KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Learning comfort objective from lane change demonstrations for optimal control

Stijn Staring

Thesis submitted for the degree of Master of Science in Mechanical Engineering

Thesis supervisor:
Prof. dr. ir. Jan Swevers

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

Mentor:
Dr. ir. Son Tong

Academic year 2019 – 2020

Preface

I am very pleased to present my master thesis to complete my Master of Science in Mechanical Engineering. Conducting this research was an informative process in which I was able to apply the knowledge and skills that I gained during my studies. Writing my thesis and thus completing my studies would not have been possible without the support of many people for which I am grateful. First of all, I would like to thank my promoter Professor Swevers from the KU Leuven and my mentor, Dr. Tong from Siemens for the professional discussions and tips in order to improve my results. I also want to thank Flavia Acerbo, employee at Siemens, who answered my questions on several occasions. Last but not least, I would like to thank my family for their ongoing support during all phases of my studies. They have always been my biggest fans and I could not have done this without the opportunities they have given me.

Stijn Staring

Contents

Abstract

The human driver transforms from being an active traffic participant into a passive agent due to the developments in autonomous driving. Consequently a motorist involvement becomes limited to perceiving and rating decisions taken by the vehicle. In order to enhance user acceptance and to increase competitiveness, it is important that a notion of driver specific comfort is taken into account i.e. trade offs made between different comfort contributors.

Therefore the objective of this research is to develop an algorithm making use of feature-based inverse reinforcement learning which is able to learn driver specific experiences of comfort during a lane change, captured in weights of a comfort objective function. Learning is done by looking at observations performed by the human driver because these are carried out based on an internal unknown objective function to generate comfortable paths. The retrieved objective function after learning, will be an approximation of the real one used by the human agent.
Comfort features, which are integrated kinematic vehicle signals in a formulation so that they determine a notion of comfort, are used in order to quantify similarity between observed and learned vehicle paths. The contribution of this thesis is to implement the theoretical idea of feature-based learning with practical relevant vehicle models e.g. a 15 degrees of freedom Amesim model. When the comfort objective is identified, it can be embedded in a path planning formulation for autonomous driving.

In order to validate the developed algorithms for learning driver specific weights, data is generated with a known objective function from where the weights are tried to find back. Implementations are attained that make use of a non-linear bicycle model and a more complex 15 degrees of freedom Amesim model. Learning is done making use of gradient descent, wherefore an alternative method is proposed based on an optimized weight step size.

Results were retrieved that show that the lateral weights, that determine a lane change, were accurately learned. Due to a good match between learned and observed feature values, the observed kinematic vehicle signals were adequately learned. This research was performed with the help of Siemens Digital Industries Software.

Abstract

De menselijke bestuurder transformeert van een actieve naar een passieve deelnemer van het verkeer door ontwikkelingen die worden verwezenlijkt in onderzoek naar zelfrijdende auto's. Dit heeft als consequentie dat de rol van de automobilist beperkt wordt tot het waarnemen en beoordelen van handelingen, uitgevoerd door een zelfrijdend voertuig. Om consumentenacceptatie te bevorderen en de competitiviteit van het product te verhogen, is het belangrijk dat specifiek bestuurderscomfort gemodelleerd kan worden. Dit wil concreet zeggen, de afwegingen die gemaakt worden tussen de verschillende variabelen die bijdragen tot comfort.

Daarom is onderzoek uitgevoerd met als doel een algoritme te ontwikkelen dat gebruikt maakt van 'feature-based inverse reinforcement learning' dat in staat is om specifieke comfort bevindingen bevat in de gewichten van een comfortobjectieffunctie te leren. Het maneuver gekozen om dit te doen is een rijbaanwisseling. Het algoritme moet leren door te kijken naar observaties van het maneuver uitgevoerd door een menselijke bestuurder, omdat deze gebaseerd zijn op een interne, onbekende objectieffunctie. De objectieffunctie verkregen na het leerproces, benadert de werkelijke interne functie die gebruikt werd door de menselijke bestuurder.
Comfortfeatures, wat geïntegreerde voertuigsignalen zijn in een formulering zodat ze een notie van comfort bevatten, worden gebruikt om de overeenkomst uit te drukken tussen het geobserveerde en geleerde voertuigtraject. De bijdrage van deze thesis is het theoretische idee van feature gebaseerd leren te gebruiken met praktisch relevant voertuigmodellen zoals het 15 vrijheidsgraden bevattende Amesim model. Wanneer het comfortobjectief is geïdentificeerd, kan het gebruikt worden om trajecten te plannen voor een zelfrijdend voertuig.

