5.968,\ 3.770,\ 2.000\right]$. Figure 4.11 gives the averaged error that each individual $\boldsymbol{f}_{rel}$ vector made with respect to convergence to one for the three important lateral features. The three important lateral features shown in Figure 4.11 are respectively lateral total acceleration, lateral jerk and lateral distance displaced. The individual $\boldsymbol{f}_{rel,i}$ at convergence shows how good the match of feature values is when applying the learned weights at 4.8 with the same parameters for v_{start} and $y_{desired}$.

Figure 4.11 shows that the prematurely stop of the learning with 7 datasets in comparison with 3 datasets gives an slightly higher error on the match of feature values of the individual datasets.

4.4.4 Comparison of methods

In this section a comparison is made between the average and conflict method of combining multiple datasets. Figure 4.12 shows the averaged error defined as

Because the averaging method gives slightly more accurate feature matching further in this thesis the averaging method is used.

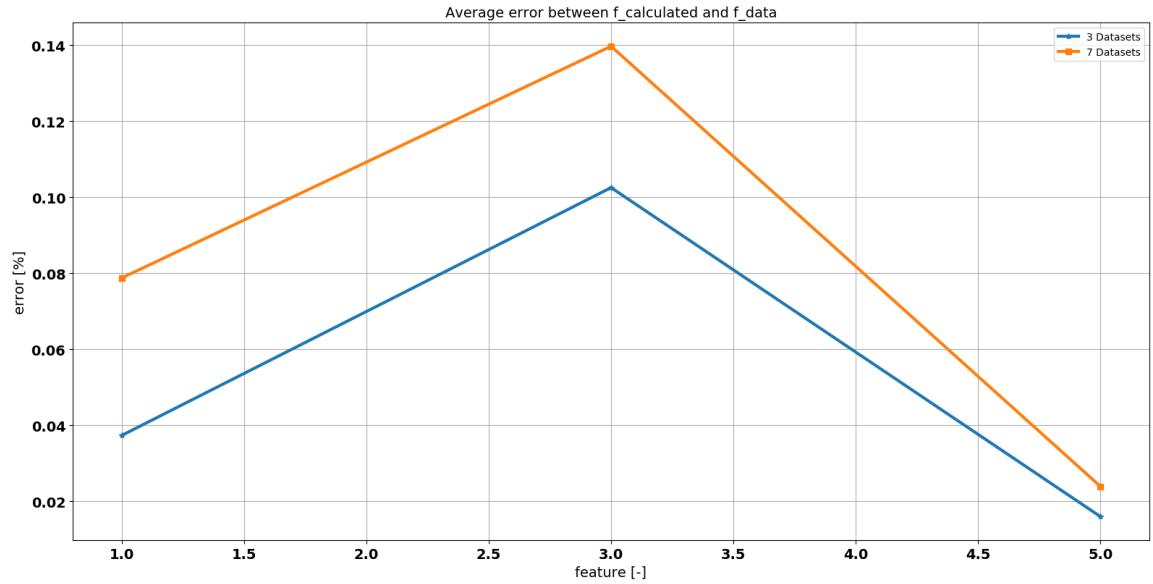


Figure 4.11: This figure shows the averaged, relative error given by $\frac{|1-f_{rel,i}|}{number\ datasets}$ of the matching of the feature values for the three lateral features.(3 datasets: blue, 7 datasets: orange)

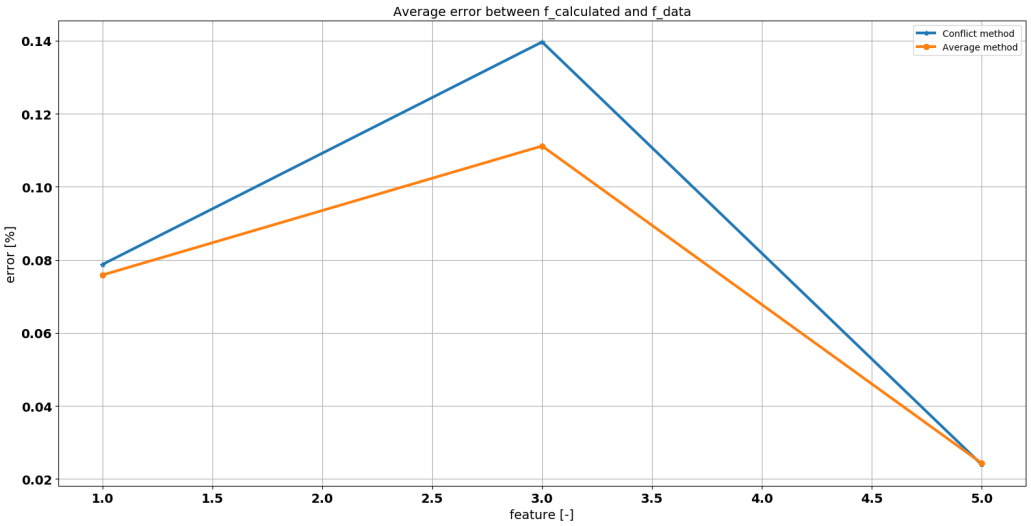


Figure 4.12: This figure shows the averaged, relative error given by $\frac{|1-f_{rel,i}|}{number\ datasets}$ of the matching of the feature values for the three lateral features. (Conflict method: blue, Averaging method: orange)

4.5 Conclusion

In this chapter the implementation of learning of ideal data was analysed. Data is considered ideal because it is produced with the same vehicle model as is used during the learning loop which calls the optimization 4.8. During this chapter first the non-linear bicycle model was discussed and its used parameters are presented in table 4.1. Next a step by step building up of the learning algorithm is shown. It is displayed how the ideal data is generated and validated. Out of this it followed that the choice of number of control points was set on 1000 and the time limit on 30 s. Afterwards the learning results were discussed and the conflict and averaging method were compared. It was concluded that for a lane change maneuver it is only important to learn the lateral weights accurately in order to explain the observed data. Feature matching was also accomplished for the longitudinal features but there was a boundary on the accuracy due to the small longitudinal feature values during a lane change. Because the conflict method gives for the same amount of datasets a slightly larger error on the ability to match feature values and because of the hypothesis that this method will behave worse when used with real driver data, the averaging method is chosen to be further used in this thesis. The estimate of the gradient $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\theta}}$ by $\boldsymbol{F}_{obs} - \boldsymbol{F}(\boldsymbol{r}_{expected})$ was found adequate in order to match the learned and observed feature values.

Chapter 5

Learning from complex vehicle model

Because the non-linear bicycle model makes abstraction of dynamics that are applicable in a real vehicle, a more complex model is introduced to improve the reality factor of the simulations. In order to achieve this the 15 degrees of freedom amesim model as can be seen in Figure 5.1, is provided by Siemens. The parameters of this model are tuned by the company in order to behave similar to a testcar they are currently using. The model has as inputs the amount of throttle, braking and steerwheelangle.

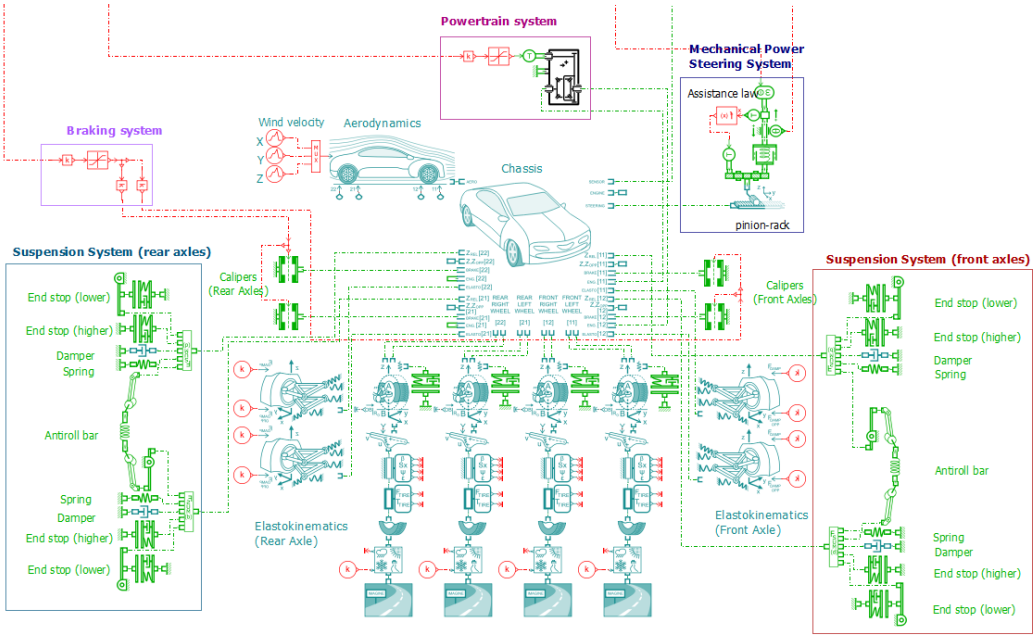


Figure 5.1: The 15 dof amesim model with as inputs the amount of throttle, braking and steerwheelangle.

The Amesim model serves as a black box and no direct dynamic equations are available

to include in 4.8 as was done with the non-linear bicycle model 4.3. Therefore a path is planned with the simpler non-linear bicycle model and afterwards a model predictive control tracking algorithm is applied on the Amesim model in order to follow its reference. The working principles of a MPC is discussed in 2.2. Diagram 5.2 shows the flow of handlings that is done during learning with the Amesim model.

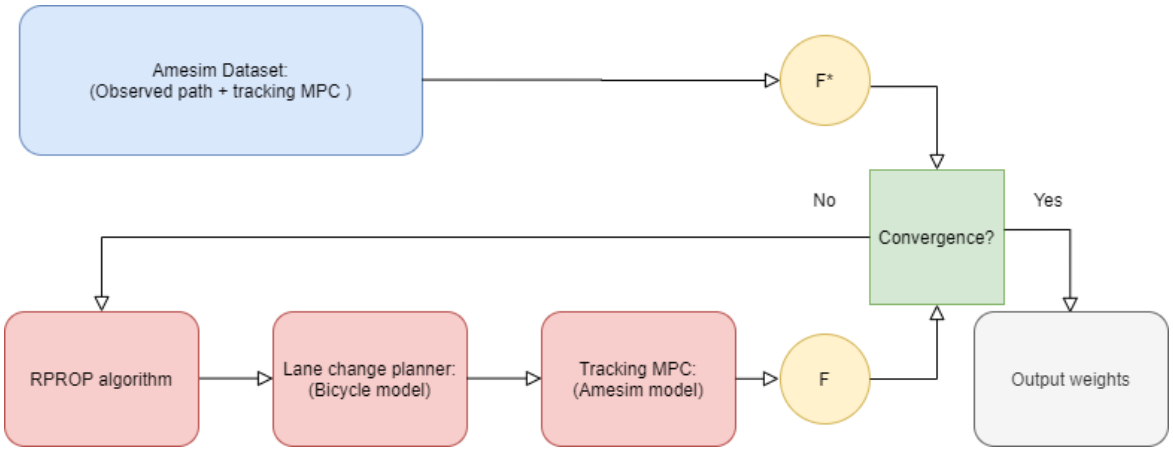


Figure 5.2: The flow of learning with the Amesim model.

As is described in Diagram 5.2 the feature vector $\boldsymbol{F}(\boldsymbol{r})$ is calculated based on the path tracked by the Amesim model and used in the convergence block to estimate the gradient $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\theta}}$. The estimation is calculated by $\boldsymbol{F}_{obs} - \boldsymbol{F}(\boldsymbol{r}_{expected})$. The reason why the planned path has to be tracked is to avoid to include a vehicle mismatch error in the learning process. This that it is otherwise possible that even when the learning algorithm is capable to match the feature vectors $\boldsymbol{F}(\boldsymbol{r})$ and $\boldsymbol{F}^*(\boldsymbol{r})$, coming form two different vehicle models, the learned kinematic signals of the bicycle model will not represent the observations produced by the Amesim model accurate enough. In order to check this, table 5.1 [1] shows the different feature values for a lane change $V_0 : 22.22\ \frac{m}{s},\ L : 3.47\ m$ generated with both the models.

Out of table 5.1 it can be concluded that the feature values, that will serve as the feature values of the observations and were produced using the bicycle and Amesim model, are almost the equivalent. However although the feature values of the observations were the same, not all kinematic signals also give an accurate match as can be seen in Appendix D. This will further be discussed in section 5.1.

This chapter also gives an answer in section 5.2 on the question if the estimate of the gradient $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\theta}}$ by $\boldsymbol{F}_{obs} - \boldsymbol{F}(\boldsymbol{r}_{expected})$ is still sufficient in order to match the learned and observed feature values.

[1] Note that the feature values of the bicycle model are slightly different than the ones in table 4.3. This is because in the previous table the time limit was set on 30 s and here time limit is 25 s. This is done to have a better useable time discretization of 0.025 s to input as reference signal to the Amesim model.

Feature Value	Bicycle model	Amesim model
Nr.1	5.28e-8	3.33e-7
Nr.2	0.37	0.38
Nr.3	1.41e-7	1.20e-4
Nr.4	0.57	0.49
Nr.5	1.89e-6	9.40e-6
Nr.6	30.99	31.05

Table 5.1: This table shows the feature values of a lane change V_0 : 22.22 $\frac{m}{s}$, L : 3.47 m for respectively the Bicycle and Amesim model.

5.1 Tracking MPC

To be able to integrate the Amesim model in the learning process in an adequate manner, good tracking is desirable for the important lateral features that determine the lane change. A good tracking is therefore wanted for: $y(t), a_y(t)$ and $j_y(t)$.

5.1.1 MPC formulation

First the OCP that will be called during the MPC has to be defined. This is done by using the non-linear bicycle model defined with 10 states as seen in 4.2, inside the OCP formulation given by 5.1. 5.1 uses as objective an error function with respect to the reference with will assure the following behaviour. The control horizon of the MPC is N_{MPC} and equal to 50 points which means a control horizon of 1.25 s because the reference is sampled with T_{pl} equal to 0.025 s. The parameters of the bicycle model stay the same as the ones listed in 4.1. In order to formulate the gap closing constraint, which connects the previous states to the next in the chosen multiple shooting formulation, Runge-Kutta integration is embedded in function I. The first time that the OCP is solved and the current state is not yet outputted by the virtual sensors of the Amesim model, the initial states are set equal to the first reference point. The path constraints can be seen in 5.2. The path constraints are at first sight not necessary but contribute by decreasing the feasible solution space for the solver. It is checked that these constraints are not binding, which means that the found solution is reachable even if the constraints were removed.

$$
\begin{aligned}
\min_{\boldsymbol{X}(.),\boldsymbol{U}(.)} \quad & E(\boldsymbol{X}(.),\boldsymbol{U}(.)) \\
\text{s.t.} \quad & \boldsymbol{X}^{k+1} = I(\boldsymbol{X}^k,\boldsymbol{U}^k) \quad k = [0,\cdots,N_{MPC}-1] \\
& \boldsymbol{X}^0 = \boldsymbol{X}_{current} \\
& \boldsymbol{F}(\boldsymbol{X}^k) \geq 0 \qquad\qquad k = [0,\cdots,N_{MPC}] \\
& \boldsymbol{X}^k \in \mathbb{R}^{10\times 1} \qquad\qquad k = [0,\cdots,N_{MPC}] \\
& \boldsymbol{U}^k \in \mathbb{R}^{2\times 1} \qquad\qquad k = [0,\cdots,N_{MPC}-1] \\
& N_{MPC} \in \mathbb{N}
\end{aligned}
\tag{5.1}
$$

$$\boldsymbol{F} = \begin{Bmatrix} -\frac{Width\ Lane}{2} \leq y^k \leq \frac{3 \cdot Width\ Lane}{2}, & k = [0, \cdots, N_{MPC}] \\ 0 \leq x^k, & k = [0, \cdots, N_{MPC}] \\ -\frac{\pi \cdot 5}{180} \leq \psi^k \leq \frac{\pi \cdot 5}{180}, & k = [0, \cdots, N_{MPC}] \\ v_{start} - 1 \leq v_x^k \leq v_{start} + 1, & k = [0, \cdots, N_{MPC}] \end{Bmatrix} \tag{5.2}$$

The error function that serves as the objective of the OCP is given by the following equation. The weights that gave the best tracking results are shown in table 5.2.

$$\begin{aligned} objective = {} & W_1(\boldsymbol{x}[2:end] - \boldsymbol{ref}_x)^T(\boldsymbol{x}[2:end] - \boldsymbol{ref}_x) + W_2(\boldsymbol{y}[2:end] - \boldsymbol{ref}_y)^T(\boldsymbol{y}[2:end] - \boldsymbol{ref}_y) \\ & + W_3(\boldsymbol{v}_x[2:end] - \boldsymbol{ref}_{v_x})^T(\boldsymbol{v}_x[2:end] - \boldsymbol{ref}_{v_x}) + W_4(\boldsymbol{v}_y[2:end] - \boldsymbol{ref}_{v_y})^T(\boldsymbol{v}_y[2:end] - \boldsymbol{ref}_{v_y}) \\ & + W_5(\boldsymbol{\psi}[2:end] - \boldsymbol{ref}_{\psi})^T(\boldsymbol{\psi}[2:end] - \boldsymbol{ref}_{\psi}) + W_6(\dot{\boldsymbol{\psi}}[2:end] - \boldsymbol{ref}_{\dot{\psi}})^T(\dot{\boldsymbol{\psi}}[2:end] - \boldsymbol{ref}_{\dot{\psi}}) \\ & + W_7 \dot{\boldsymbol{t}}_r^T \dot{\boldsymbol{t}}_r + W_8 \dot{\boldsymbol{\delta}}_s^T \dot{\boldsymbol{\delta}}_s + W_9 \dot{\boldsymbol{a}}_x^T \dot{\boldsymbol{a}}_x \end{aligned}$$

$$\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{v}_x, \boldsymbol{v}_y, \boldsymbol{\psi}, \dot{\boldsymbol{\psi}}, \boldsymbol{a}_x \in \mathbb{R}^{(N_{MPC}+1)\times 1} \qquad \boldsymbol{ref}_i, \dot{\boldsymbol{t}}_r, \dot{\boldsymbol{\delta}}_s \in \mathbb{R}^{N_{MPC}\times 1}$$

Weight	Value
W1	10
W2	10
W3	30
W4	1.0
W5	100
W6	1.0
W7	5.0
W8	0.01
W9	0.01

Table 5.2: Overview of the weights used in the objective of 5.1.

The ultimate weights displayed were attained by trail and error but there is an intuitive explanation on why these states were used and which order of magnitude the weights were given.
It was first tried to focus on path tracking in order to see how the other states would differ when the Amesim model drove exactly the same path as the non-linear vehicle and therefore $x(t), y(t)$ and $\psi(t)$ were included. They define the orientation of the vehicle. Nervous input behaviour was noticed and in order to smooth the input, they were given a small weight. Only a small weight is given to assure good tracking behaviour. It was observed that this strategy resulted in a good tracking of the important signals for the learning algorithm: $y(t), a_y(t)$ and $j_y(t)$. As will be explained in 5.1.2 the Amesim model made a deceleration at the begin. In order to give the model time to stabilize before the start of the maneuver, a straight driving part of 15 s and 2.5 s was respectively added before and after the reference

lane change maneuver outputted by 4.8. To faster remove the oscillation of the longitudinal signals, v_x and a_x were included in the error function.

The initial guess when the OCP is solved for the first time, is only needed for v_x. This is because otherwise an invalid value would emerges in the calculation of the slip angle according to equation 4.6. When the OCP is defined and implemented in CasADi, the opti environment is saved as a function that can be called during the MPC. In order to improve the solving time of 5.1, the solver is switched to a SQP method using an active-set QP solver instead of IPOPT as was used in 4.8. "Interior point methods (like IPOPT) are generally robust at finding a minimizer, but the barrier parameter will make the algorithm walk away from a perfect initial guess, only to come back after a while." [9] A hot starting was implemented by feeding the solution of a previous MPC iteration as initial guess for the optimization variables and the lambda multipliers which made it logical to switch the solver and solving method. Together with the straight driving parts, a simulation of 40 s is performed and one iteration of the MPC where one OCP is solved, takes around 0.15 s. The main time consumed to calculate the solution is mainly the loading done before the actual running of the MPC.

With the OCP defined, it can be included in a MPC formulation. Every MPC iteration uses the current state of the Amesim model and a part of the reference that pertains to the current control horizon. The Amesim model is evaluated at a sampling rate of $T_s = 0.01\ s$ and the optimization 5.1 is called at a rate of $T_{MPC} = 0.1\ s$. The reference has a sampling rate of $T_{pl} = 0.025\ s$. In order to connect the tracking MPC with the Amesim model using different sampling times, the Simulink model of Figure 5.3 is build.

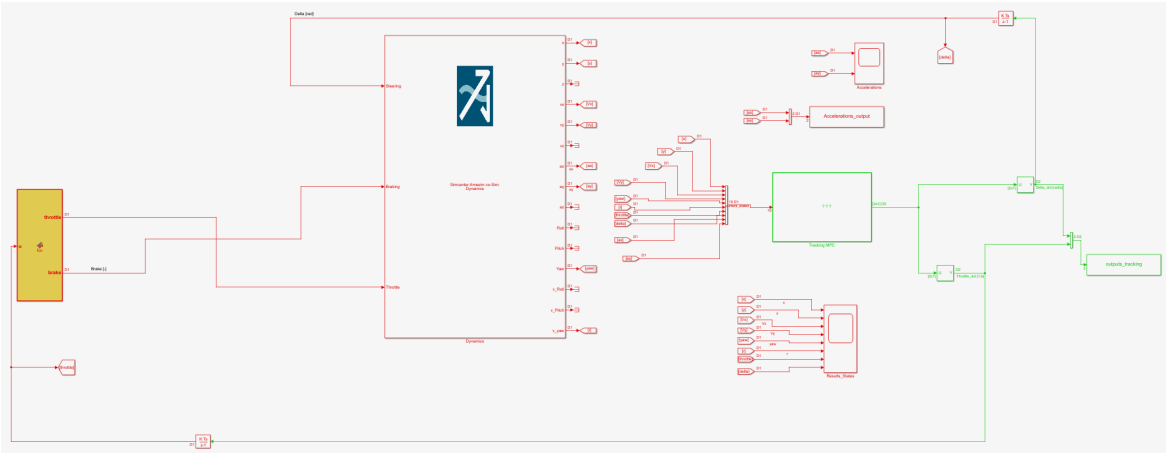


Figure 5.3: This figure shows hows the tracking MPC is connected with the Amesim model using different sampling times. (red: T_s : 0.01 s and green: T_{MPC} : 0.1 s)

The three inputs to the Amesim model (largest block) in Figure 5.3 are δ_s, t_r and

braking but the control outputs of 5.1 give $\dot{t}_r$ and $\dot{\delta}_s$. Therefore 'Forward Euler integration' blocks are introduced using for 0.1 s the control outputs of 5.1 in order to calculate every 0.01 s the inputs for the Amesim model. The throttle of the bicycle model can theoretically become positive and negative. These two states were separated and feeded to the Amesim model as throttling or braking.

5.1.2 Tracking results

In this section the results of the tracking MPC are presented. For a full overview of the results, reference is made to Appendix D. Figure 5.4 shows that the tracking of the path (blue) of the reference (red) is done with an accuracy of 10^{-3}.

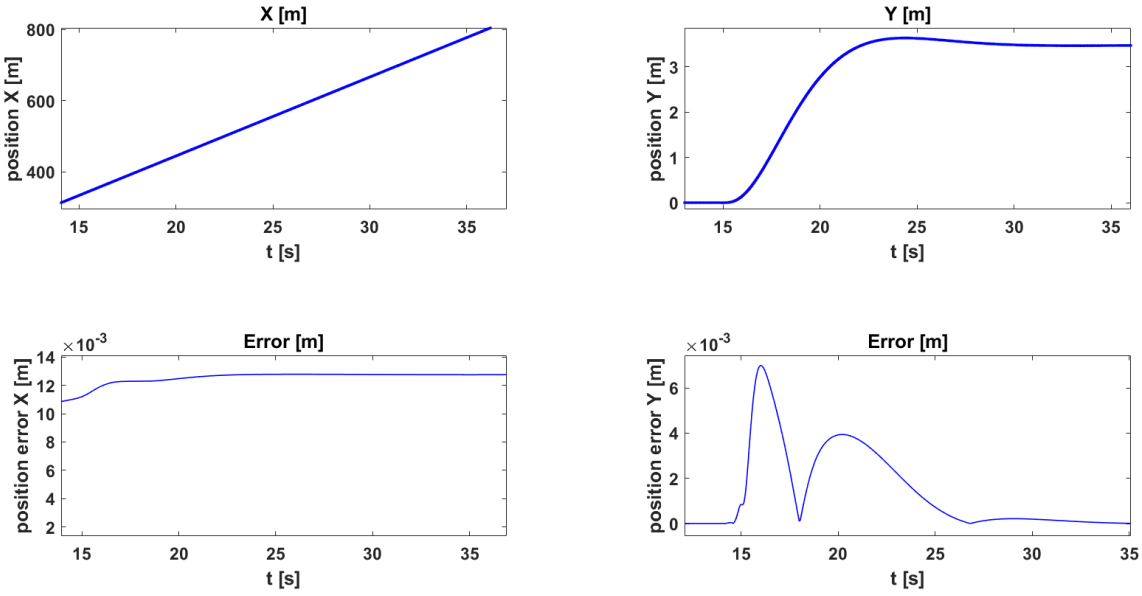


Figure 5.4: This figure shows the tracking result of the reference (red) by the Amesim model (blue).

During the analysing of the results it was concluded that the important signals were followed sufficiently well to be able to use the tracking MPC in the learning configuration of Figure 5.2. However there were also differences seen between the reference and the followed trajectory. This justifies the reasoning that not all states have to be included in the error function of 5.1, because tracking all states accurate will give worse results due to inherent vehicle model mismatch. The first difference is that the Amesim model needs a bigger steerwheelangle than was predicted by the reference. Also the amount of throttle needed to drive forward at a constant speed is slightly different because of a more complete way of aerodynamic resistance modelled by the Amesim model in comparison to equation 4.4 used by the bicycle model.
Another observation is that the lateral jerk and first derivative of the steerwheelangle display less high peaks than the reference. Also when there is closely zoomed in on

the longitudinal jerk around 15 s, which coincides with the start of the lane change and 22 s which is more at the end, nervous behaviour of the longitudinal jerk is witnessed. Around 15 s a bump in the reference of the throttle can be seen because that is the point where straight driving at constant speeds ends and the reference is from there on produced by a solution of 4.8. The influence on v_x and a_x is however marginal. Further it is worth to note that the jerks don't come directly from the Amesim model but the signal is post-processed by numerical differentiation.

To be able to generate adequately smooth jerk signals, it was necessary to include smooth inputs of throttle and steerwheelangle. Figure 5.5 shows oscillating behaviour of the lateral jerk when not the first derivative of throttle and steerwheelangle are used as control outputs of 5.1. That the jerk signal is dependent on the first derivative of throttle and steeringwheelangle is a logical result, because also in the analytical jerk equations of the non-linear bicycle model (Appendix A) these variables were embedded.[2]

5.2 Learning results with the Amesim model

Diagram 5.2 shows in blue the observed path that is generated out of 4.8 with known weights whereafter the path based on a bicycle model is tracked by a tracking MPC to get the kinematic signals of the 15 dof Amesim model. From this $\boldsymbol{F}^*(\boldsymbol{r})$ is calculated which stays constant during whole the learning process.

The learning loop shown in red in diagram 5.2 starts at the lane change planner, where 4.8 is called with an initial guess of the weights an all-one vector. After the planned path is outputted, tracked[3] and the associated feature vector $\boldsymbol{F}(\boldsymbol{r})$ is calculated, it is checked if the two feature vectors match accurate enough in the convergence block. If this is not the case, the difference of the features is taken as an estimate for $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\theta}}$ and used in the RPROP algorithm in order to generate a new planned path by calling 4.8.

Three simulations are conducted with respectively one, two and three datasets. Table 5.3 shows which longitudinal speed is driven at the begin of the lane change (V_0) and the desired lateral displacement (L). As is noted before in Chapter 4, if only the start speed is varied, the lateral feature values stay the same and the longitudinal features stay very small. This means that these datasets are seen as almost the same when learning. When the lateral desired displacement is varied, this gives a clear difference in the lateral feature values. An overview of the different features of the individual datasets is given in table 5.4. [4]

[2] For the results of Figure 5.5, no straight driving reference path is added. The lane change directly starts at $time = 0.0$.

[3] The first 10 s of the tracked Amesim signal is removed. The remaining maneuver consists of 5 s straight driving before doing the actual lane change.

[4] In the calculation of the feature values the 5 s straight driving before the lane change is included. This contributes to a higher value for feature number six.

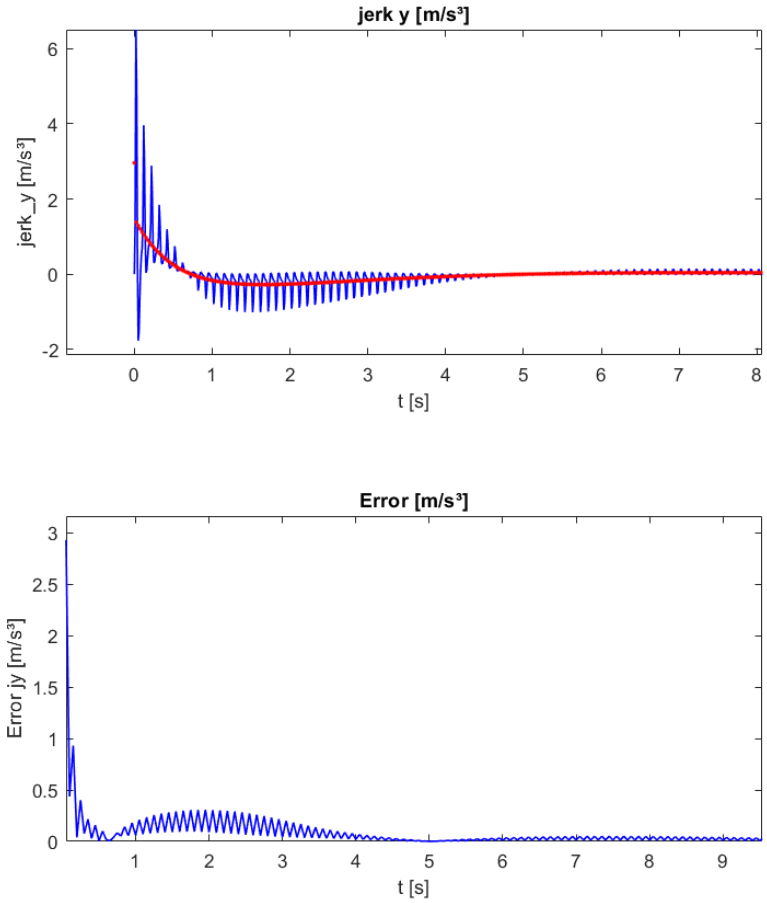


Figure 5.5: This figure shows the obtained oscillating signals for the jerk when a piecewise signal of δ_s and t_r is applied.

1 Dataset	2 Datasets	3 Datasets
$V0: 22.22 - L: 3.47$	$V0: 22.22 - L: 3.47$	$V0: 22.22 - L: 3.47$
	$V0: 25.00 - L: 6.94$	$V0: 25.00 - L: 6.94$
		$V0: 27.78 - L: 3.47$

Table 5.3: This table shows which lane changes were used during the three different simulations.

The weights learned for the three different simulations is presented in table 5.5 and $\boldsymbol{f}_{rel}$ at convergence is given in table 5.6. $\boldsymbol{f}_{rel}$ displays the matching of the averaged learned feature values with the observed one. Convergence for the three simulations is reached after 28 iterations. The convergence criteria used is matching the lateral features (2,4,6) with a tolerance of 10^{-3}.

Feature Value	V022.22 - L3.47	V025.00 - L6.94	V027.78 - L3.47
Nr.1	5.18e-07	5.97e-06	4.01e-07
Nr.2	0.38	1.51	0.38
Nr.3	12.29e-05	7.23e-05	5.37e-06
Nr.4	0.51	2.09	0.52
Nr.5	9.50e-06	7.08e-05	3.71e-05
Nr.6	91.25	364.97	91.25

Table 5.4: This table shows the different features for the individual datasets used during simulation.

Weight	1 Dataset	2 Datasets	3 Datasets
Nr.1	33.4230	5.2294	3.9036
Nr.2	5.0000	5.0000	5.0000
Nr.3	2.8662e-06	1.0636	1.9803
Nr.4	6.0018	6.0018	6.0018
Nr.5	4.4445	1.2705	1.1869
Nr.6	2.0011	2.0011	2.0011

Table 5.5: This table shows the weights learned for the different simulations that start from an all one vector and uses as chosen weights $\left[\,4.0,\ 5.0,\ 1.0,\ 6.0,\ 1.0,\ 2.0\,\right]$.

$\boldsymbol{f}_{rel}$	1 Dataset	2 Datasets	3 Datasets
Nr.1	1.1315	1.0097	0.9643
Nr.2	1.0000	1.0002	1.0003
Nr.3	0.2865	0.4496	0.4867
Nr.4	1.0000	1.0002	1.0002
Nr.5	1.0718	1.0030	0.9937
Nr.6	1.0000	1.0000	1.0000

Table 5.6: This table shows the final values of $\boldsymbol{f}_{rel}$.

Out of table 5.5 and 5.6 it is concluded that the lateral chosen weights are found back and an accurate match of the lateral features is achieved. It is observed that also the longitudinal weights will come closer to their chosen value and give a better match for the longitudinal feature values, when more observed lane changes are included in the learning.

Figure 5.6 presents the observed paths, the initial solution when the weight vector is chosen equal to an all-one vector and the learned solution when 2 datasets are used. Figure 5.7 shows the error made between the observed and learned path which is of order of size of 10^{-4}. All the resulting kinematic signals are displayed in Appendix E.

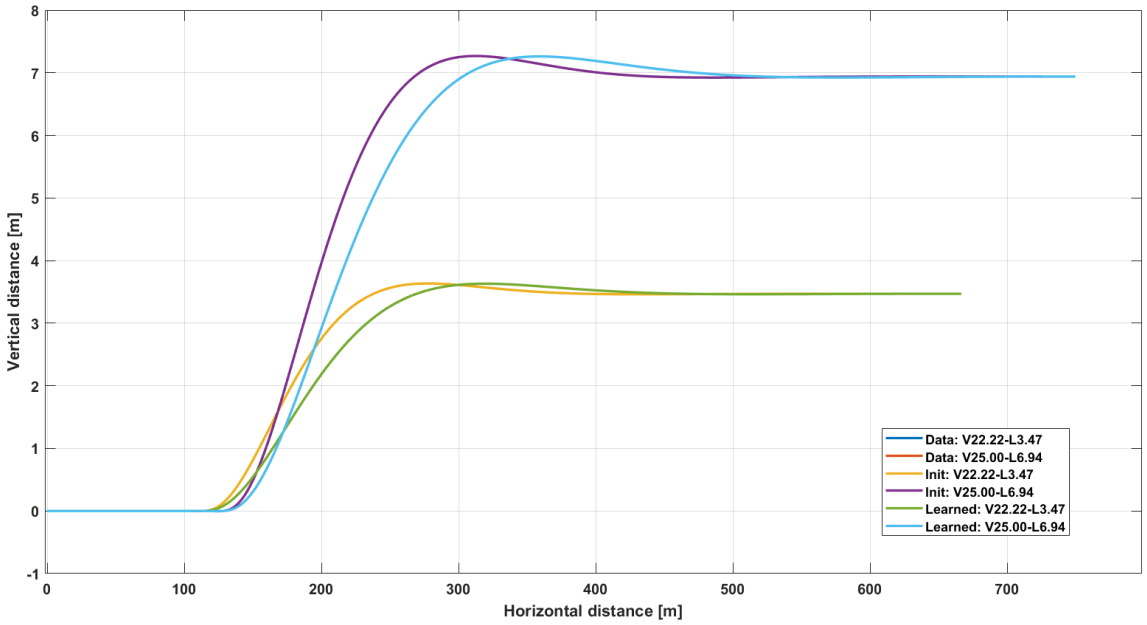


Figure 5.6: This Figure shows the observed paths, the initial solution retrieved with as weights an all one vector and the learned solution.

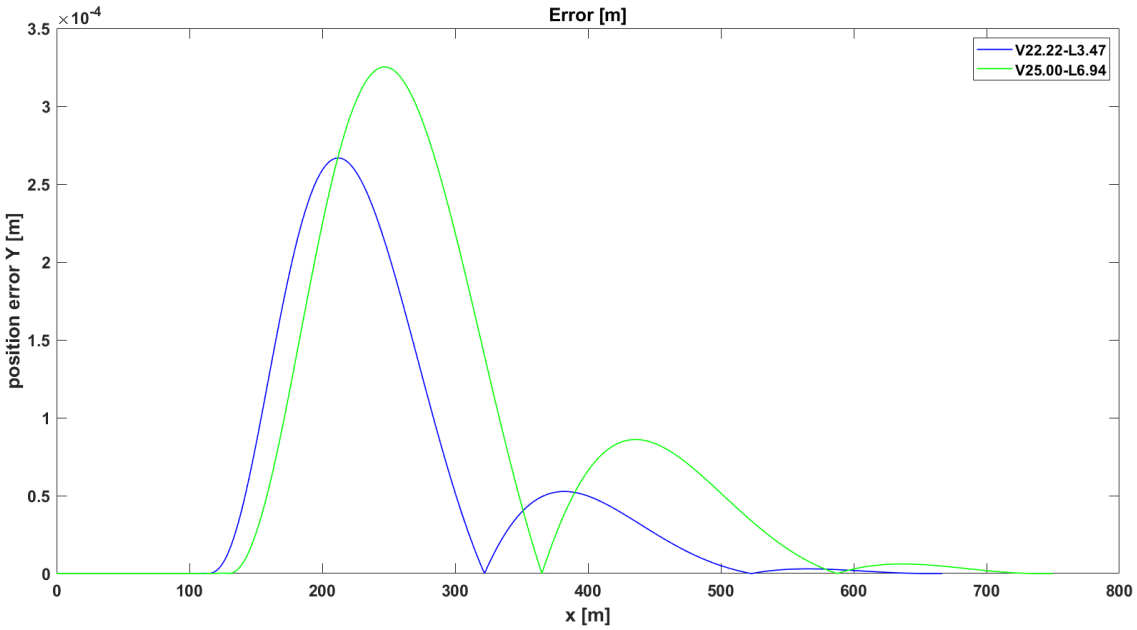


Figure 5.7: This Figure shows the error made between the observed and learned paths.

Figure 5.8 shows the convergence of $\boldsymbol{f}_{rel}$ towards one during the learning iterations. Except for the longitudinal jerk (feature 3) all the feature values converge towards the observed ones.

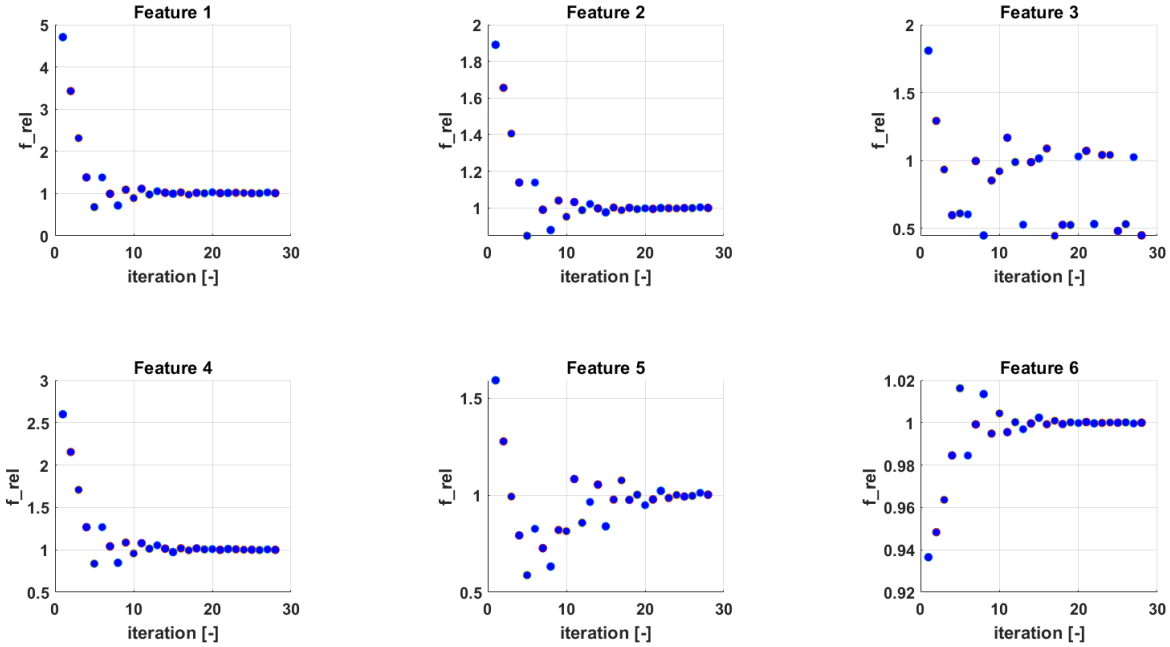


Figure 5.8: This Figure shows $\boldsymbol{f}_{rel}$ during the learning iterations.

From the retrieved $\boldsymbol{f}_{rel}$ of table 5.6 and in Figure 5.8, the feature value for the longitudinal jerk (Nr. 3), jumps out for its bad feature matching behaviour. (0.4867 when three datasets are used) A reason for this is as discussed in 5.1.2, bad quality of the jerk data which contributes to a worsen learning. The jerk during the lane change can be seen in Figure 5.9. At time zero the longitudinal behaviour is still not fully stabilized and at second 5 the lane change starts.

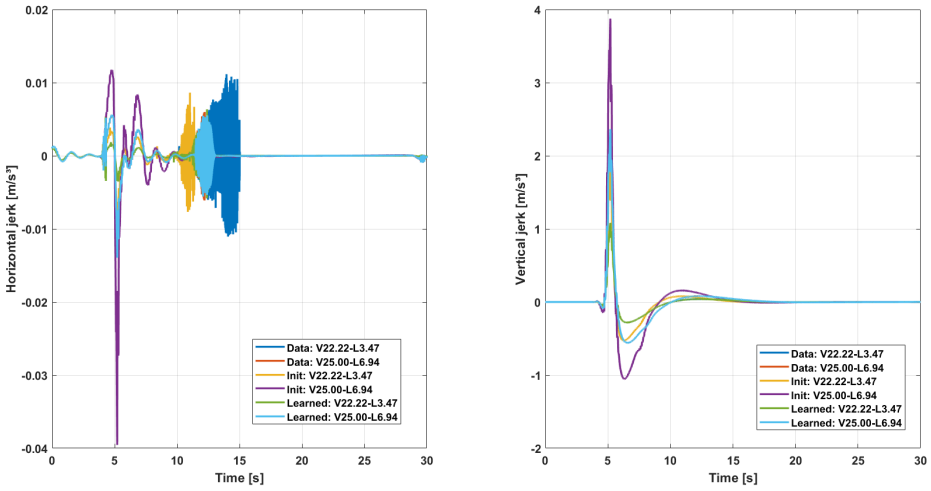


Figure 5.9: This Figure shows the observed jerk signals, the initial solution retrieved with as weights an all one vector and the learned solution.

5.3 Conclusion

In this chapter the learning of driver specific comfort parameters with a realistic 15 dof Amesim model was discussed. The flow of the different steps of the learning process and how the Amesim model would be embedded, was looked into. The question posed was: "Is it possible to find back the underlying comfort weights when the gradient $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\theta}}$ is estimated by $\boldsymbol{F}_{obs} - \boldsymbol{F}(\boldsymbol{r}_{expected})$".

First the formulation of the used tracking MPC was presented and its tracking performance evaluated. The results are that tracking of the planned path was possible to an accuracy of 10^{-3} and also the other important features a_y and j_y could be tracked to a satisfying accuracy as can be seen in Appendix D. However δ_s, t_r in steady-state and the peaks of $\dot{\delta}_s$ and j_y, gave a difference with the reference when the path generated by the non-linear bicycle model is being accurate tracked. Next a notion was made about the bad quality of the obtained longitudinal jerk signal. Eventually it was stated that only a smooth signal for the lateral jerk can be retrieved when the input signals are at least of first order.

When the tracking MPC was validated the learning results with the Amesim model were discussed. Three simulations were conducted with a different amount of observed lane changes. Out of the results it could be concluded that every time there is a good convergence found for $\boldsymbol{f}_{rel}$ and the chosen lateral weights are found back. When more observed lane changes are included during the learning also the results for the longitudinal weights improve.
Only the convergence of the longitudinal jerk stayed deficient, which is due to bad quality of the longitudinal jerk data that was used to learn the weights of.

Chapter 6

Conclusion

The final chapter contains the overall conclusion. It also contains suggestions for future work and industrial applications.
Application: say something about hierarchical control. The comfort controller can work as an inferior controller of the safety of the vehicle.

Appendices

Appendix A

Jerk equations of the non-linear bicycle model

In this section the jerk equations that were derived from the non-linear bicycle model and used in the analytical learning algorithm are displayed.