Om de ontwikkelde algoritmes te valideren, is data gegenereerd met een op voorhand gekend objectief dat zal geprobeerd worden teruggevonden. Implementaties zijn verwezenlijkt die gebruik maken van een 'non-linear bicycle model' en een complex Amesim model. Tijdens het leerproces is de gradiënt afdaling methode gebruikt waarvoor een alternatief wordt voorgesteld dat gebaseerd is op een optimalisatie van de stap grootte. Resultaten werden bekomen die aantonen dat de laterale wegingsfactoren die bepalen hoe een rijbaanwisseling wordt uitgevoerd, accuraat geleerd kunnen worden. Door een goede match tussen de geleerde en geobserveerde featurewaarden werden de geobserveerde voertuigsignalen accuraat geleerd. Dit onderzoek is uitgevoerd met de hulp van Siemens Digital Industries Software.

List of Figures and Tables

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations

MS	Multiple shooting
SS	Single shooting
Gs	Gsteeringfactor
OCP	Optimal control problem
MPC	Model predictive control
ODE	Ordinary differential equation
ND	Number of datasets
dof	Degrees of freedom

Symbols

	Vectors are printed in bold.
N	The number of control actions performed during optimization Eq. (4.8).
N_{MPC}	Amount of control points in the control horizon of Eq. (5.1).
m	Amount of observed lane changes.
Z	Amount of features.
$\boldsymbol{r}$	The 2D path followed during a lane change maneuver.
$\boldsymbol{F}(\boldsymbol{r})$	Feature vector $\in \mathbb{R}^{Z\times 1}$ with on its entries the different feature values for learned path $\boldsymbol{r}$.
$\tilde{\boldsymbol{F}}(\boldsymbol{r})$	Observed feature vector $\in \mathbb{R}^{Z\times 1}$ with on its entries the different feature values for the observed path $\boldsymbol{r}$.
$\boldsymbol{F}_{diff}$	The difference between the observed and expected feature vector: $\tilde{F}(\boldsymbol{r}) - F(\boldsymbol{r})$.
V_0	Longitudinal speed at the start of the lane change.
V_{des}	Desired longitudinal speed during a lane change. Taken equal to V_0 during this thesis.
L	Desired lateral distance travelled at the end of the lane change.

$\boldsymbol{\theta}$	Weights that are learned in the objective function Eq. (4.12).
$\boldsymbol{f}_{obs}$	Feature vector calculated from observed lane change path according to Eq. (4.2.2).
$\boldsymbol{f}_{learned}$	Feature vector as calculated according to Eq. (4.2.2) from the lane change path following out of solving Eq. (4.8).
$\boldsymbol{f}_{rel}$	Relative feature vector defined as the element wise division of the learned feature vector by the observed one e.g. $\frac{\boldsymbol{f}_{learned}}{\boldsymbol{f}_{obs}}$.
t_r	Amount of throttle given.
δ	Angle of the front wheel of the bicycle model.
δ_s	The steer wheel angle, related to δ by G_s according to $\delta_s = Gs \cdot \delta$.
Δu	Update value used in the RPROP algorithm in section 3.2.2.

Chapter 1

Introduction

1.1 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] Out of investigation it followed that 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 is exactly comfort during driving and how can it be measured?
Driving comfort is a personal experience and also depends 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. [21] 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 a path planning algorithm.

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

1.2 Problem formulation and link with previous studies

In order to identify specific comfort preferences of the driver that are quantized by parameters, the vehicle should be able to learn by demonstration. [11]
Despite that each driver has its own preferences, they are based on a common notion of comfort where different trade-offs are made. For example, some drivers prefer more aggressive driving than others, which will be manifested in a different set of parameters then which will be attained for a defensive driver for the same comfort criteria. The comfort criteria will later in this thesis be translated into an objective function where different weights are used in order to quantify different comfort trade-offs made. This approach is in literature called inverse optimal control or inverse reinforcement learning because it is learning the objective function for an optimal control problem.

In order to learn the weights 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 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. Chapter 3 will give a more detailed description on the literature study conducted in order to investigate the state of the art comfort modelling. Here will also the theory behind inverse reinforcement learning be discussed.

1.3 Thesis objective and structure

The goal of this thesis is to build further on the research of learning by demonstration conducted in [11] and to refine this idea by making use of more realistic vehicle models. Concretely the thesis is focussed on the implementation and validation of an algorithm that learns weights in a comfort objective function which can explain observed driver data.
The learning process is done offline. After the different weights are identified, the learned objective can be used to plan paths which can be followed by an autonomous vehicle, making use of an online tracking MPC.