A.1 Equations

The equations of jerk in function of the non-linear vehicle states is the derivate of the total acceleration of the centre of gravity. The undermentioned equations were validated with 'MUPAD', a symbolic toolbox in Matlab.

$$\begin{aligned} j_{x,t} = \frac{1}{M} \cdot (&-sin\delta t_r c\dot{\delta} + cos\delta \dot{t_r} c + cos\delta 2K_{y,f} arctan(\frac{v_y + \omega a}{v_x})\dot{\delta} \\ &+ sin\delta 2K_{y,f} \frac{v_x a_{t,y} + v_x \dot{\omega} a - a_{t,x} v_y - a_{t,x} \omega a}{v_x^2 + v_y^2 + 2v_y \omega a + \omega^2 a^2} - cos\delta 2K_{y,f} \delta\dot{\delta} \\ &- sin\delta 2K_{y,f}\dot{\delta} + \dot{t_r} c - c_{r1} 2 v_x a_{t,x}) \end{aligned}$$

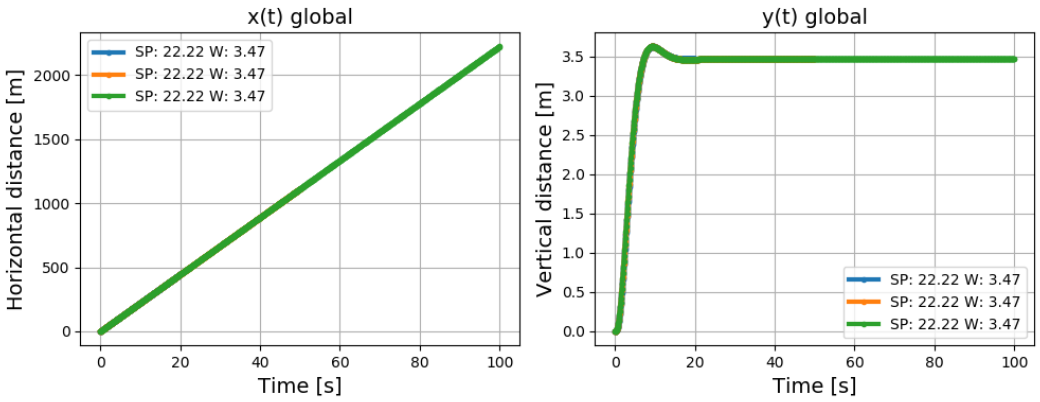
$$\begin{aligned} j_{y,t} = \frac{1}{M} \cdot (&cos\delta t_r c\dot{\delta} + sin\delta \dot{t_r} c + sin\delta 2K_{y,f} arctan(\frac{v_y + \omega a}{v_x})\dot{\delta} \\ &- cos\delta 2K_{y,f} \frac{v_x a_{t,y} + v_x \dot{\omega} a - a_{t,x} v_y - a_{t,x} \omega a}{v_x^2 + v_y^2 + 2v_y \omega a + \omega^2 a^2} - sin\delta 2K_{y,f} \delta\dot{\delta} \\ &+ cos\delta 2K_{y,f}\dot{\delta} - 2K_{y,r} \frac{v_x a_{t,y} - v_x \dot{\omega} b - a_{t,x} v_y + a_{t,x} \omega b}{v_x^2 + v_y^2 - 2v_y \omega b + \omega^2 a^2}) \end{aligned}$$

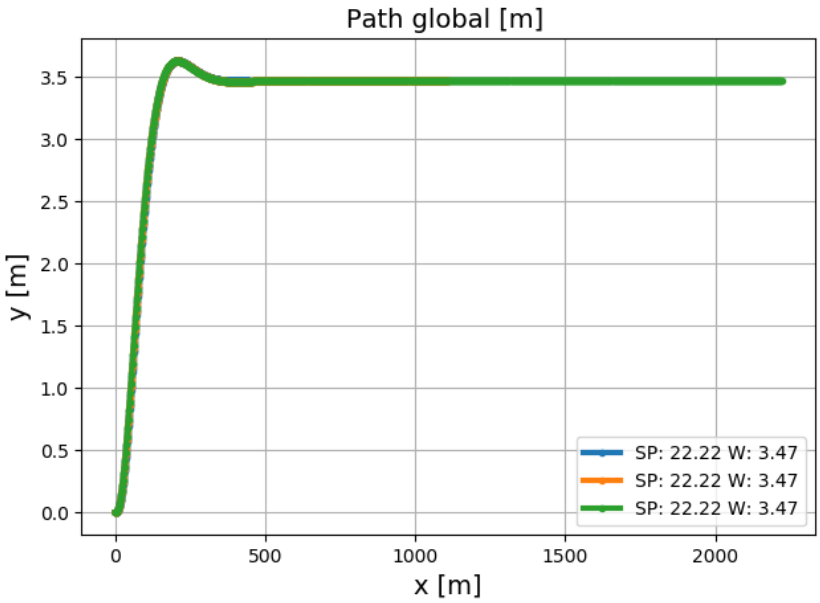
Appendix B

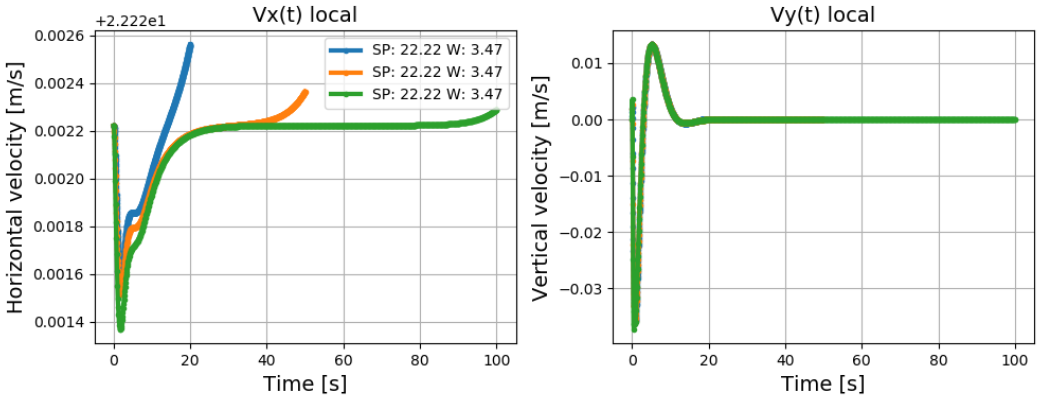
Results of the ideal data validation

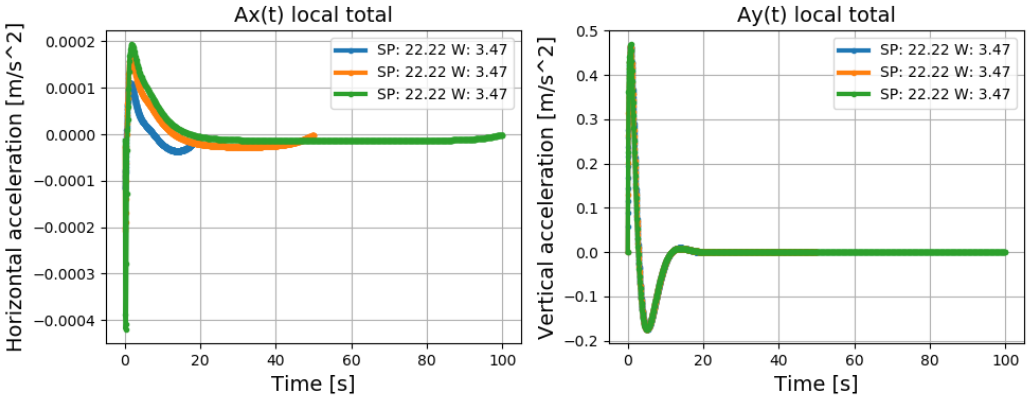
In this appendix all the kinematic signals of the bicycle model that were checked in order to validate the ideal data, are presented. The lane change used for validation has as initial speed of $22.22\frac{m}{s}$ and uses a desired lateral distance of $3.47\ m$ First the different results of varying the time limit are shown and next come the results with regard to varying the amount of control points N.

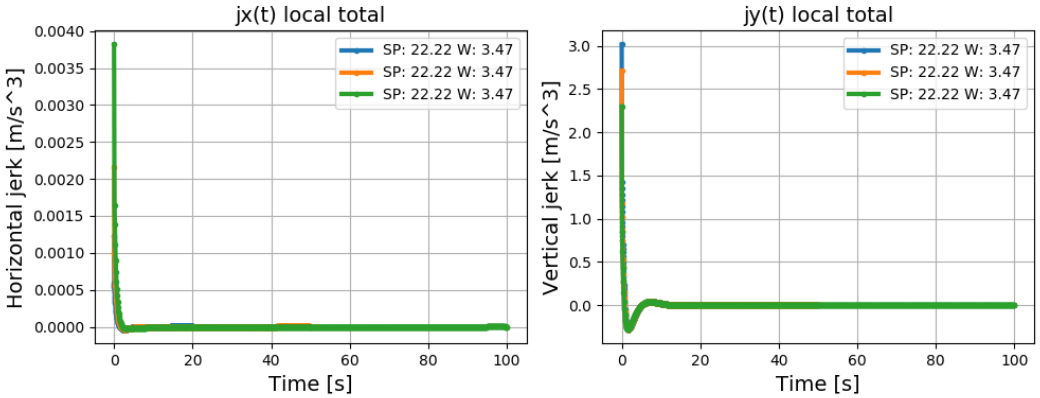
B.1 Time limit



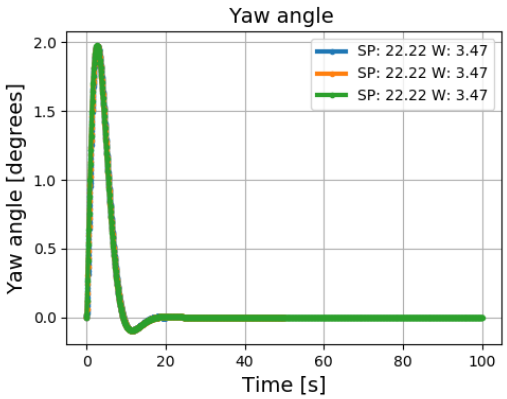

Path global [m]
y [m]
x [m]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47

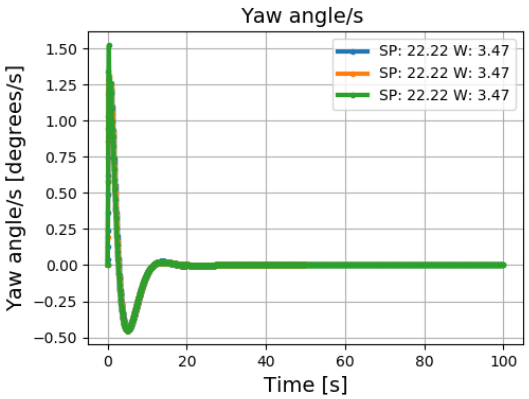

Vx(t) local
+2.222e1
Horizontal velocity [m/s]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Vy(t) local
Vertical velocity [m/s]
Time [s]

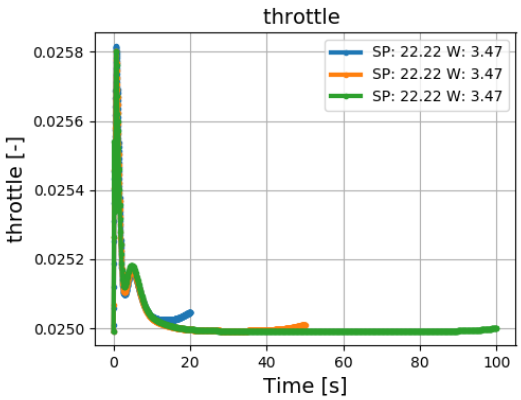

Ax(t) local total
Horizontal acceleration [m/s^2]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Ay(t) local total
Vertical acceleration [m/s^2]
Time [s]

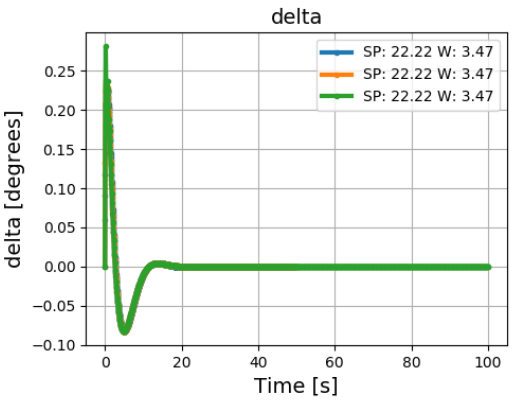

jx(t) local total
Horizontal jerk [m/s^3]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47

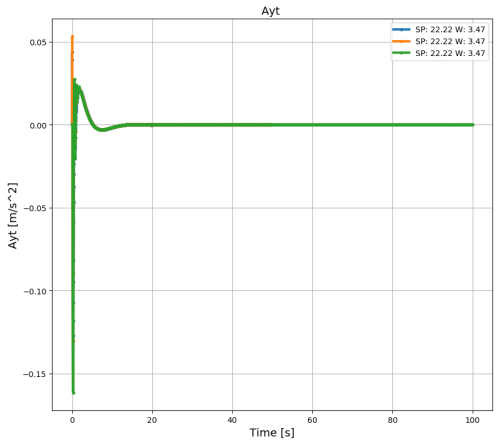
jy(t) local total
Vertical jerk [m/s^3]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47

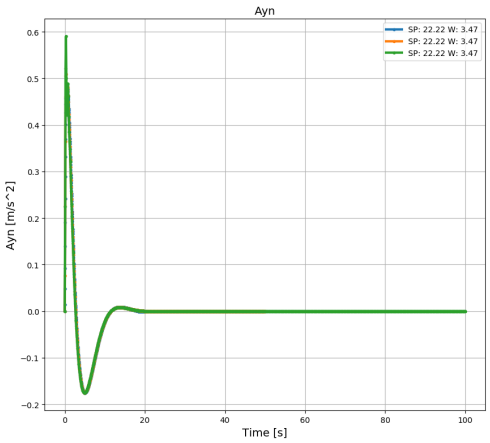

Yaw angle
Yaw angle [degrees]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47

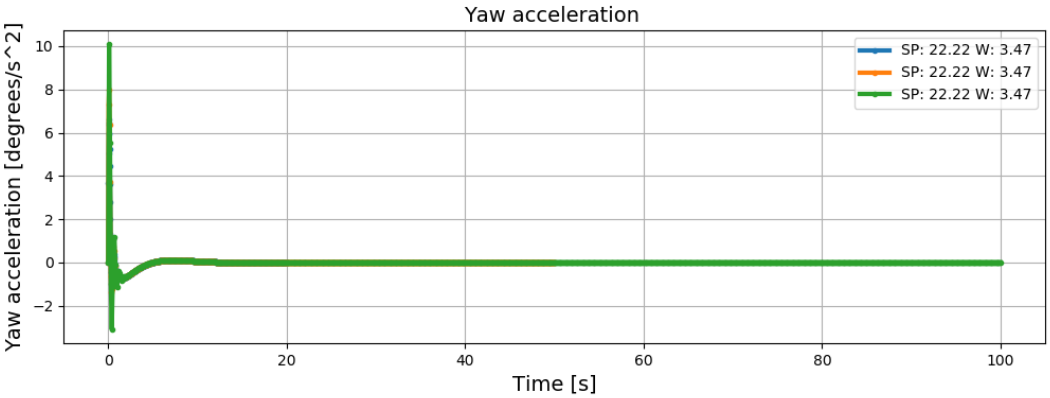

Yaw angle/s
Yaw angle/s [degrees/s]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47

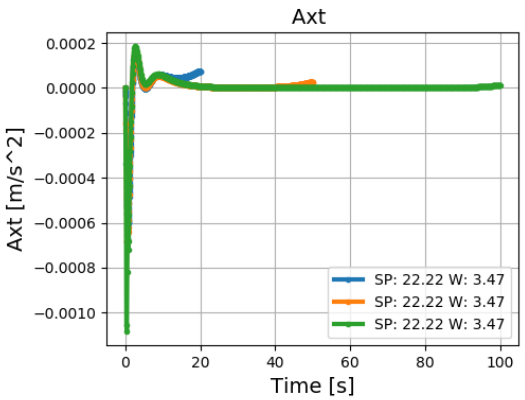

throttle
throttle [-]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47

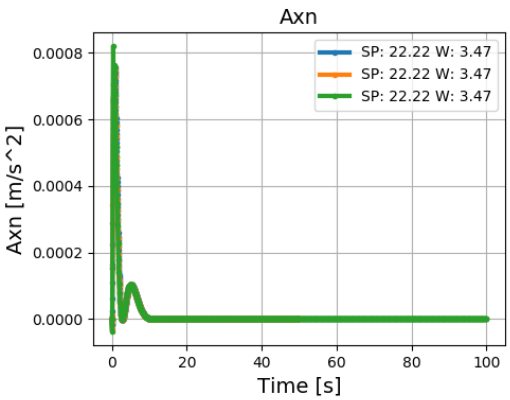

delta
delta [degrees]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47

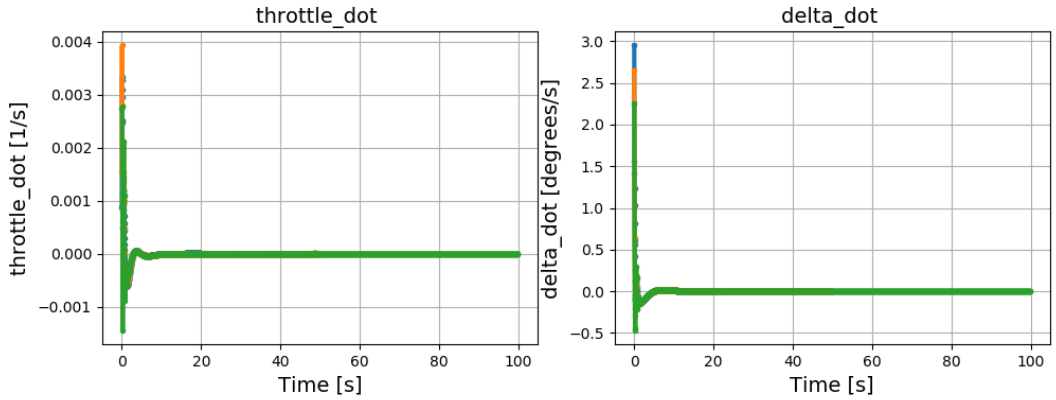

Ayt
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Ayt [m/s^2]
Time [s]


Ayn
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Ayn [m/s^2]
Time [s]

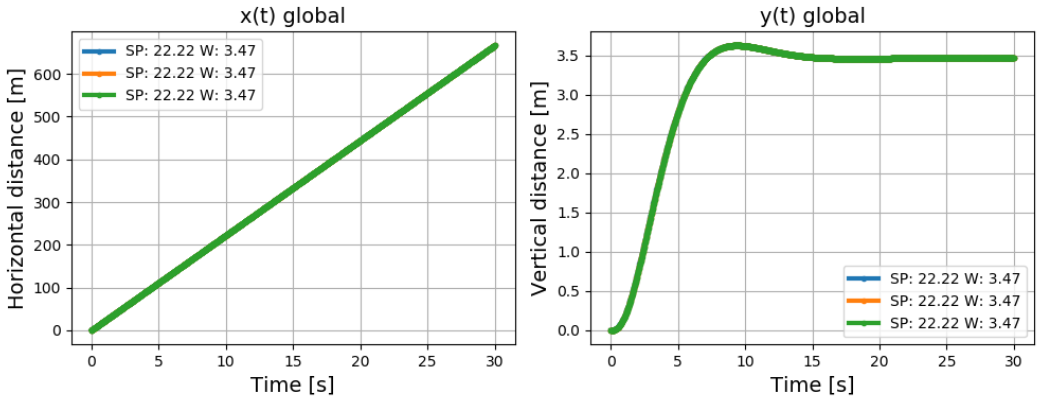

Yaw acceleration
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Yaw acceleration [degrees/s^2]
Time [s]

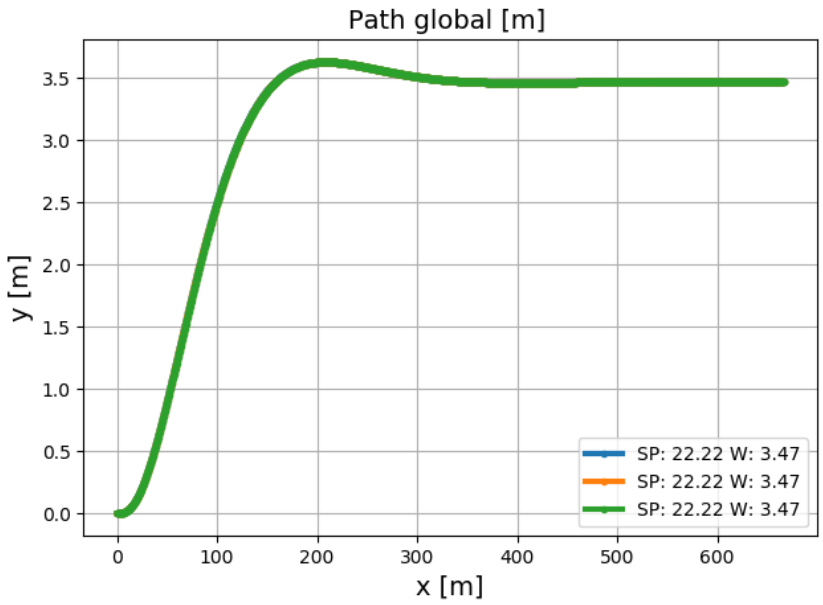

Axt
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Axt [m/s^2]
Time [s]

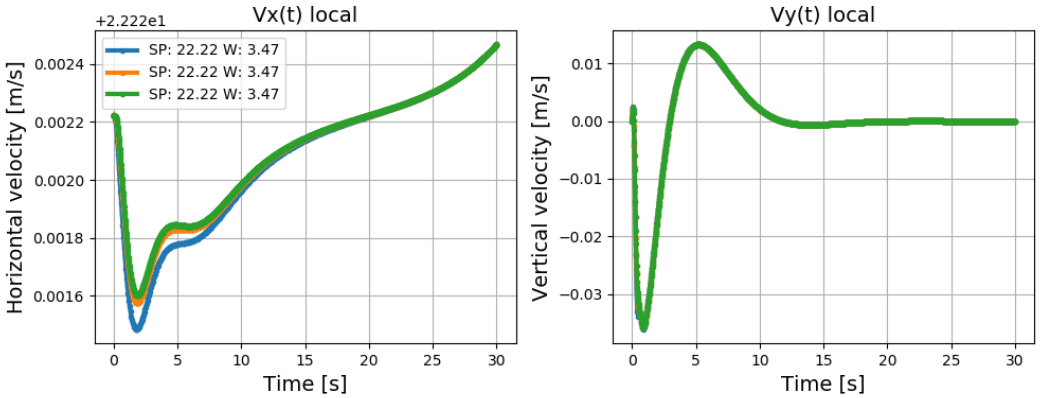

Axn
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Axn [m/s^2]
Time [s]


throttle_dot
throttle_dot [1/s]
0.004
0.003
0.002
0.001
0.000
−0.001
0
20
40
60
80
100
Time [s]
delta_dot
delta_dot [degrees/s]
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
0
20
40
60
80
100
Time [s]