To conduct the inverse learning control there is looked at data generated by simulations where it is assumed that the vehicle is driving on an straight road on high

way speed. The maneuver investigated is a lane change maneuver as can be seen in Figure 1.2. In order to generate data, the non-linear bicycle and a more realistic 15 dof of freedom Amesim model is used.

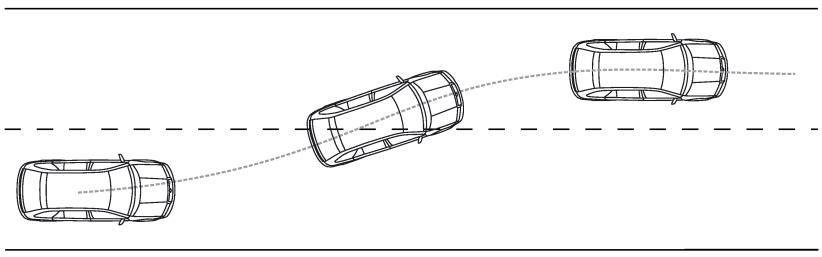


Figure 1.2: Example of a lane change which is the investigated maneuver in this thesis.

The execution of this research is conducted with the support of "Siemens Digital Industries Software" located in Leuven which made it possible to preserve a direct link with reality. Software was made available i.d. Simcenter Amesim.

The structure of the thesis is build up as follows. First the reader is made acquainted with optimal control concepts that will be used in chapter 2. In chapter 3 the state of the art of comfort modelling is discussed. This is split up in two parts. First the question what comfort during driving means, will be answered in section 3.1. The second part (section 3.2) concerns a discussion of the inverse reinforcement learning concept used. Chapter 4 gives insight in learning from ideal data. The non-linear bicycle model used is discussed in section 4.1 whereafter the formulation of the learning algorithm follows in section 4.2. Next the generation of ideal data is analysed and validated in section 4.3. After this different methods for learning form multiple datasets is discussed and results were analysed in section 4.4. In chapter 5 the non-linear bicycle model will be substituted by a more complex 15 degrees of freedom Amesim model. First the flow of the learning algorithm is discussed in section 5.1. Next the developed tracking MPC needed, is presented in section 5.2. Section 5.3 gives an overview of the learning results. In chapter 6 an enhancement on gradient descent is proposed in order to update the weights when going to the next learning iteration. Finally a general conclusion of the thesis follows in chapter 7.

Chapter 2

Background in optimal control problems

This chapter gives some background information on the theory behind optimal control problems (OCP) in section 2.1. After a global introduction and a discussion about discretization and time shooting methods, this chapter introduces the concept of model predictive control (MPC) in section 2.2.

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 input and state constraints." [12].

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

$\boldsymbol{q}$ 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 a system can be infinite, but in order to sufficiently model a system only a well chosen set of states is needed. Typical controls $\boldsymbol{u}$ for a vehicle are steerwheelangle and the amount of throttle which can be directly linked to the amount of propulsion force. Also higher order controls are possible.
The objective function l of the optimization problem Eq. (2.1) is integrating over the desired control horizon T. In the objective function it is indicated what should be minimized and this is in function of the different states and controls. $E(\boldsymbol{q}(T))$ describes the terminal cost and the 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 are retrieved by integrating this ODE over the control horizon. Q and U represent the feasible space wherein solutions for the OCP can be sought. It is worth noting that the control horizon time T itself can be an optimization variable. What comes out of a 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 spaces $\boldsymbol{q}(t) \in Q, \quad \boldsymbol{u}(t) \in U$. 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. [20]

2.1.1 Time discretization

The optimal control problem Eq. (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. Several ways to do this are available and summarized in Figure Eq. (2.1).[9]

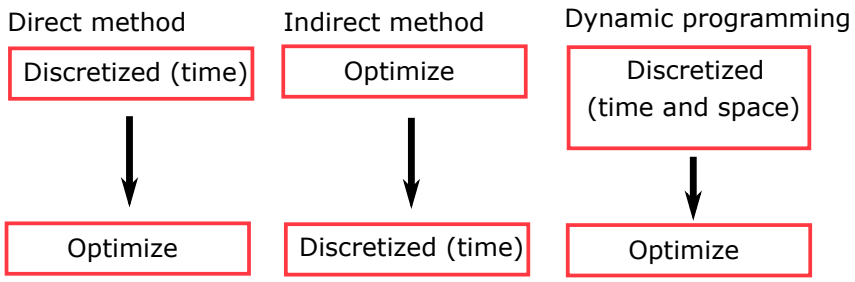


Figure 2.1: Overview of different discretization methods.