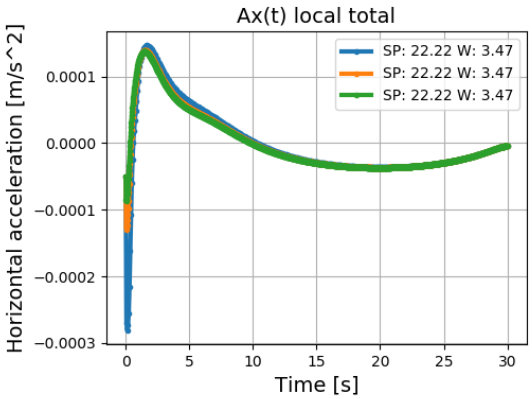
B.2 Amount of optimization points

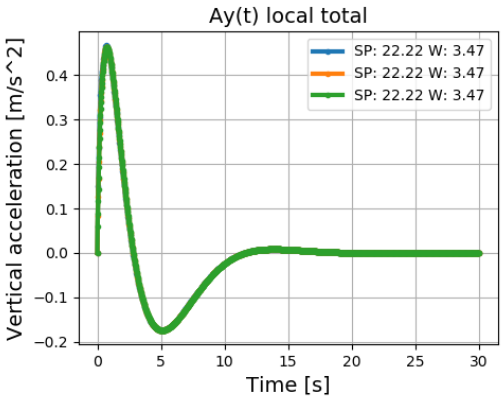


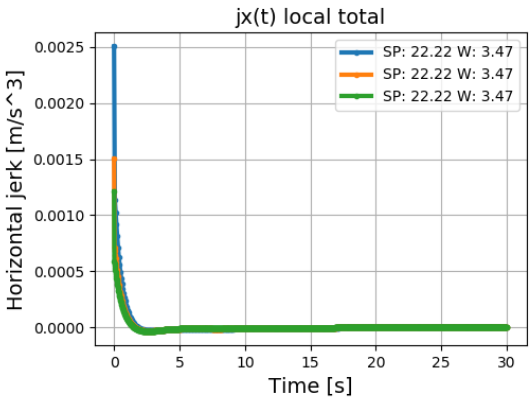


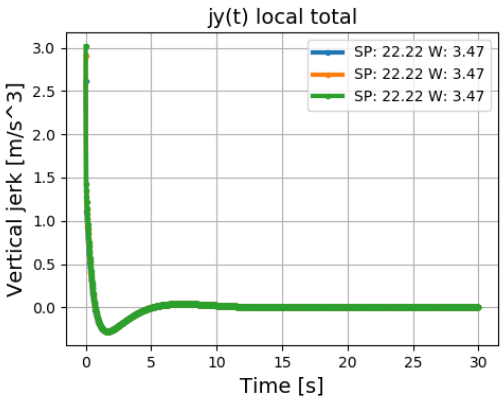

Vx(t) local
+2.222e1
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Horizontal velocity [m/s]
Time [s]

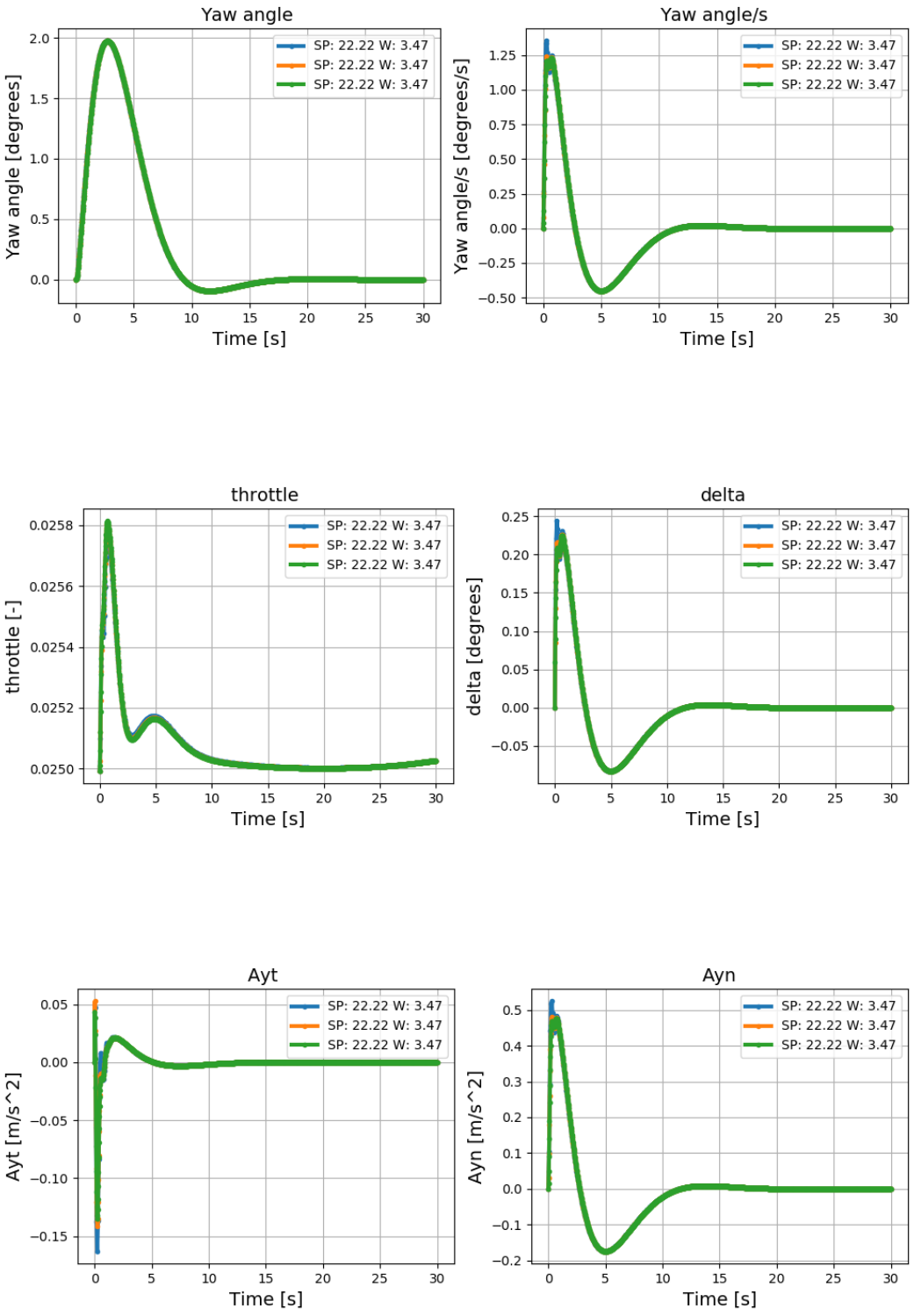
Vy(t) local
Vertical velocity [m/s]
Time [s]

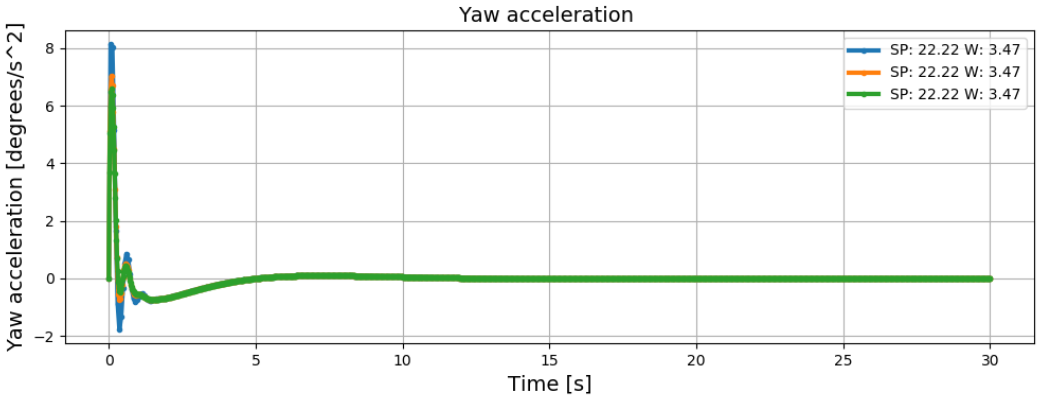

Ax(t) local total
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Horizontal acceleration [m/s^2]
Time [s]

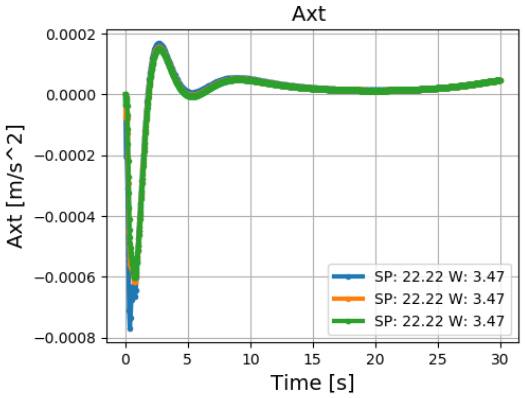

Ay(t) local total
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Vertical acceleration [m/s^2]
Time [s]

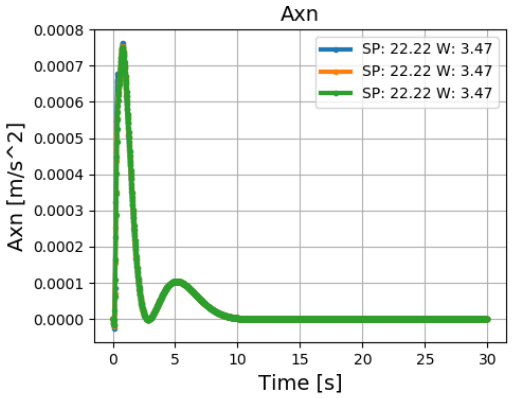

jx(t) local total
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Horizontal jerk [m/s^3]
Time [s]

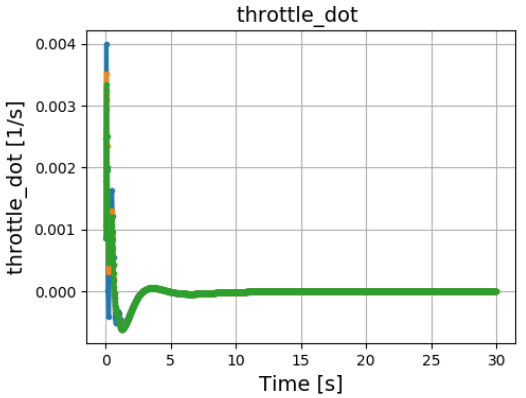

jy(t) local total
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Vertical jerk [m/s^3]
Time [s]

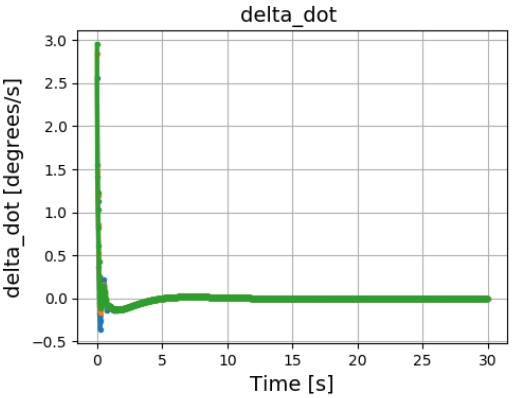

Yaw angle
Yaw angle [degrees]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Yaw angle/s
Yaw angle/s [degrees/s]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
throttle
throttle [-]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
delta
delta [degrees]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Ayt
Ayt [m/s^2]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Ayn
Ayn [m/s^2]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47


Yaw acceleration
Yaw acceleration [degrees/s^2]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47


Axt
Axt [m/s^2]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47


Axn
Axn [m/s^2]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47

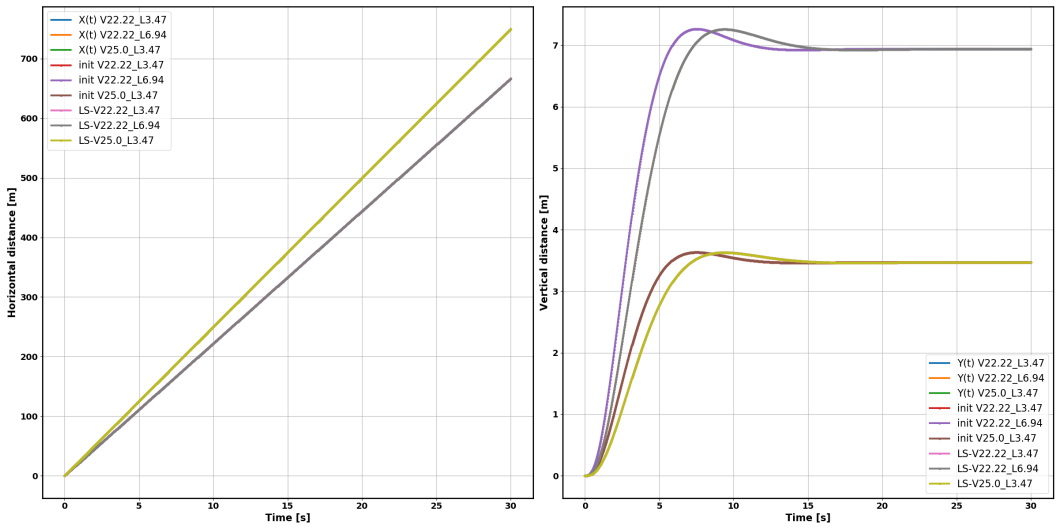

throttle_dot
throttle_dot [1/s]
Time [s]

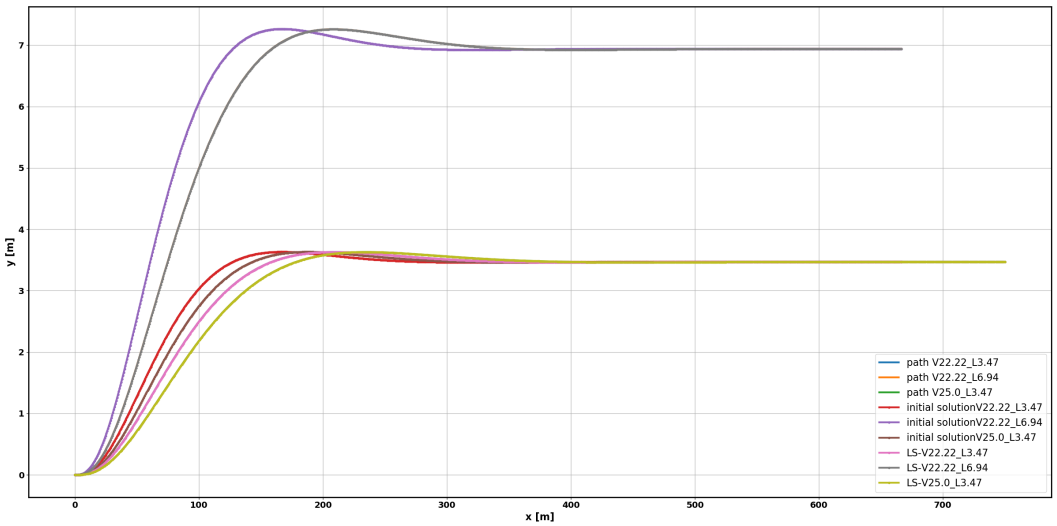

delta_dot
delta_dot [degrees/s]
Time [s]

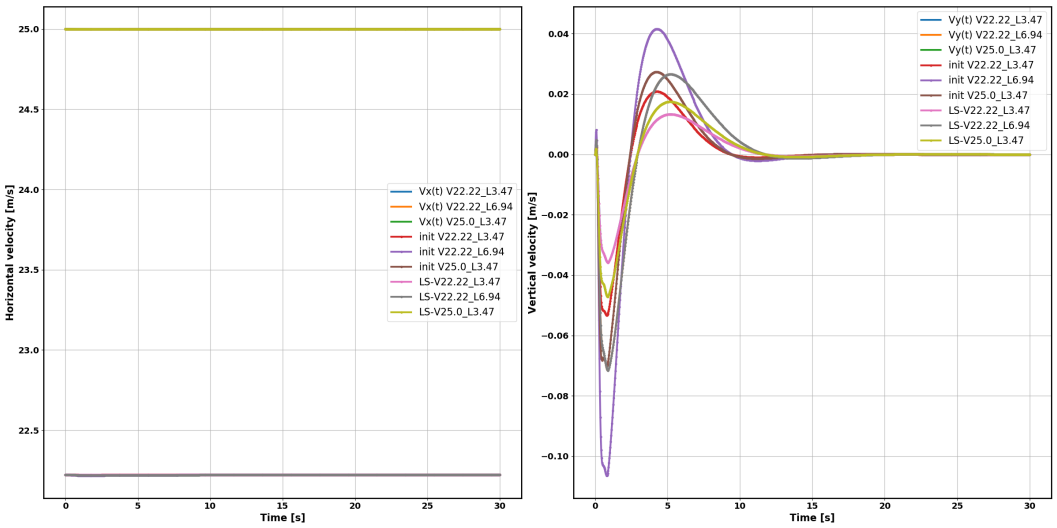
Appendix C

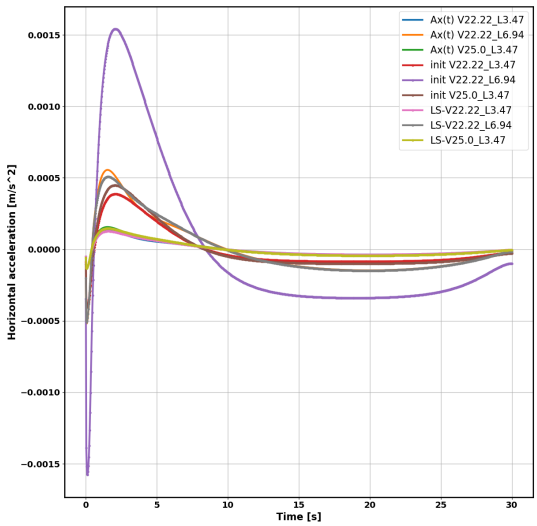
Results of the averaging learning on 3 datasets

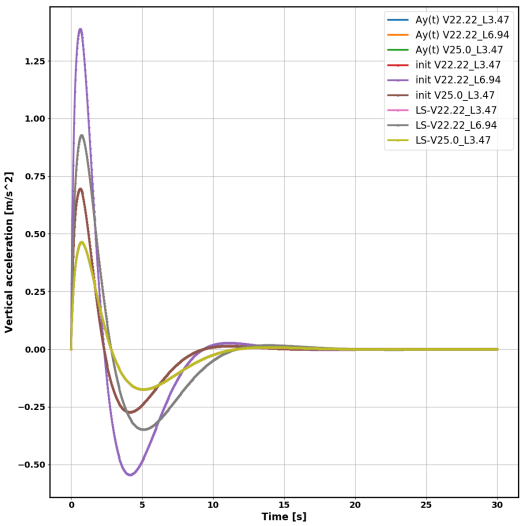
The figures shown in this appendix serve as a completion on the figures that were presented in the discussion of section 4.4.2. It concerns the results of the learning algorithm that learns from three ideal datasets. First the resulting kinematic signals of the bicycle model during the different lane changes are shown. It should be noted that Figure C.1 and C.2 show the angle of the front wheel of the bicycle model. In order to obtain the steerwheelangle, this relation is linearised by the factor $Gs = 16.96$ which means that $\delta_{SWA} = Gs \cdot \delta_{front}$. Further Figure C.3 displays the convergence during the learning process and plots $\boldsymbol{f}_{rel}$ over the iterations. Figure C.4 shows the absolute difference between the learned and observed features. In FigureC.5 the learning of the weights towards the final ones are presented. Figure C.6 gives the difference of current weight with respect to the previous one and as last Figure C.7 shows which of the three RPROP cases that is used in order to update a certain weight.