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 into account, is making use of spline based optimization formulations. [12]

Two different shooting approaches exist:

1. Multiple shooting (MS)
   During multiple shooting every new time sample new states and controls are introduced and taken as optimization variables. Control changes are only allowed on the different discrete time instances which leads to a piece wise control signal. The blue bars in Figure 2.2 (left) indicates the control signal 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}, time integration is embedded in $\boldsymbol{F}$ according to $\boldsymbol{q}^{k+1} = \boldsymbol{F}(\boldsymbol{q}^k, \boldsymbol{u}^k)$. These connections are put as constraints in the optimization and are called in literature 'path closing constraints' [9]. Eq. (2.2) presented the discretized version of Eq. (2.1) when using the multiple shooting approach.

$$
\begin{aligned}
\min_{\boldsymbol{q}(.),\boldsymbol{u}(.)} \quad & \sum_{k=0}^{N-1} \frac{l(\boldsymbol{q}^k, \boldsymbol{u}^k) + l(\boldsymbol{q}^{k+1}, \boldsymbol{u}^{k+1})}{2} \cdot \Delta T + E(\boldsymbol{q}^N) \\
\text{s.t.} \quad & \boldsymbol{q}^{k+1} = \boldsymbol{F}(\boldsymbol{q}^k, \boldsymbol{u}^k) \quad k = [0, \cdots, N-1] \\
& \boldsymbol{q}^0 = \boldsymbol{q}_{measured} \\
& \boldsymbol{q}^k \in Q, \qquad\qquad k = [0, \cdots, N] \\
& \boldsymbol{u}^k \in U \qquad\qquad k = [0, \cdots, N-1] \\
& \boldsymbol{q}^N \in Q_f, \quad N \in \mathbb{N}
\end{aligned}
\tag{2.2}
$$

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. The mathematical formulation of this is shown in Eq. (2.3). In Figure 2.2 (right) are the states that result from integration indicated in green. [9]

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

In this thesis a multiple shooting approach is used and a Runge-Kutta numerical integration scheme is implemented in order to define the path closing constraints. 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

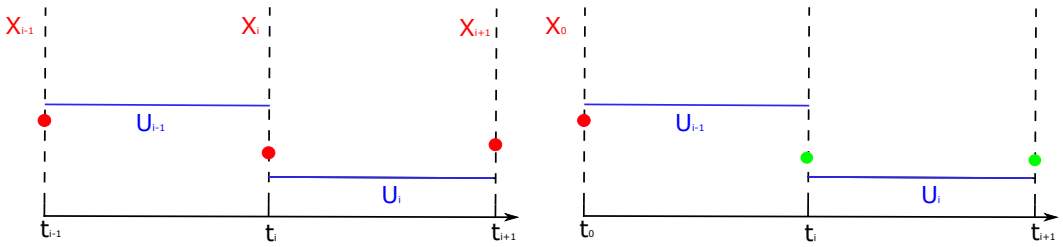


Figure 2.2: Schematic view of the time shooting approaches (left: multiple shooting, right: single shooting).

higher stability with respect to the chosen time step. [12]

It can be noted that 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 certain control. Therefore these matrices can be used more efficiently with respect to calculation load. In single shooting every state depends on the begin state and different controls which gives smaller, more densely 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 control technique. 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 given.

During MPC, optimizations are solved in a loop to be able to notice disturbances, changing environments and model-plant mismatch. Therefore it makes use of a moving control horizon[1]. Every iteration an OCP is solved starting from the current state and using a certain control horizon. From the resulting control signals only the first control value is applied for one time interval of the finite control 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 tradeoff 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 that only the first control of the computed control signal will be applied. This will induces that the system will change its current state.

[1]There are also other implementations e.g. shrinking time horizon, where the control horizon gets smaller every after each iteration.

Next a new OCP can be solved. A single OCP-iteration only takes the model of the system into account to calculate the control signals. The iterative way of solving makes the MPC able to deal with unexpected acquired states through feedback that is given by each new current state. As stated earlier the downside of this approach is a bigger computational load, which makes efficiently written software a necessity. [13]

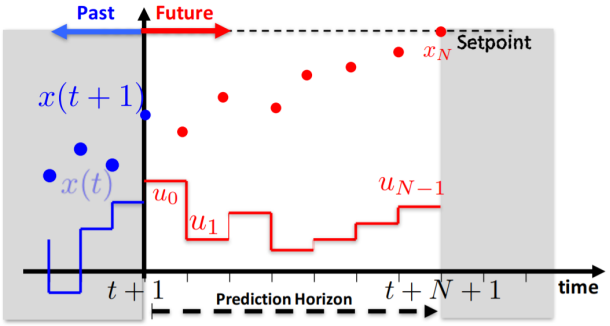


Figure 2.3: Visualization of the optimal control problem solved in one iteration of the MPC (Source: [13]).