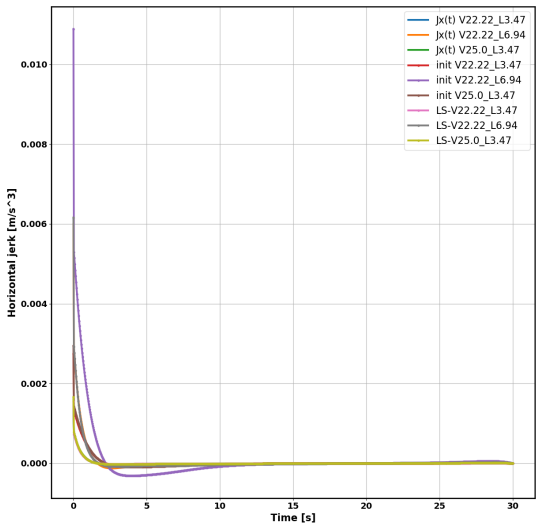


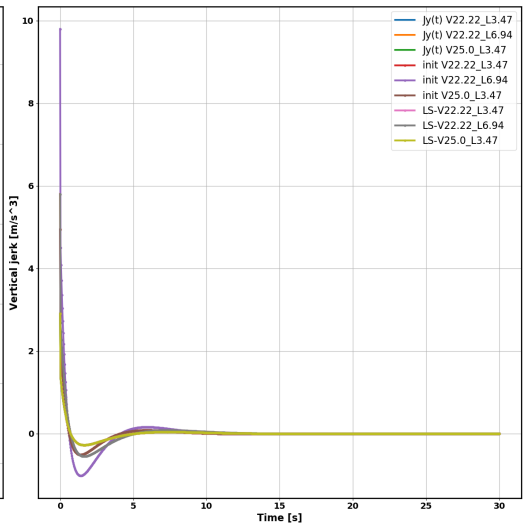

y [m]
x [m]
path V22.22_L3.47
path V22.22_L6.94
path V25.0_L3.47
initial solutionV22.22_L3.47
initial solutionV22.22_L6.94
initial solutionV25.0_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.0_L3.47


Horizontal velocity [m/s]
Time [s]
Vx(t) V22.22_L3.47
Vx(t) V22.22_L6.94
Vx(t) V25.0_L3.47
init V22.22_L3.47
init V22.22_L6.94
init V25.0_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.0_L3.47
Vertical velocity [m/s]
Time [s]
Vy(t) V22.22_L3.47
Vy(t) V22.22_L6.94
Vy(t) V25.0_L3.47
init V22.22_L3.47
init V22.22_L6.94
init V25.0_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.0_L3.47


Ax(t) V22.22_L3.47
Ax(t) V22.22_L6.94
Ax(t) V25.0_L3.47
init V22.22_L3.47
init V22.22_L6.94
init V25.0_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.0_L3.47
Horizontal acceleration [m/s^2]
Time [s]


Ay(t) V22.22_L3.47
Ay(t) V22.22_L6.94
Ay(t) V25.0_L3.47
init V22.22_L3.47
init V22.22_L6.94
init V25.0_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.0_L3.47
Vertical acceleration [m/s^2]
Time [s]


Jx(t) V22.22_L3.47
Jx(t) V22.22_L6.94
Jx(t) V25.0_L3.47
init V22.22_L3.47
init V22.22_L6.94
init V25.0_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.0_L3.47
Horizontal jerk [m/s^3]
Time [s]


Jy(t) V22.22_L3.47
Jy(t) V22.22_L6.94
Jy(t) V25.0_L3.47
init V22.22_L3.47
init V22.22_L6.94
init V25.0_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.0_L3.47
Vertical jerk [m/s^3]
Time [s]

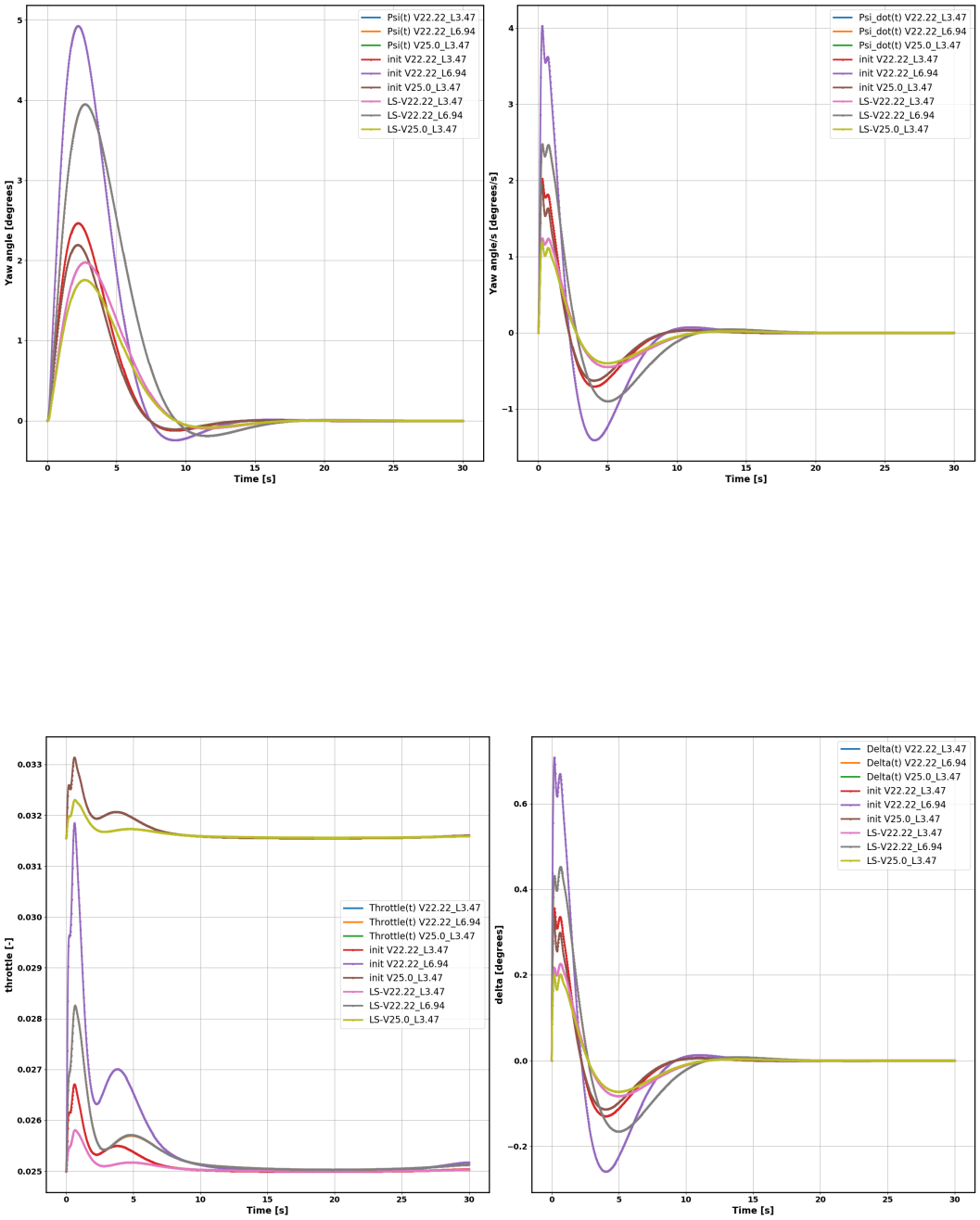
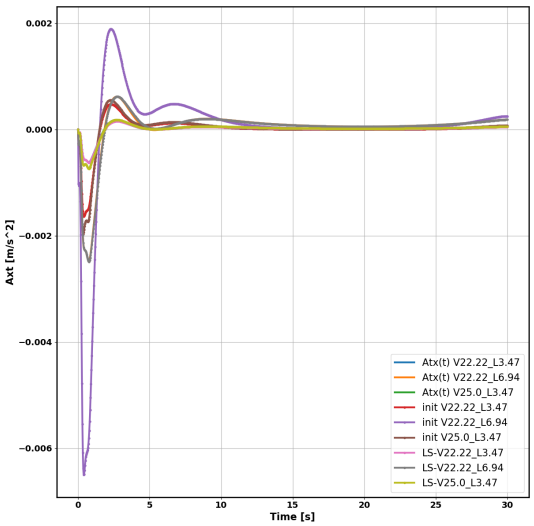
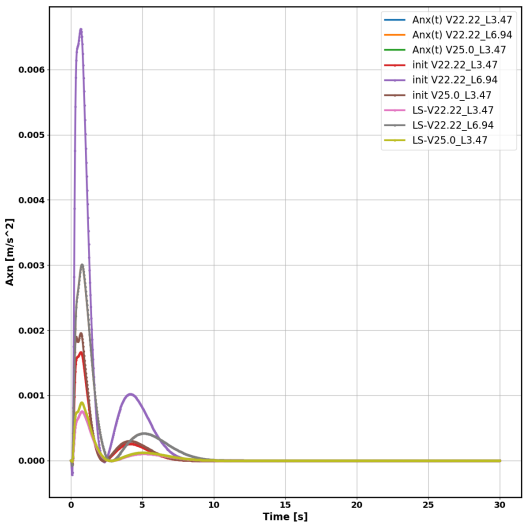
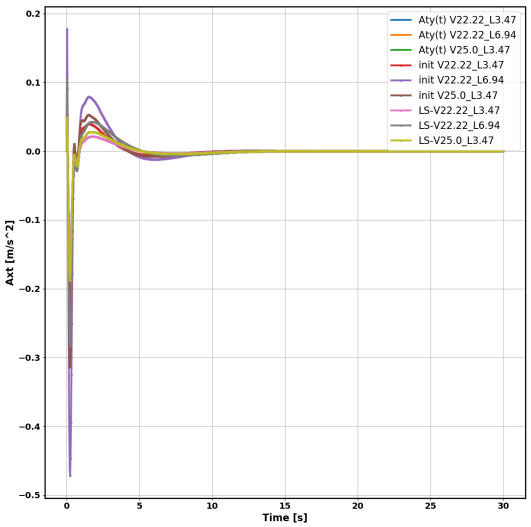
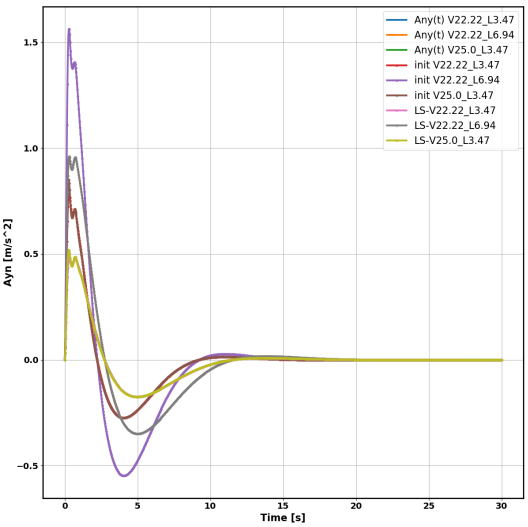


Figure C.1: This figure shows the amount of throttle and the angle of the front wheel of the bicycle model during the lane change maneuvers.


Axt [m/s^2]
Time [s]
Atx(t) V22.22_L3.47
Atx(t) V22.22_L6.94
Atx(t) V25.0_L3.47
init V22.22_L3.47
init V22.22_L6.94
init V25.0_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.0_L3.47


Axn [m/s^2]
Time [s]
Anx(t) V22.22_L3.47
Anx(t) V22.22_L6.94
Anx(t) V25.0_L3.47
init V22.22_L3.47
init V22.22_L6.94
init V25.0_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.0_L3.47


Axt [m/s^2]
Time [s]
Aty(t) V22.22_L3.47
Aty(t) V22.22_L6.94
Aty(t) V25.0_L3.47
init V22.22_L3.47
init V22.22_L6.94
init V25.0_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.0_L3.47


Ayn [m/s^2]
Time [s]
Any(t) V22.22_L3.47
Any(t) V22.22_L6.94
Any(t) V25.0_L3.47
init V22.22_L3.47
init V22.22_L6.94
init V25.0_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.0_L3.47

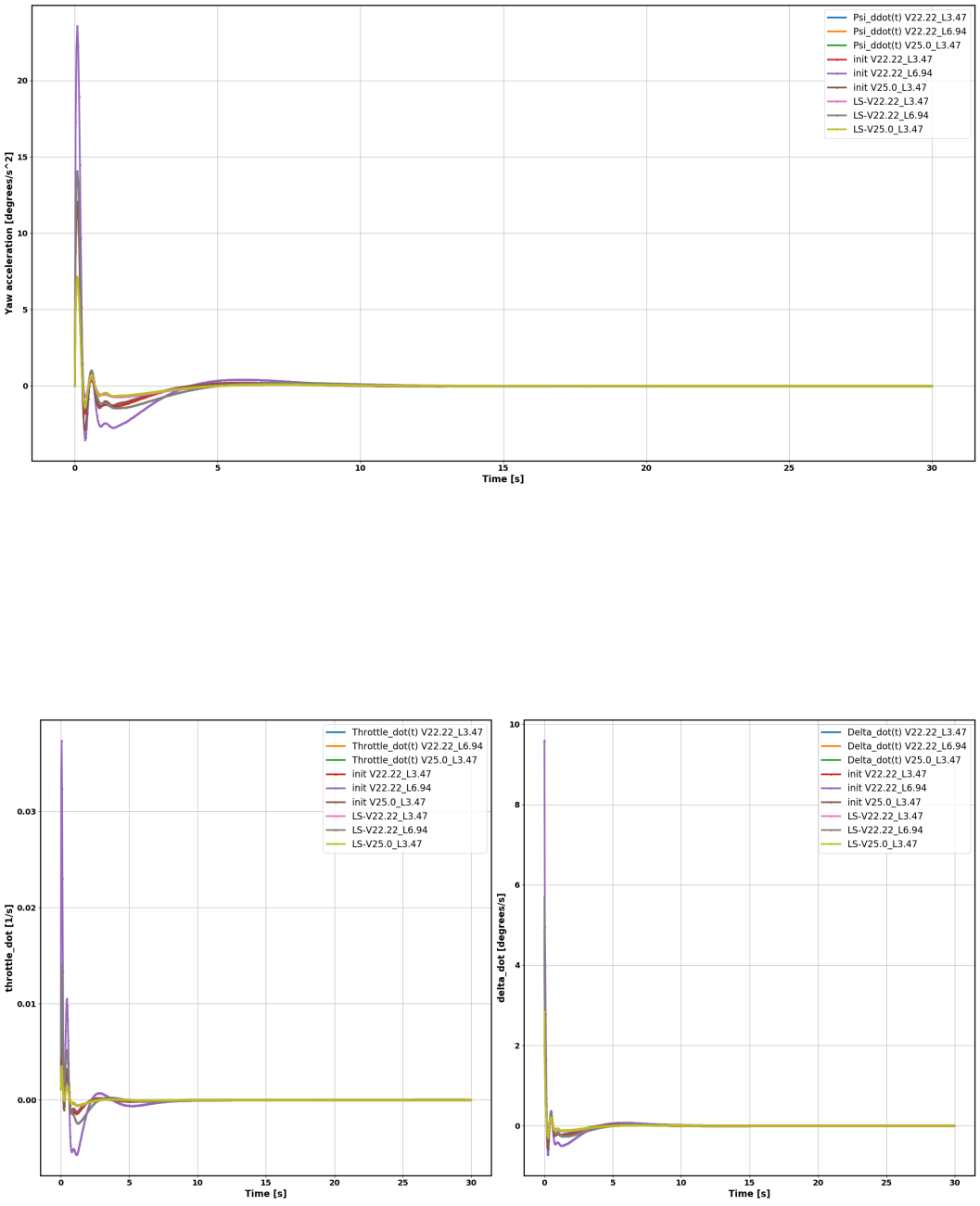


Figure C.2: This figure shows the amount of first derivative of throttle and the first derivative of the angle of the front wheel of the bicycle model is given as input during the lane change maneuvers.

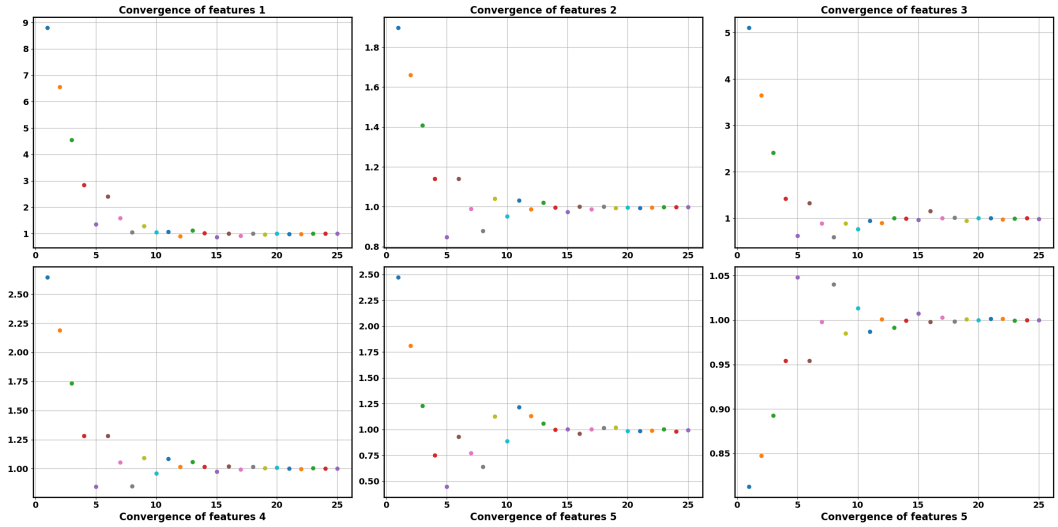


Figure C.3: The convergence of $\boldsymbol{f}_{rel}$ towards one during the learning iterations.

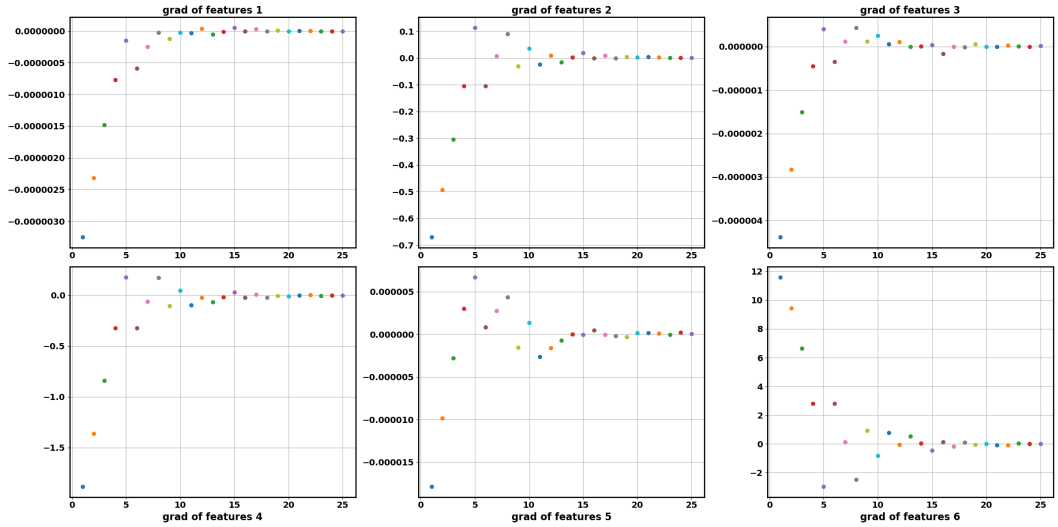


Figure C.4: The gradient $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\theta}}$ estimated by $\boldsymbol{F}_{obs} - \boldsymbol{F}(\boldsymbol{r}_{expected})$ towards zero during the different learning iterations.

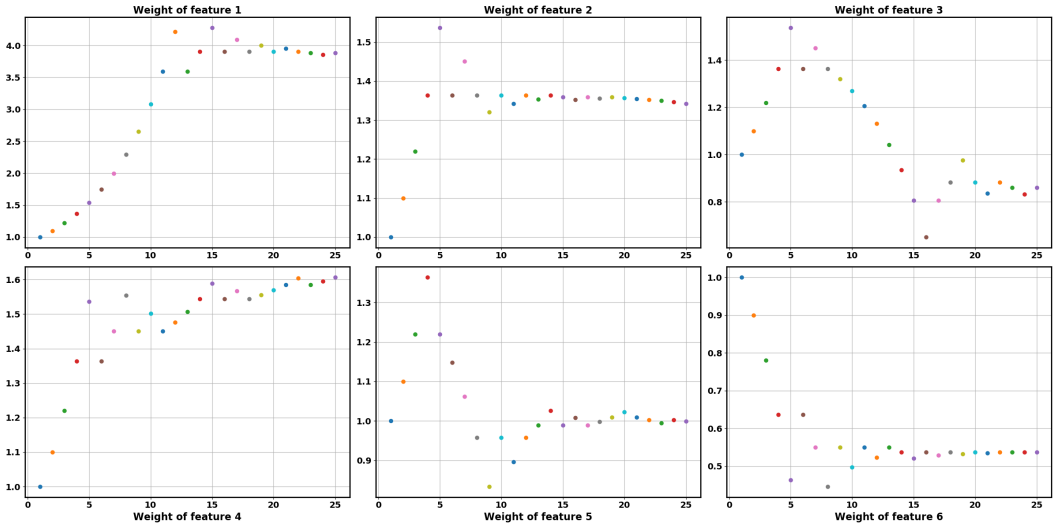


Figure C.5: The learned weights during the different learning iterations.

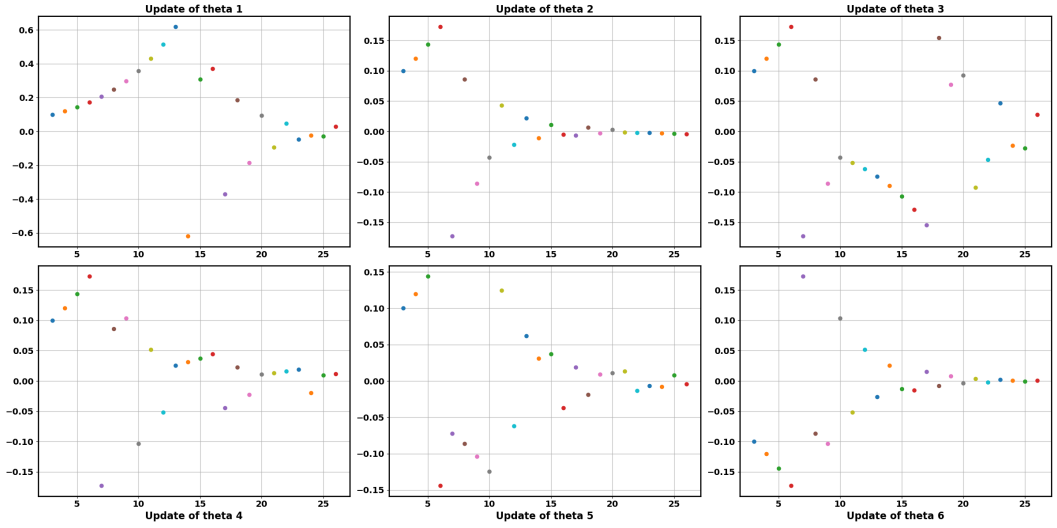


Figure C.6: The difference of $\boldsymbol{\theta}$ with respect to one used in the previous iteration.