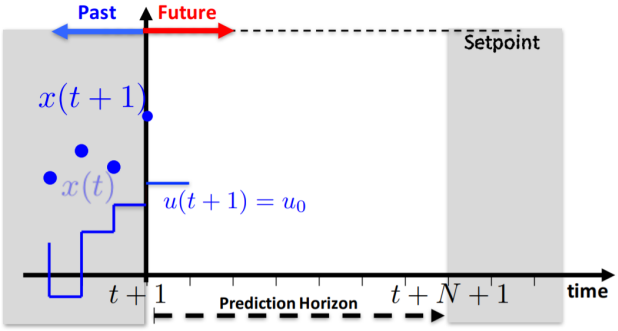


Figure 2.4: Visualization of the application of the first step of the calculated control signal during one iteration of the MPC (Source:[13]).

2.3 Conclusion

In this chapter the reader was made acquainted with concepts of optimal control and model predictive control, which will be used in this thesis. First a continuous OCP was considered and options how to discretize it were discussed. The option to discretize time best fitted for this thesis, is the "Direct method" together with a multiple shooting approach. It is made sure to take the time step small enough in order to avoid constraint violation.

Chapter 3

State of the art modelling of comfort

As discussed in the introduction, the goal of this thesis is to implement a method to capture personal experience of comfort during driving. This will be done by using an inverse reinforced learning where the weights in the objective function are learned from demonstration. To be able to do this, it is necessary that a literature study is done about how comfort is defined and to understand how to implement an inverse reinforced learning approach.

This chapter is structured as follows. First the question what comfort during driving means, will be answered in section 3.1. The second part (section 3.2) concerns a discussion of the inverse reinforcement learning concept used.

3.1 What is 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 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 additionally have to be taken into account 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 taken, is that motion sickness will be more prominent in autonomous driving due to the loss of control. Is is also argued that the distance to an obstacle is contributing to a comfortable feeling.

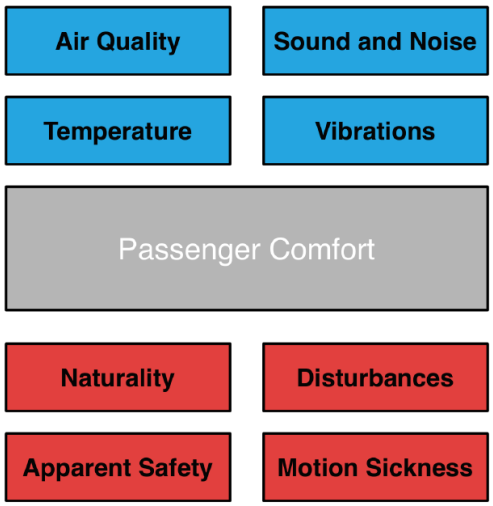


Figure 3.1: Overview of comfort parameters in autonomous vehicle with old parameters (blue) and new ones (red).(Source: [7])

In 'Analysis of Driving Style Preference in Automated Driving' [3] three studies were completed in order to capture the definition of comfortable driving in autonomous vehicles.
During the first study, human drivers drove manually with their own driving style a set of maneuvers and in this data, there was sought after relevant metrics that could be used in defining comfortable driving. The results of this research was that accelerations play a key role in comfort but are not the only factor. [3]

The different comfort metrics found were:

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

Quickness of the maneuver indicates that it is received as comfortable when the vehicle gives a good response and reaches the desired lateral displacement faster. Additionally comfortable driving as assessed in [3], can be summarized as sufficiently smooth trajectories and keeping enough headway distance in order to have a feeling of control and safety. These results suggest that when an algorithm is evaluating comfort, a notion of the surrounding traffic should be present. It was found when the traffic density on the road is higher, 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 the feeling of a fast response of the vehicle, translating in early peaks of acceleration. Vertical vibrations come not in the 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 retrieved. "Out of this it followed that accelerations are again playing a key factor." [3] For lane changes it could be concluded that maneuvers with a small lateral acceleration and early perceivable onset were more comfortable.

That also jerk plays an important role in the attained amount of comfort, is confirmed by [8] where it is stated that: "Jerk has been shown to elicit a stronger influence on comfort than acceleration".