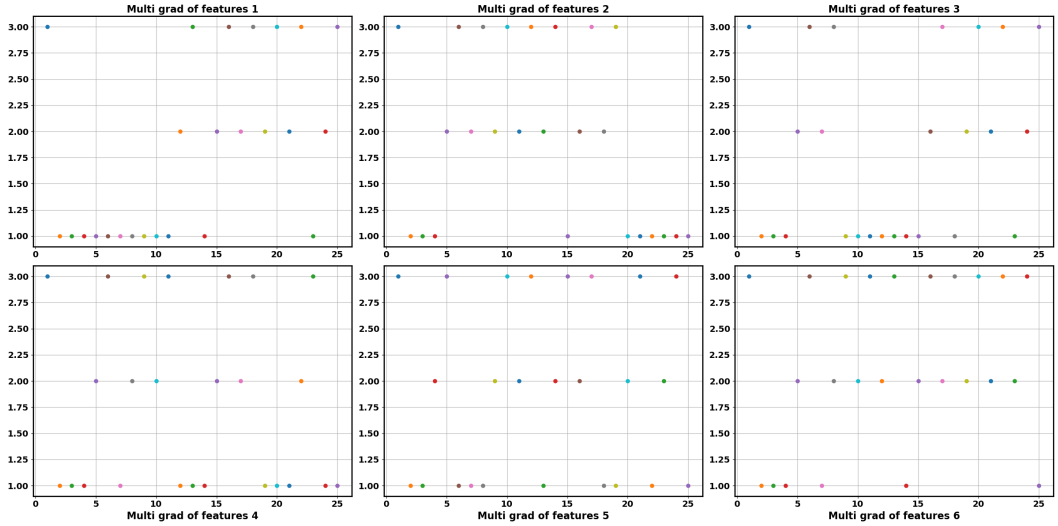
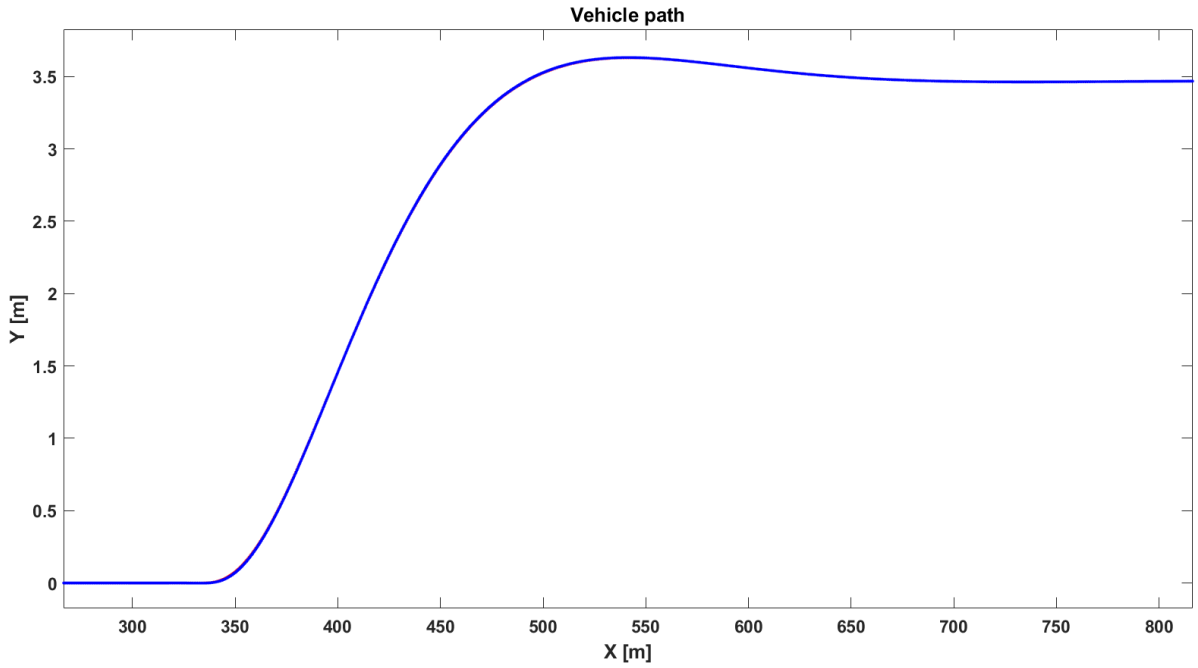


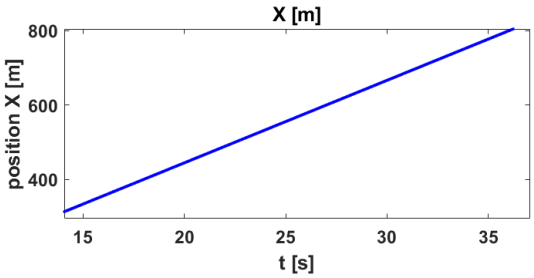
Figure C.7: This figure shows which case of the three available in the RPROP algorithm is chosen during the learning iterations.

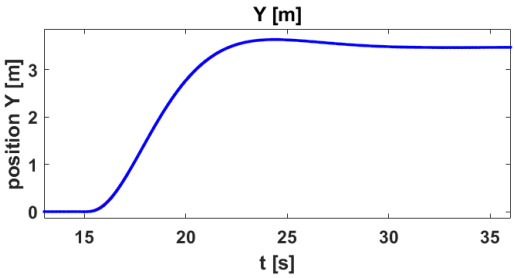
Appendix D

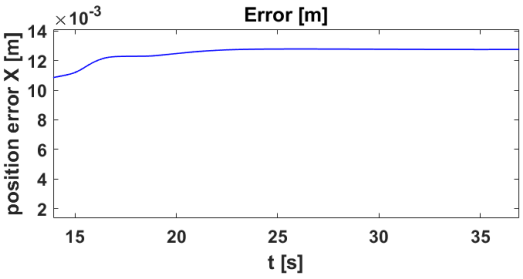
Accuracy results of the tracking MPC

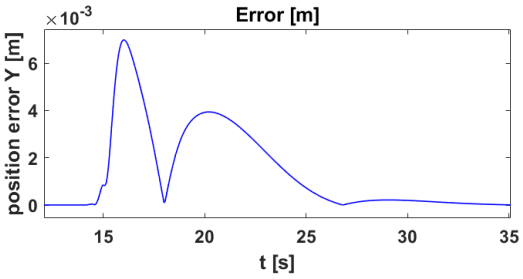
This appendix shows the results that are discussed in 5.1.2. In red the reference is shown that was derived from 4.8 with using the bicycle model and in blue the trajectory completed by the Amesim model. During the learning process with the Amesim model (section 5.2) the first 10 s of the Amesim trajectory is thrown away in order to remove the unstable longitudinal behaviour at the start of the simulation.

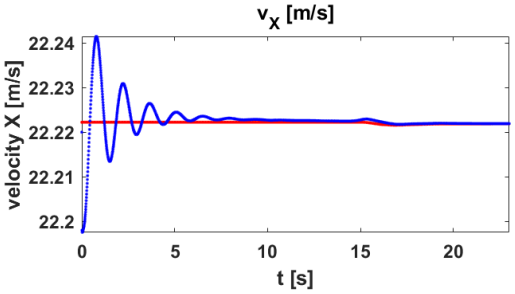


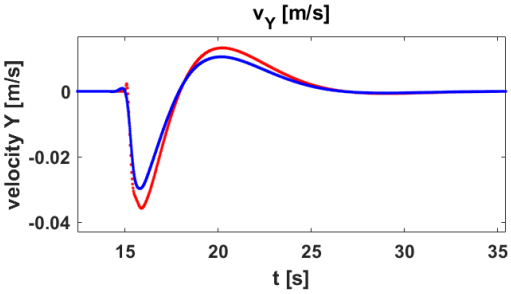

X [m]
position X [m]
800
600
400
15
20
25
30
35
t [s]

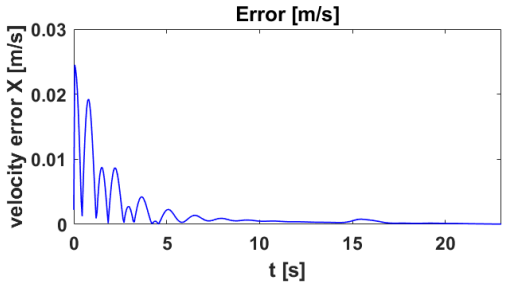

Y [m]
position Y [m]
3
2
1
0
15
20
25
30
35
t [s]

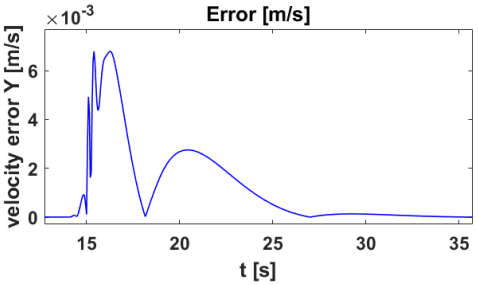

Error [m]
×10-3
position error X [m]
14
12
10
8
6
4
2
15
20
25
30
35
t [s]

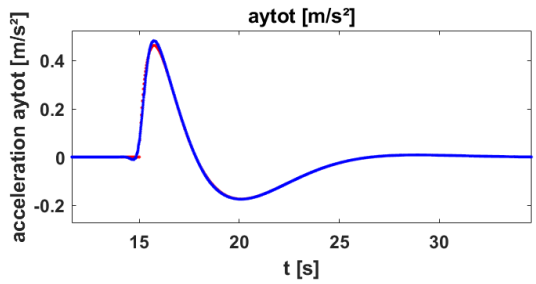

Error [m]
×10-3
position error Y [m]
6
4
2
0
15
20
25
30
35
t [s]

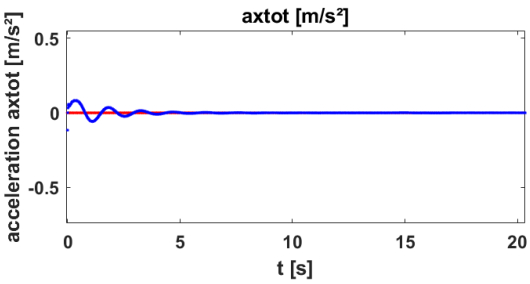

vX [m/s]
velocity X [m/s]
22.24
22.23
22.22
22.21
22.2
0
5
10
15
20
t [s]

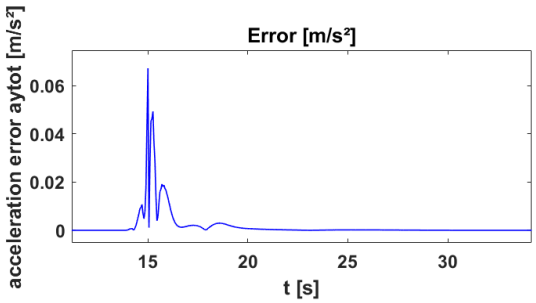

vY [m/s]
velocity Y [m/s]
0
-0.02
-0.04
15
20
25
30
35
t [s]

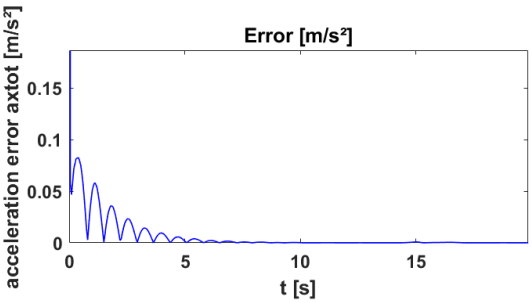

Error [m/s]
velocity error X [m/s]
0.03
0.02
0.01
0
0
5
10
15
20
t [s]

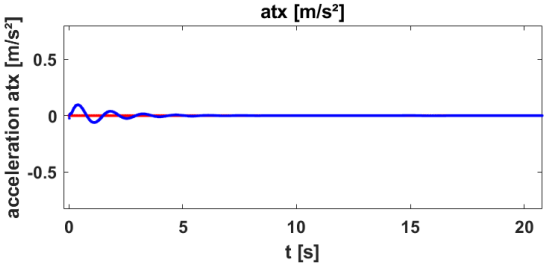

Error [m/s]
×10-3
velocity error Y [m/s]
6
4
2
0
15
20
25
30
35
t [s]

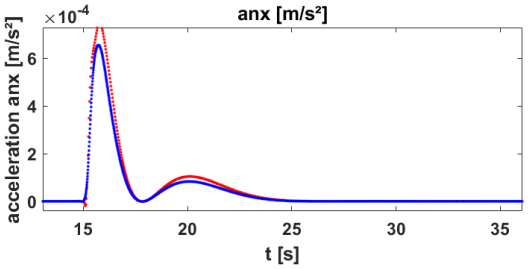

aytot [m/s²]
acceleration aytot [m/s²]
0.4
0.2
0
-0.2
15
20
25
30
t [s]

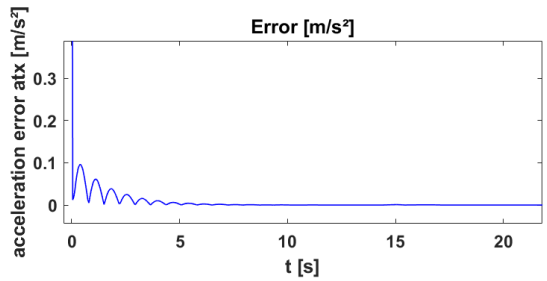

axtot [m/s²]
acceleration axtot [m/s²]
0.5
0
-0.5
0
5
10
15
20
t [s]

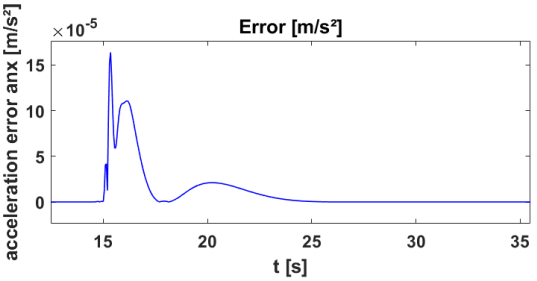

Error [m/s²]
acceleration error aytot [m/s²]
0.06
0.04
0.02
0
15
20
25
30
t [s]

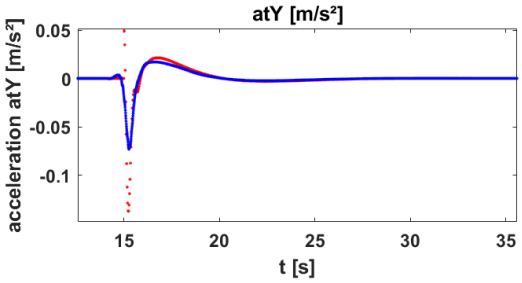

Error [m/s²]
acceleration error axtot [m/s²]
0.15
0.1
0.05
0
0
5
10
15
t [s]

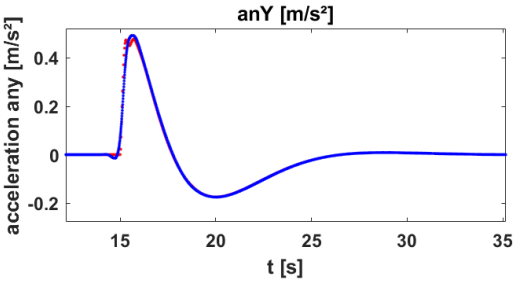

atx [m/s²]
acceleration atx [m/s²]
0.5
0
-0.5
0
5
10
15
20
t [s]

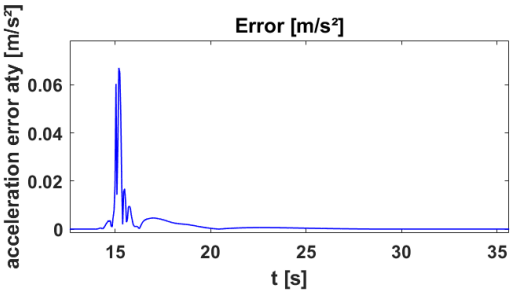

×10-4
anx [m/s²]
acceleration anx [m/s²]
6
4
2
0
15
20
25
30
35
t [s]

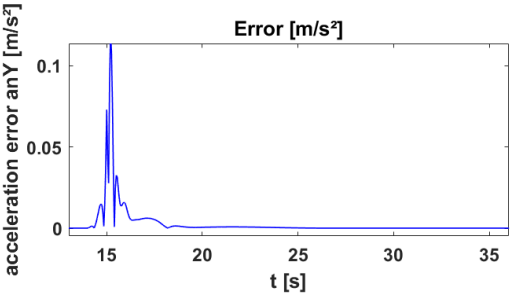

Error [m/s²]
acceleration error atx [m/s²]
0.3
0.2
0.1
0
0
5
10
15
20
t [s]

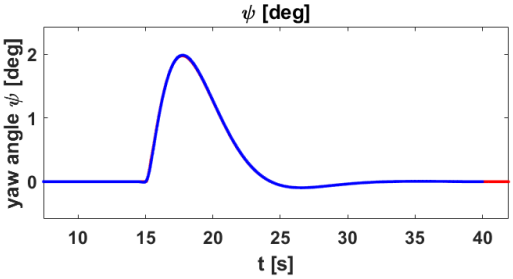

×10-5
Error [m/s²]
acceleration error anx [m/s²]
15
10
5
0
15
20
25
30
35
t [s]

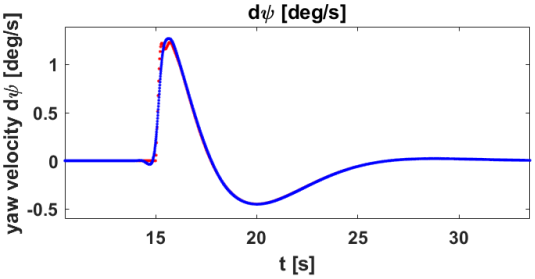

atY [m/s²]
acceleration atY [m/s²]
0.05
0
-0.05
-0.1
15
20
25
30
35
t [s]

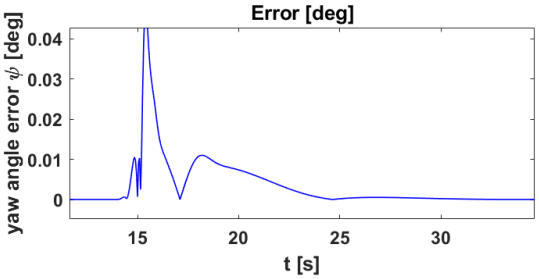

anY [m/s²]
acceleration any [m/s²]
0.4
0.2
0
-0.2
15
20
25
30
35
t [s]

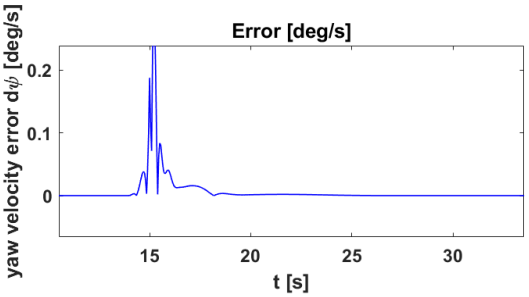

Error [m/s²]
acceleration error aty [m/s²]
0.06
0.04
0.02
0
15
20
25
30
35
t [s]

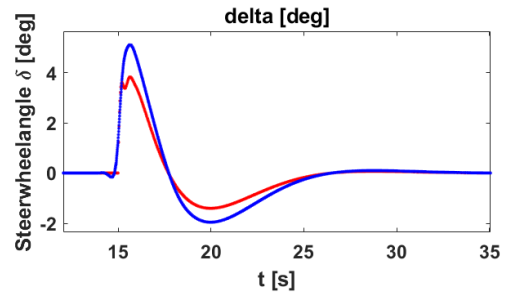

Error [m/s²]
acceleration error anY [m/s²]
0.1
0.05
0
15
20
25
30
35
t [s]

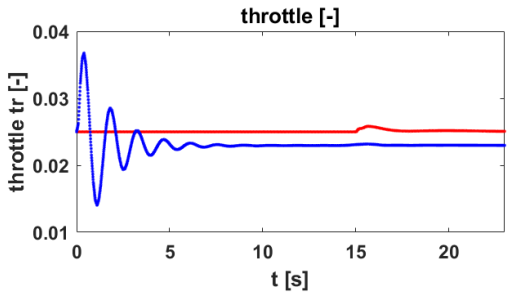

ψ [deg]
yaw angle ψ [deg]
2
1
0
10
15
20
25
30
35
40
t [s]

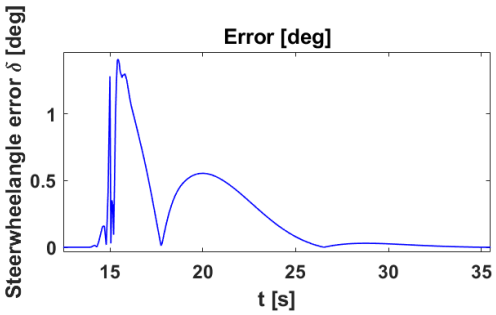

dψ [deg/s]
yaw velocity dψ [deg/s]
1
0.5
0
-0.5
15
20
25
30
t [s]

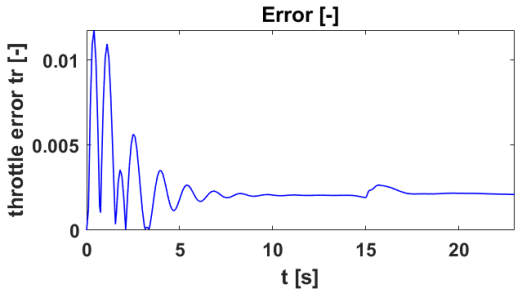

Error [deg]
yaw angle error ψ [deg]
0.04
0.03
0.02
0.01
0
15
20
25
30
t [s]

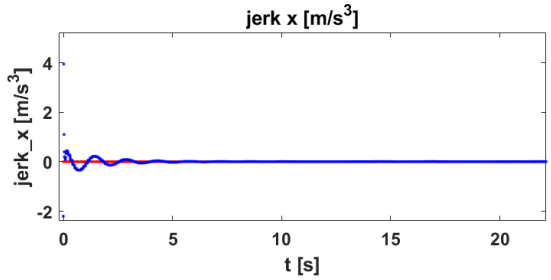

Error [deg/s]
yaw velocity error dψ [deg/s]
0.2
0.1
0
15
20
25
30
t [s]

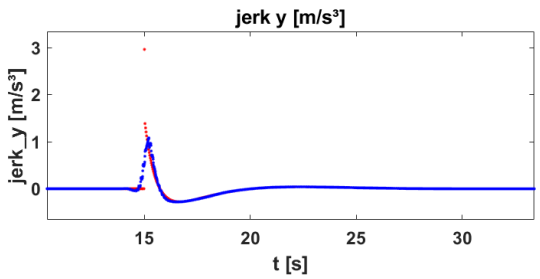

delta [deg]
Steerwheelangle δ [deg]
t [s]

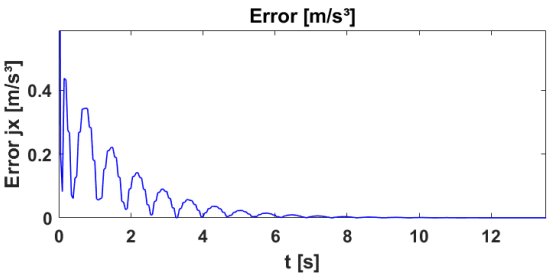

throttle [-]
throttle tr [-]
t [s]

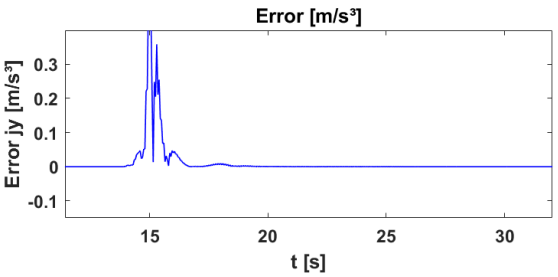

Error [deg]
Steerwheelangle error δ [deg]
t [s]