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 besides lead to suboptimal solutions in comparison with from data learned parameters. Inverse reinforcement learning is the activity of learning an agents reward function from observations.

The goal of the learning algorithm explained in this thesis is to derive the parameters θ_j of different linearly combined comfort criteria f_j combined in the cost function J. When the parameters are learned, the objective function J will as best as possible explain the observed data. As the match with the observed data gets better during learning, it is assumed that J is getting closer to the inner comfort function of the individual driver. However, it should be noted that the driver takes unknowingly a lot of comfort criteria into account and they are not all linearly relating as is suggested by Eq. (3.1). Therefore the comfort cost function in Eq. (3.1), will always be an approximation of reality. Although as discussed in [11], it is possible to capture the magnitude of the main drivers that contribute to the comfort experienced by the users.

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

In order to create a generative model that creates vehicle paths $\boldsymbol{r}_i$ with equivalent kinematic characteristics as the path that was observed $\tilde{\boldsymbol{r}}_i$, 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 onto a scalar value and the difference between the demonstrated and calculated features give a clear indication about the similarity of the kinematic signals i.e. displacement,velocity, acceleration and jerk.

3.2.1 Feature based reinforcement learning

A feature value maps a signal onto a scalar and encapsulates a comfort criteria. The higher the scalar value the more discomfort is experienced by the driver. An example is given by Eq. (3.2) that measures the amount of accelerations in a maneuver.

$$f_j : \boldsymbol{r} \rightarrow f_j(\boldsymbol{r}) = \int_0^T a_x(t)^2 + a_y(t)^2 dt \tag{3.2}$$

The different feature values are collected in a feature vector $\boldsymbol{F} \in \mathbb{R}^Z$ with on its entries the different feature values f_j. The path that the centre of gravity of the vehicle is following can be represented by Eq. (3.3).

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

The human driver is not a deterministic agent and is modelled by a stochastic distribution $p(\boldsymbol{r}|\boldsymbol{\theta})$. For certain weights $\boldsymbol{\theta} \in \mathbb{R}^Z$ 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 Eq. (3.4). [4]. 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 cost function $\boldsymbol{\theta}^T \boldsymbol{F}(\boldsymbol{r})$ where agents are exponentially more likely to select trajectories with lower cost. [11] The observed feature vector $\tilde{\boldsymbol{F}} \in \mathbb{R}^Z$ has on its entries the different feature values $\tilde{f}_j$.
In order to explain the observed feature vector, the weights $\boldsymbol{\theta}$ need to be found that match the expected features vector $\boldsymbol{F}(\boldsymbol{r}_{expected})$ with $\tilde{\boldsymbol{F}}$. $\boldsymbol{r}_{expected}$ is defined as $E(p(\boldsymbol{r}|\boldsymbol{\theta}))$. To go towards a match with the observed feature vector, a gradient descent method can be used with an estimation of the gradient $\frac{\partial \boldsymbol{F}_{diff}}{\partial \boldsymbol{\theta}}$ by $\boldsymbol{F}_{obs} - \boldsymbol{F}(\boldsymbol{r}_{expected})$ and with $\boldsymbol{F}_{diff}$ the difference between the feature vectors. [4, 10]. Only the sign of the gradient is used and there are two directions for every weight update: increase or decrease. Therefore an intuitive explanation exist for using $\boldsymbol{F}_{obs} - \boldsymbol{F}(\boldsymbol{r}_{expected})$ as estimation for the gradient. The weight is increased when the expected feature is higher than the observed one, which means that the associated comfort feature will be punished more severely during the generation of $\boldsymbol{r}_{expected}$ in the next iterate. Consequently, the weight will be decreased when the expected feature is lower than the observed one. [11] Equation 3.5 summarizes the gradient descent method with α the step size taken in the direction of descent.

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

When $\boldsymbol{\theta}_{optimal}$ is found the gradient is minimized and the feature vectors will match as closely as possible. But the problem remains how to retrieve $\boldsymbol{F}(\boldsymbol{r}_{expected})$. In order to calculate $\boldsymbol{F}(\boldsymbol{r}_{expected})$ a Hamiltonian Markov chain Monte Carlo stochastic distribution sampling method was used in [10]. In [11] a simplified and less calculation

demanding approach is proposed, where it is assumed that the expected path is also the one that is been assessed as the most comfortable by the human driver. The path that is perceived as the most comfortable can be estimated by minimizing Eq. (3.1) for certain weights. Equation 3.6 summarizes the method proposed by [11].

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

$$\frac{\partial \boldsymbol{F}_{diff}}{\partial \boldsymbol{\theta}} = \boldsymbol{F}_{obs} - \boldsymbol{F}(\boldsymbol{r}_{expected}) \tag{3.6b}$$

3.2.2 RPROP algorithm

From [14] it is known that not every step size of the gradient is leading to 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 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. The update value Δu is solely dependent on the sign of the current gradient, the sign of the gradient in the previous iteration and the update value in the previous iteration. The three possible cases for the update value during iteration t is given by:

$$\Delta u_i^t = \begin{Bmatrix} \eta^+ \cdot \Delta u_i^{t-1}, & if\ \frac{\partial f_i}{\partial \theta_i}^t \cdot \frac{\partial f_i}{\partial \theta_i}^{t-1} > 0 \\ \eta^- \cdot \Delta u_i^{t-1}, & if\ \frac{\partial f_i}{\partial \theta_i}^t \cdot \frac{\partial f_i}{\partial \theta_i}^{t-1} < 0 \\ \Delta u_i^{t-1}, & if\ \frac{\partial f_i}{\partial \theta_i}^t \cdot \frac{\partial f_i}{\partial \theta_i}^{t-1} = 0 \end{Bmatrix} \tag{3.7}$$

$$0 < \eta^- < 1 < \eta^+ \tag{3.8}$$

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.

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

Exception on Eq. (3.9):

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

and with $\theta_i^{t+1} = \theta_i^t + \Delta\theta_i^t$, $i \in \mathbb{N}_{[1\cdots Z]}$ 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 Eq. (3.7) the step size is then reduced and in order to go back to the previous situation the update of the weight is done as indicated by Eq. (3.10). In order to not again decrease the update value when going back to the previous situation where the gradient will again change it's sign, $\frac{\partial f_i}{\partial \theta_i}^t$ is set to zero.
If the derivative retains its sign with respect to the previous iteration, the update value is slightly increased in order to accelerate convergence in shallow regions." [17] Parameters set by the user are $\Delta u_i^0, \Delta u_i^{max}, \Delta u_i^{min}, \eta^+$ and η^-. In this thesis following values were chosen: $\Delta u_i^0 = 0.1$, $\Delta u_i^{max} = 1.0$, $\Delta u_i^{min} = 10^{-7}$, $\eta^+ = 1.2$ and $\eta^- = 0.5$.

3.3 Conclusion

In order to find parameters of driving comfort that will be used as comfort criteria, a literature study was conducted which was mainly questionnaire based. The results were that in order to be able to evaluate comfort, higher order kinematic variables like accelerations and jerks must be considered. These variables are important to acquire 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 driving save be taken into account. Therefore comfort is naturally be influenced by the environment of the vehicle. It was found that when more traffic is present, a higher tolerance for less smooth trajectories is granted. Finally it was explained that if the goal of the maneuver was more rapidly completed this influenced the amount of comfort in a positive way.

In the second part of the literature study the concept of inverse reinforcement learning was clarified and it was explained that it is the process of identifying the unknown objective of the human driver. Next there has been looked into feature based learning and a practical method was found in order to retrieve $\boldsymbol{\theta}_{opti}$ making use of gradient descent. After this the RPROP algorithm was introduced that will be used to update the weights while making use of the gradient descent method in order to match expected feature values with the 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 $\boldsymbol{\theta}^T \boldsymbol{F}(\boldsymbol{r})$ as 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 and generating data. Thereby is the approximation that comfort of a driver is modelled by a linear relation of features is exactly fulfilled. Data is generated by choosing a set of weights in order to produce kinematic vehicle signals by minimizing Eq. (3.6). A validation of the developed learning algorithm is done when the chosen weights used in generating the data are found back.

First the non-linear bicycle model and the used parameters is explained in section 4.1. 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.e. neglects rolling and pitching and the position of the centre of gravity is fixed on the vehicle during driving. [20] Further on, the developed algorithm to learn the weight from demonstration is discussed in section 4.2. After that it is shown how the ideal data is generated and validated in section 4.3 and finally the learning results are analysed in section 4.4.
As noted before the learning process discussed, concerns an offline optimization which allows for higher calculations loads because of the absence of pushing real time constraints.

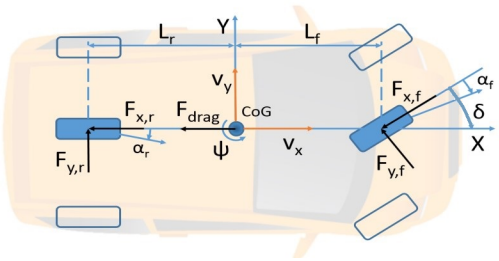


Figure 4.1: Non-linear bicycle model (Source: [18]).