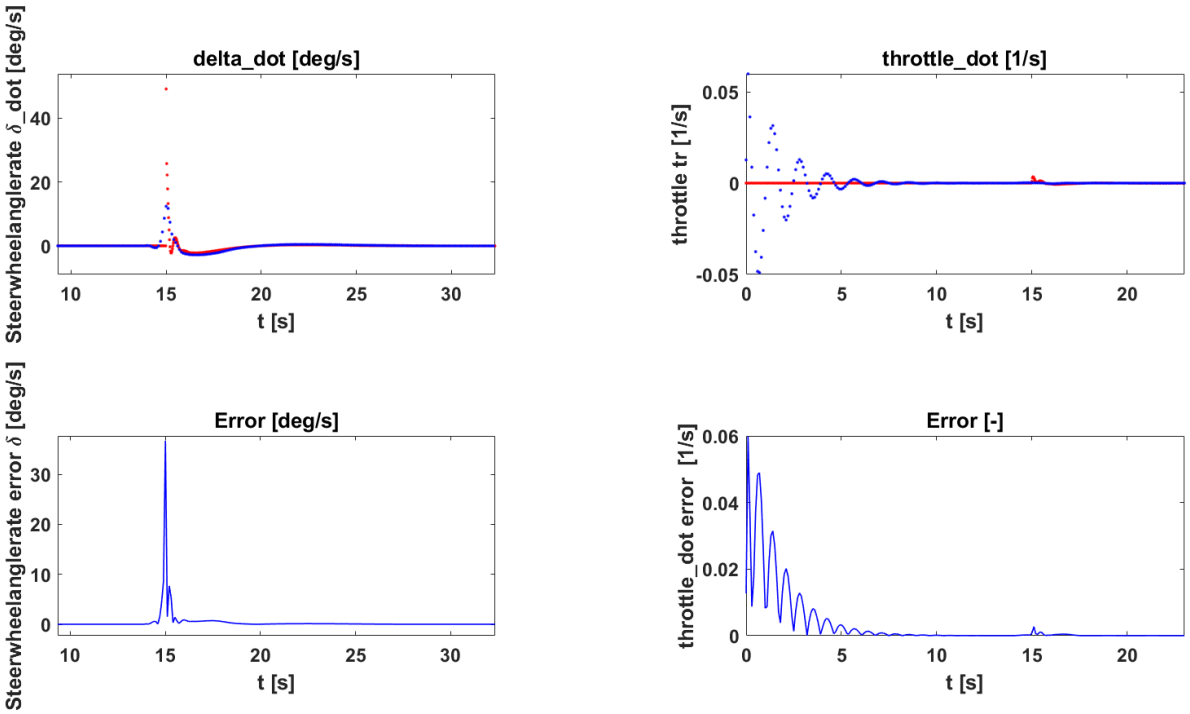

Error [-]
throttle error tr [-]
t [s]


jerk x [m/s3]
jerk_x [m/s3]
t [s]


jerk y [m/s³]
jerk_y [m/s³]
t [s]


Error [m/s³]
Error jx [m/s³]
t [s]

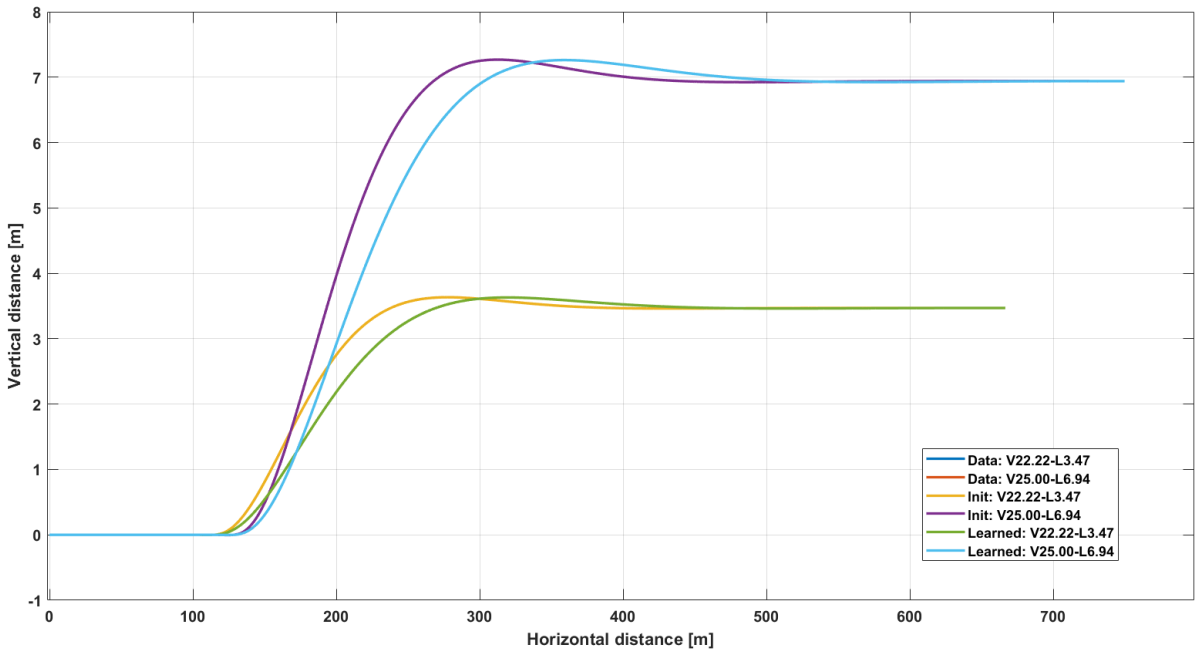

Error [m/s³]
Error jy [m/s³]
t [s]

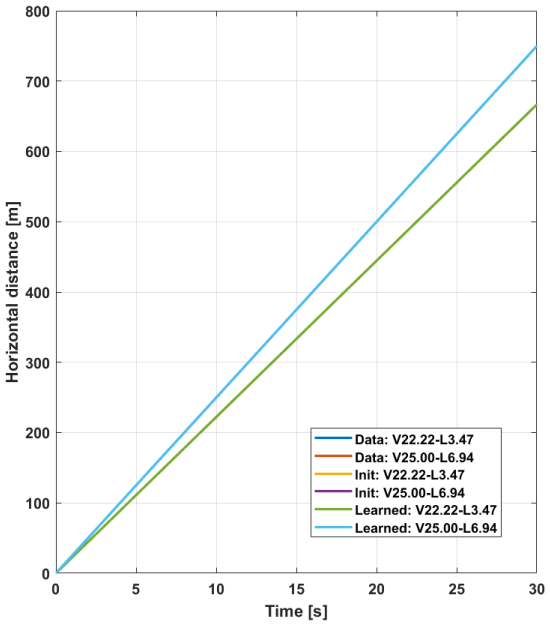

delta_dot [deg/s]
Steerwheelanglerate δ_dot [deg/s]
t [s]
throttle_dot [1/s]
throttle tr [1/s]
t [s]
Error [deg/s]
Steerwheelanglerate error δ [deg/s]
t [s]
Error [-]
throttle_dot error [1/s]
t [s]

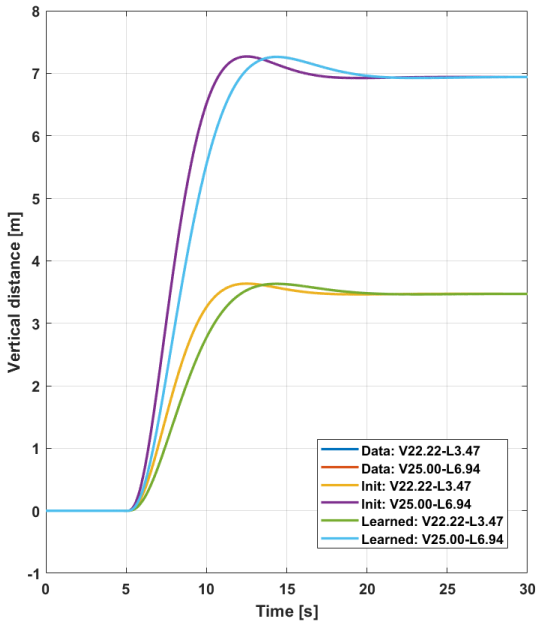
Appendix E

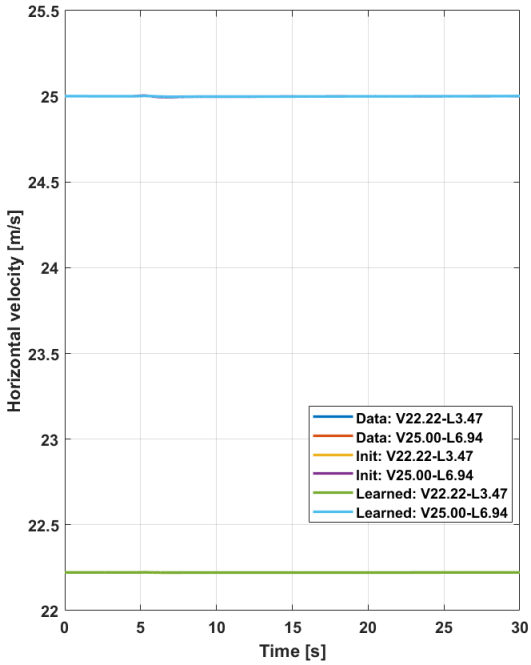
Results of learning with the 15 dof Amesim model

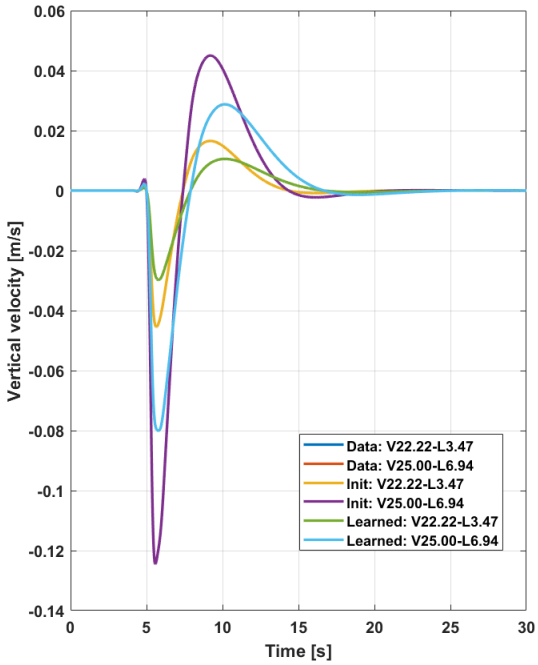
This appendix shows the learning results of the 15 dof Amesim model on two distinct as possible datasets $V022.22-L3.47$ and $V025.00-L6.94$. Except for the longitudinal total acceleration and jerk, all the learned kinematic signals match closely with the observed one. The most important conclusions about the results can be find in section 5.2.
It should be noted that Figure E.1 and E.2 show the angle of the front wheel of the bicycle model. In order to obtain the steerwheelangle, this relation is linearised by the factor $Gs = 16.96$ which means that $\delta_{SWA} = Gs \cdot \delta_{front}$. Further Figure E.3 displays the convergence during the learning process and plots $\boldsymbol{f}_{rel}$ over the iterations. Figure E.4 shows the absolute difference between the learned and observed features. In FigureE.5 the learning of the weights towards the final ones are presented. Figure E.6 gives the difference of current weight with respect to the previous one and as last Figure E.7 shows which of the three RPROP cases that is used in order to update a certain weight.

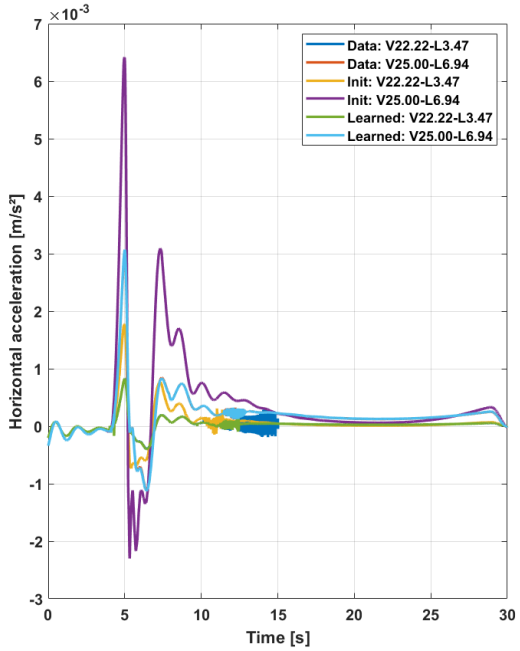

Vertical distance [m]
Horizontal distance [m]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94

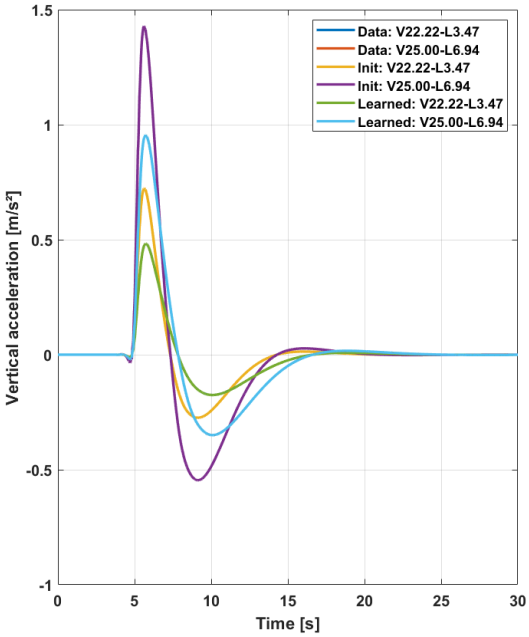

Horizontal distance [m]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94

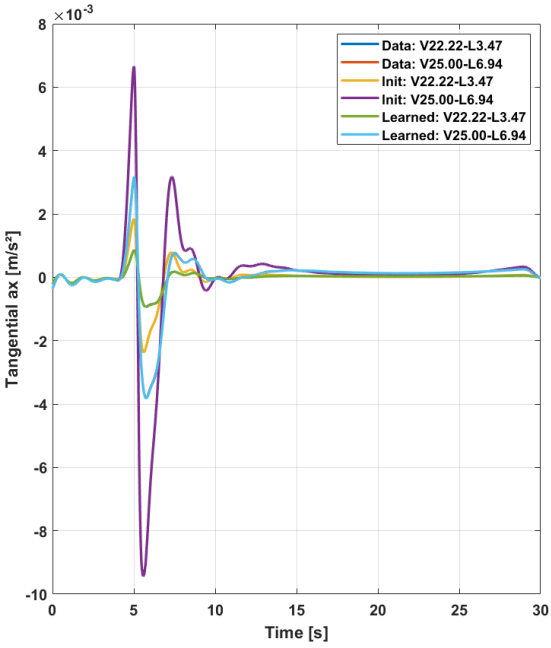

Vertical distance [m]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94

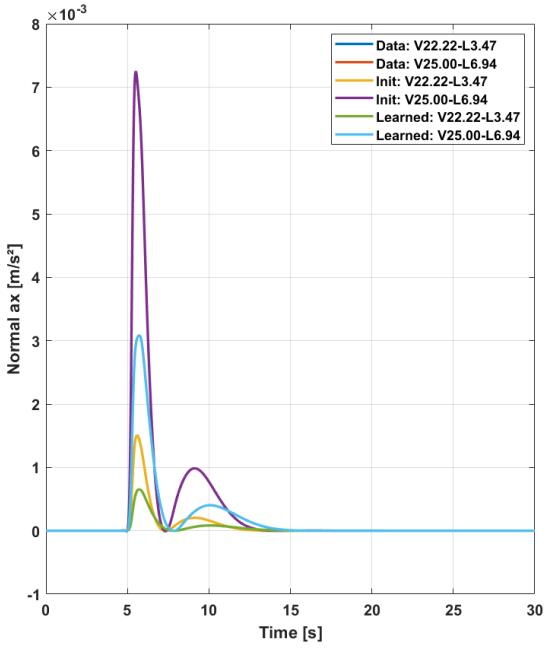

Horizontal velocity [m/s]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94

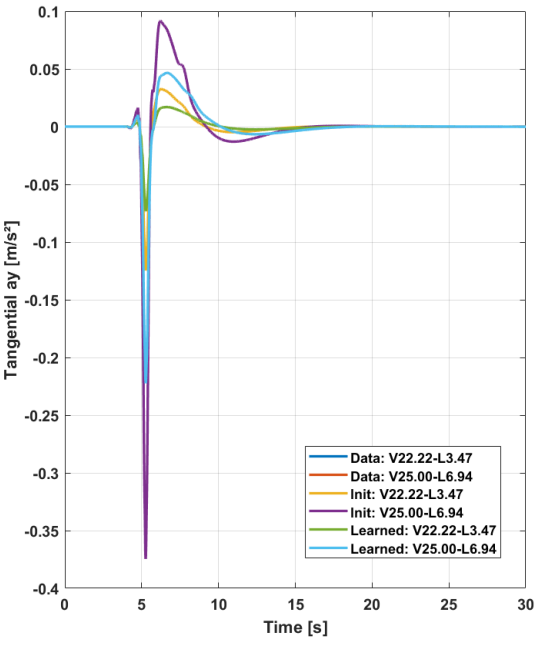

Vertical velocity [m/s]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94

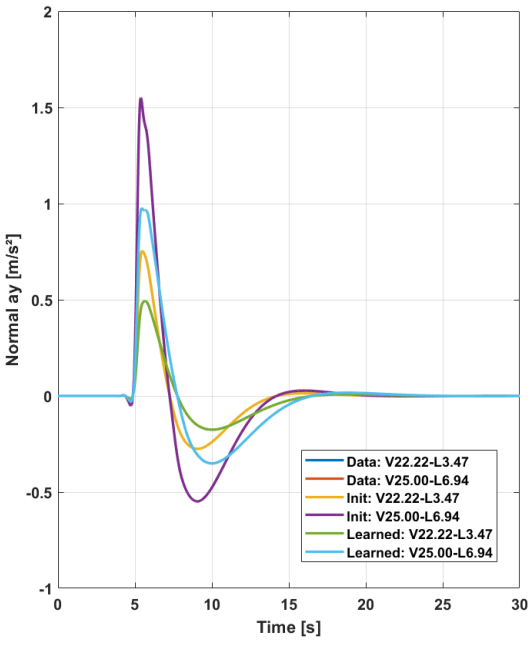

×10-3
Horizontal acceleration [m/s²]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94

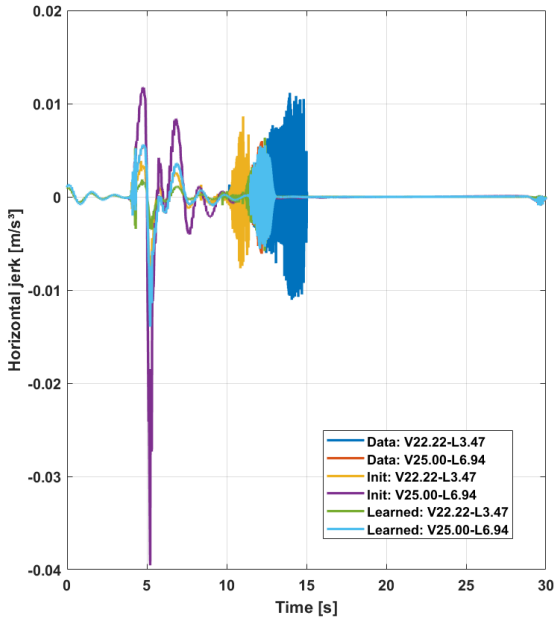

Vertical acceleration [m/s²]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94

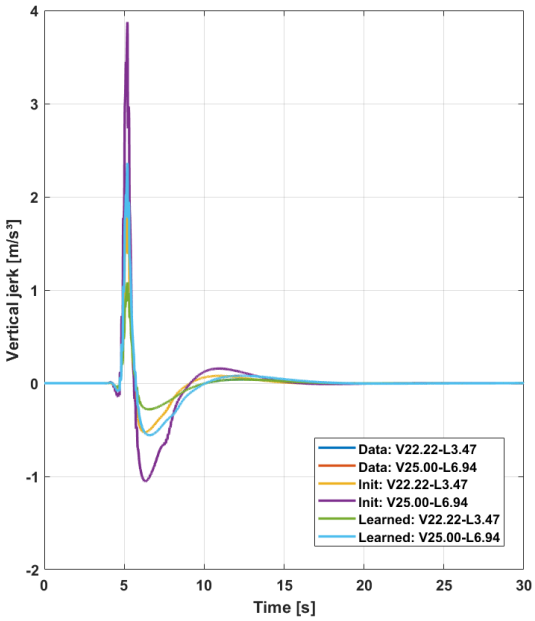

×10^-3
Tangential ax [m/s²]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94

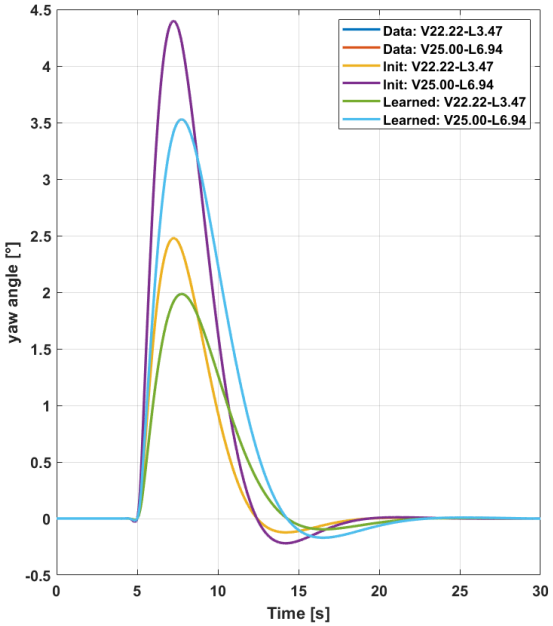

×10^-3
Normal ax [m/s²]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94

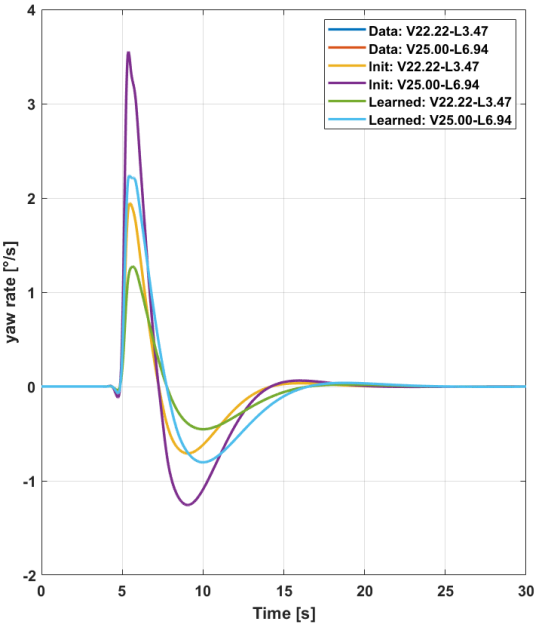

Tangential ay [m/s²]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94


Normal ay [m/s²]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94


Horizontal jerk [m/s³]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94


Vertical jerk [m/s³]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94


yaw angle [°]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94


yaw rate [°/s]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94

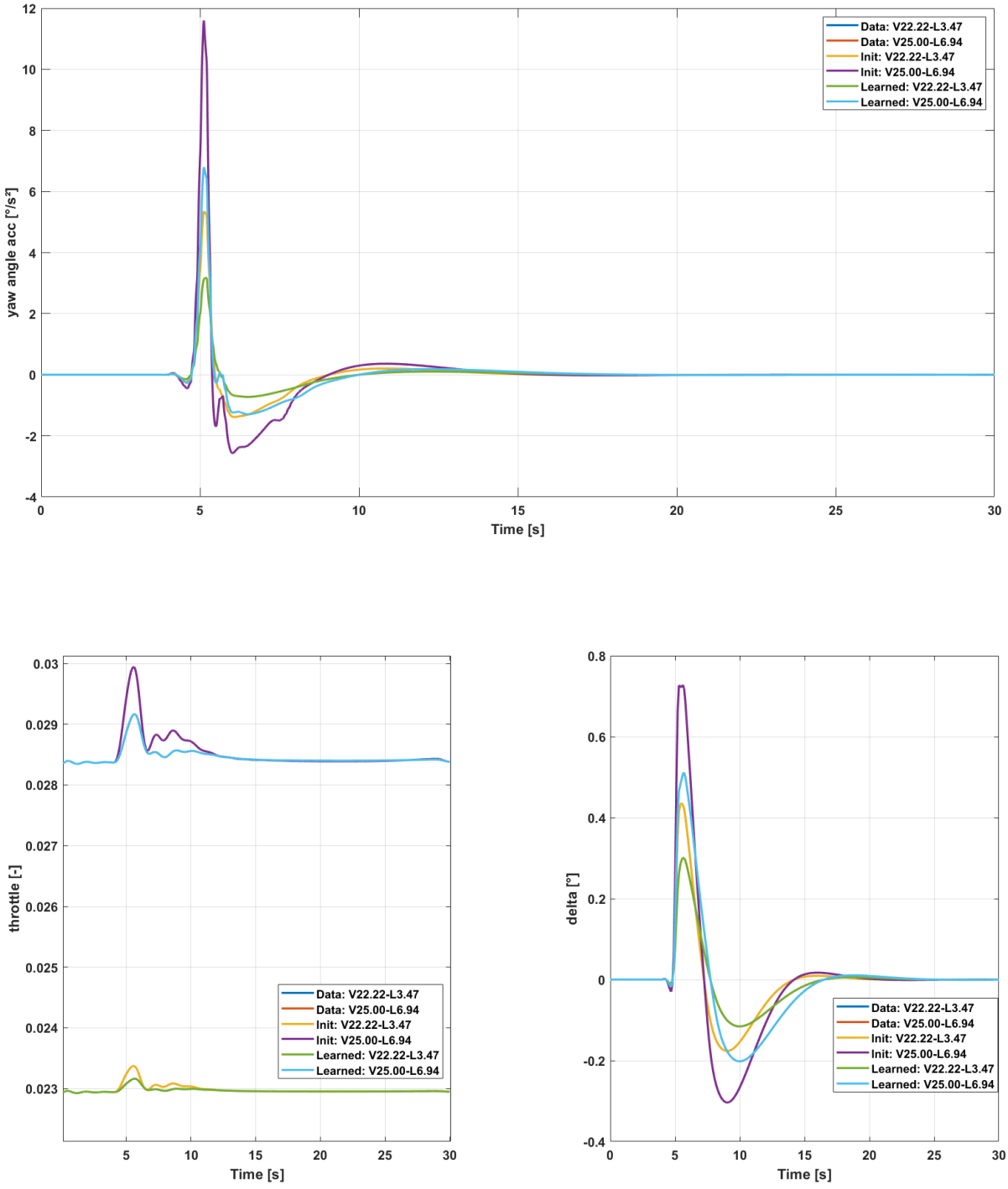


Figure E.1: This figure shows the amount of throttle and the angle of the front wheel of the bicycle model during the lane change maneuvers.

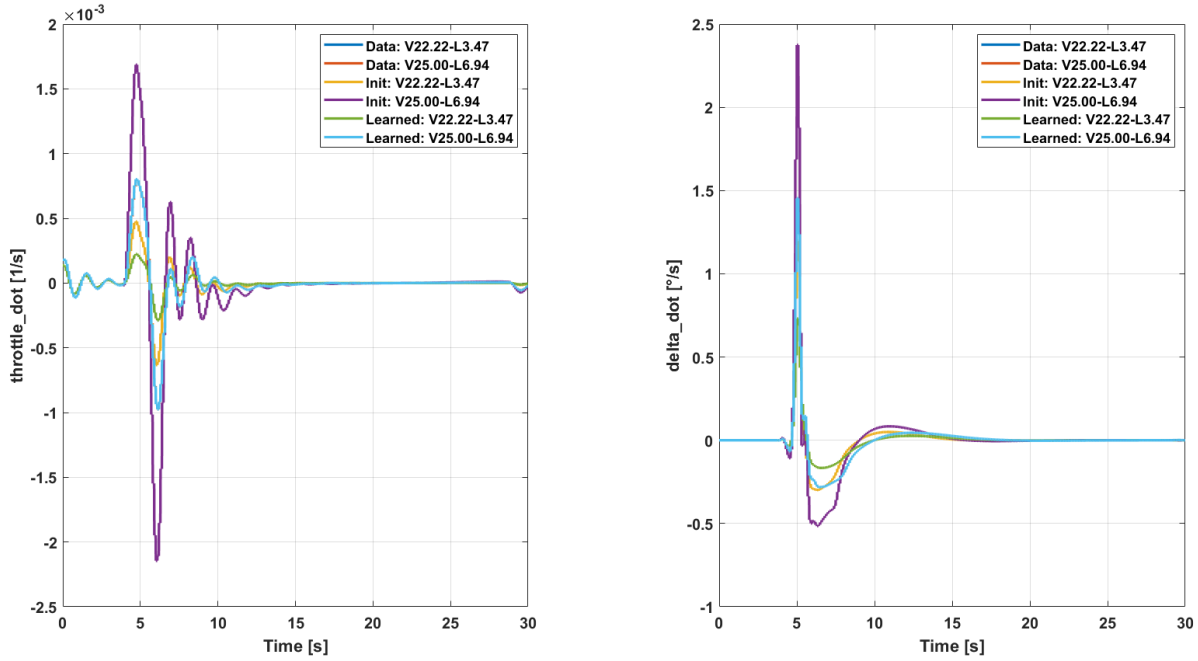


Figure E.2: This figure shows the amount of first derivative of throttle and the first derivative of the angle of the front wheel of the bicycle model is given as input during the lane change maneuvers.

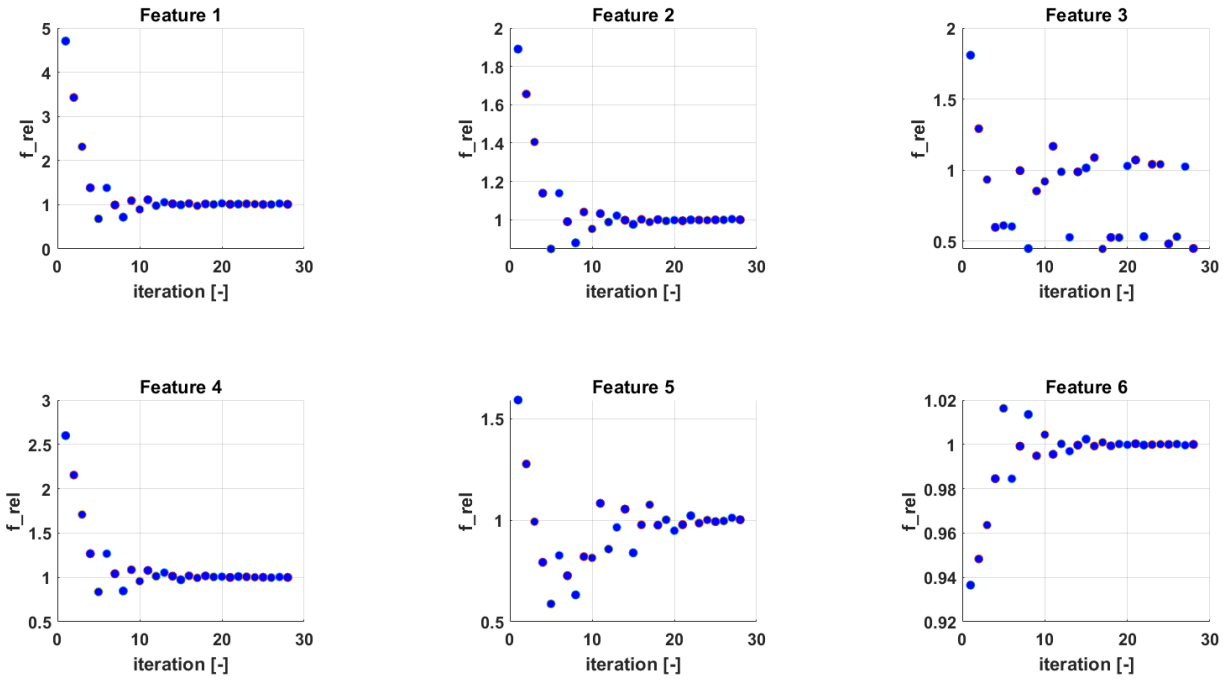


Figure E.3: The convergence of $\boldsymbol{f}_{rel}$ towards one during the learning iterations.

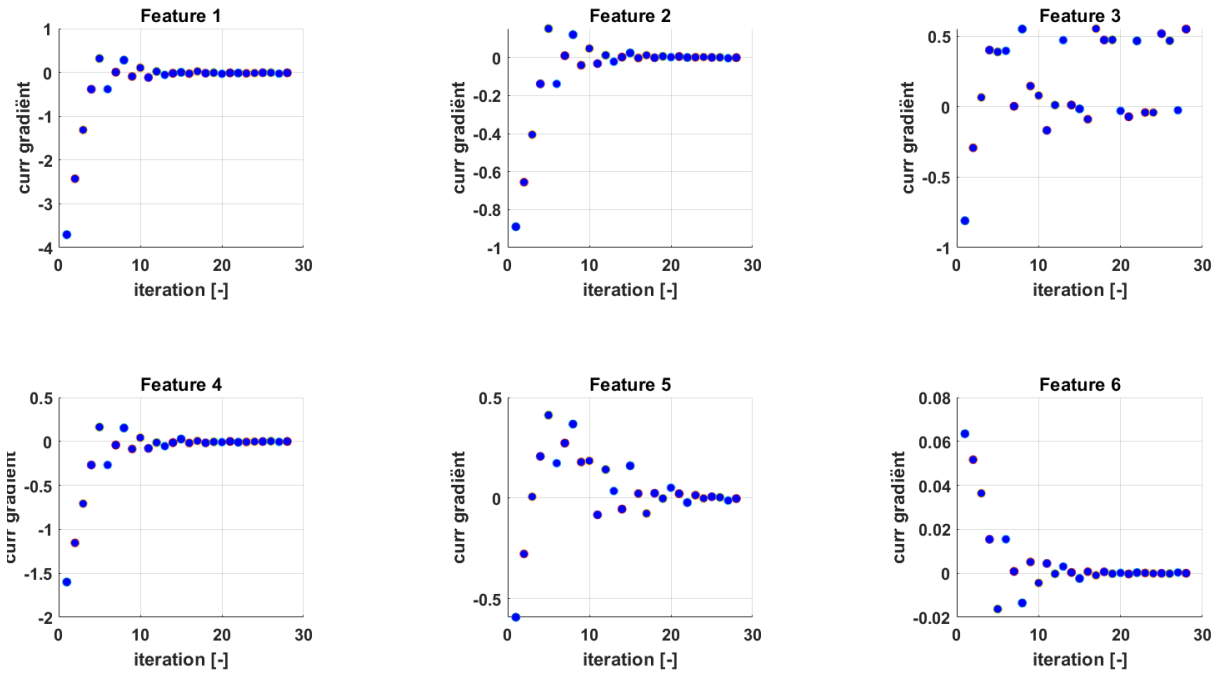


Figure E.4: The gradient $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\theta}}$ estimated by $\boldsymbol{F}_{obs} - \boldsymbol{F}(\boldsymbol{r}_{expected})$ towards zero during the different learning iterations.

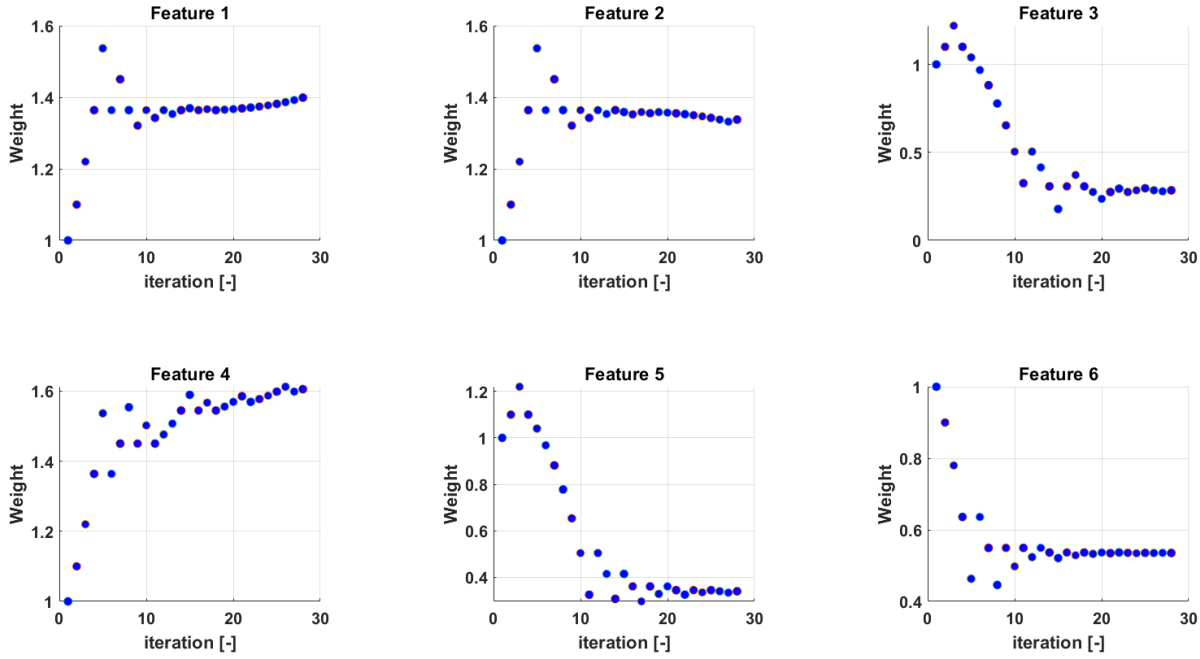


Figure E.5: The learned weights during the different learning iterations.

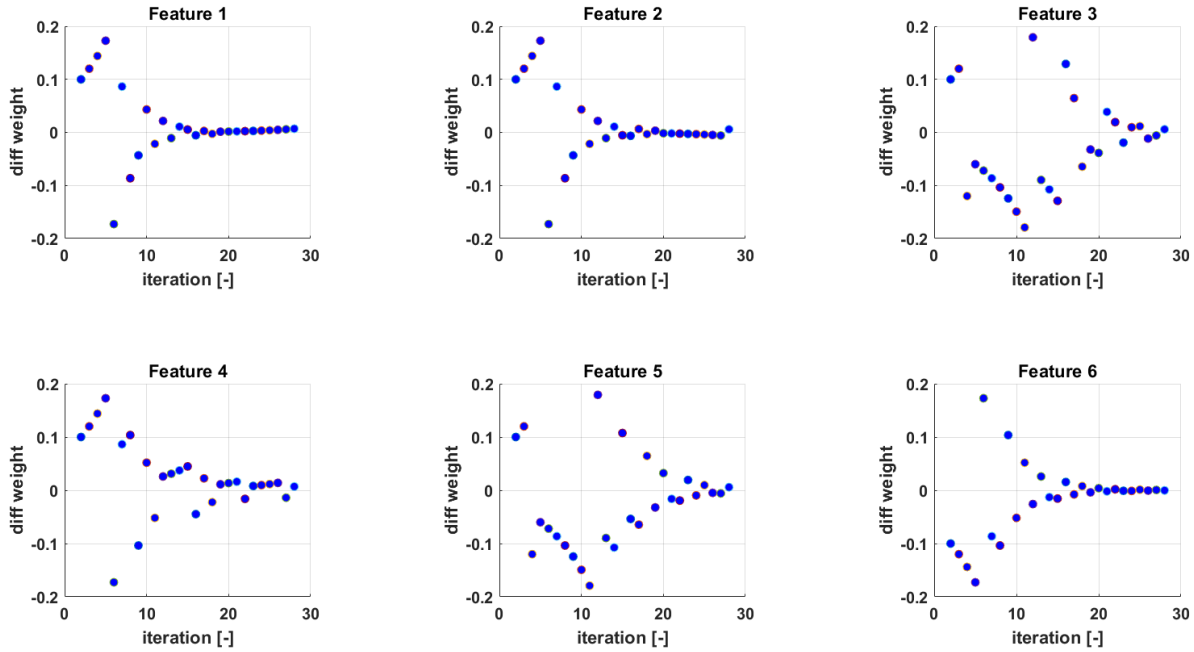


Figure E.6: The difference of $\boldsymbol{\theta}$ with respect to one used in the previous iteration.

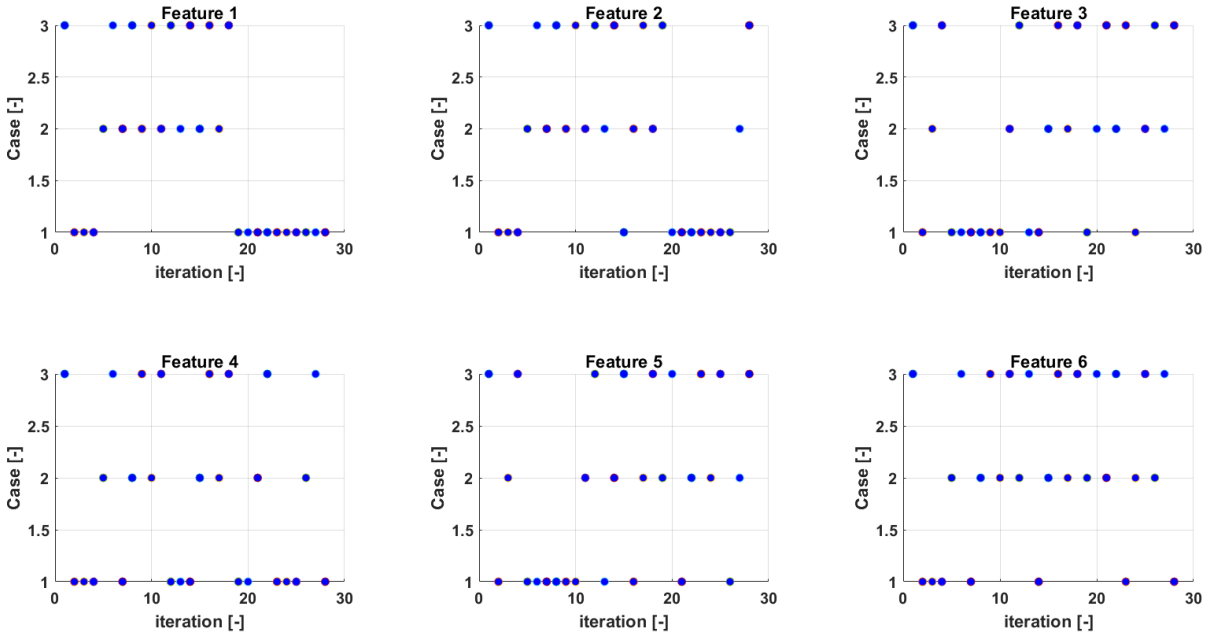


Figure E.7: This figure shows which case of the three available in the RPROP algorithm is chosen during the learning iterations.

Bibliography

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. *Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004*, pages 1–8, 2004.

[2] I. Bae, J. Moon, and J. Seo. Toward a comfortable driving experience for a self-driving shuttle bus. *Electronics (Switzerland)*, 8(9):1–13, 2019.

[3] H. Bellem. *Comfort in Automated Driving : Analysis of Driving Style Preference in Automated Driving*. PhD thesis, 2018.

[4] Brian D. Ziebart, Andrew Maas, J.Andrew Bagnell and A. K. Dey. Maximum Entropy Inverse Reinforcement Learning Brian. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, page 6, 2008.

[5] L. Daniel. Method and system for determining and dynamically updating a route and driving style for passenger comfort - US Patent, 2018.

[6] E. David. Will autonomous vehicles be safe to use? | Cybersecurity & Technology News | Secure Futures | Kaspersky.

[7] M. Elbanhawi, M. Simic, and R. Jazar. In the Passenger Seat: Investigating Ride Comfort Measures in Autonomous Cars. *IEEE Intelligent Transportation Systems Magazine*, 7(3):4–17, 2015.

[8] C. Gianna, S. Heimbrand, and M. Gresty. Thresholds for detection of motion direction during passive lateral whole-body acceleration in normal subjects and patients with bilateral loss of labyrinthine function. *Brain Research Bulletin*, 40(5-6):443–447, 1996.

[9] J. Gillis. Ya Coda course presentation, 2019.

[10] H. Kretzschmar, M. Kuderer, and W. Burgard. Learning to predict trajectories of cooperatively navigating agents. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4015–4020, 2014.

[11] M. Kuderer, S. Gulati, and W. Burgard. Learning driving styles for autonomous vehicles from demonstration. *Proceedings - IEEE International Conference on Robotics and Automation*, 2015-June(June):2641–2646, 2015.

[12] T. Mercy. *Spline-Based Motion Planning for Autonomous Mechatronic Systems*. PhD thesis, 2018.

[13] P. Patrinos. Model Predictive Control - Lecture Notes, 2019.

[14] P. Patrinos. Optimization - Lecture notes, 2019.

[15] V. Powers, C., Mellinger, D., Kushleyev, A., Kothmann, B., Kumar. Experimental Robotics. *ISER, June*, 88(287513):515–529, 2013.

[16] Prof. Amnon Shashua. Experience Counts, Particularly in Safety-Critical Areas | Intel Newsroom, mar 2018.

[17] M. Riedmiller and H. Braun. Direct adaptive method for faster backpropagation learning: The RPROP algorithm. *1993 IEEE International Conference on Neural Networks*, pages 586–591, 1993.

[18] Q. N. Tong Duy Son. Safety-Critical Control for Non-affine Nonlinear Systems with Application on Autonomous Vehicle. (August):7, 2019.

[19] M. Turner and M. J. Griffin. Motion sickness in public road transport: The effect of driver, route and vehicle. *Ergonomics*, 42(12):1646–1664, 1999.

[20] University of Warwick. Do passengers prefer autonomous vehicles driven like machines or like humans?, 2019.

[21] K. Yankov. *Potential Field Based Model Predictive Control for Autonomous Vehicle Motion Planning and Control*. PhD thesis.

[22] Yusof N.B.M. *Comfort in Autonomous Car : Mitigating Motion Sickness by Enhancing Situation Awareness through Haptic Displays Nidzamuddin Md . Yusof*. PhD thesis, Eindhoven, 2019.