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 are used.

1. Model has 6 states and 2 controls:

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

2. Model has 10 states and 2 controls:

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

x and y in the above formulation are the position of the centre of gravity of the vehicle 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 control vector of Eq. (4.1) consists of the throttle t_r and the angle of the front wheel δ. In the more extended bicycle model Eq. (4.2) throttle and front wheel angle 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 front wheel angle.

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 complete 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 tyre 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} 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 are 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 slip angles 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 approximate model in this thesis as the goal is to learn a comfortable and thus smooth lane change manoeuvre. However, these constraints will be checked during section 4.3.2.

The vehicle model parameters used are given in table 4.1. These numbers were provided by Siemens Digital Industries Software. The $Gsteerfactor$ approximates linearly the relation between the front wheel angle and the steer wheel angle turned by the driver: $\delta = \frac{\delta_s}{Gs}$.

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
Gsteerfactor 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 comfort objective function: $\boldsymbol{\theta}^T \boldsymbol{F}(\boldsymbol{r})$. Its formulation is presented in section 4.2.1. 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 displayed in Chapter 3 and on paper [11], the amount of discomfort can be modelled by the features discussed in section 4.2.2 during the timespan T of the maneuver. The scenario of a lane change on a highway is modelled. The time horizon itself is also taken as an optimization variable T.

4.2.1 Formulation of the learning algorithm

As has been discussed in chapter 3, $\boldsymbol{\theta}_{opti}$ gives the best possible fit between $\boldsymbol{F}(\boldsymbol{r}_{expected})$ and $\tilde{\boldsymbol{F}}(\boldsymbol{r})$. Without a vehicle mismatch, a match of feature values will induce a good match of the kinematic vehicle signals as will be further discussed in section 4.4. The flow of a single dataset learning algorithm can be seen in Figure 4.2.

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. From this $\boldsymbol{F}(\boldsymbol{r}_{expected})$ can be retrieved by using the definition discussed in section 4.2.2. Afterwards, the relative features $f_{rel,i}$ are calculated by element wise divide the expected feature vector by the observed one $\tilde{\boldsymbol{F}}$. 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 estimation of the gradient and the RPROP algorithm explained in Chapter 3. The

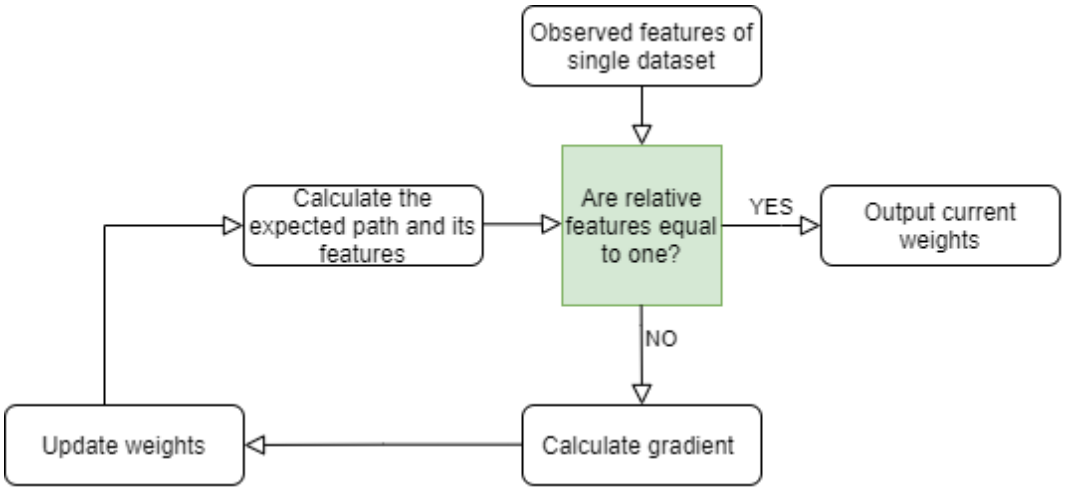


Figure 4.2: Basic flow of the reinforced learning algorithm.

new weights are used to calculate a new expected path which will result in different expected feature values $\boldsymbol{F}(\boldsymbol{r}_{expected})$. Next a more detailed description of the calculation of the expected path is given by Eq. (4.8) which uses as vehicle model Eq. (4.2).

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

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 of Eq.(4.2) during the maneuver. The time of the maneuver is represented as 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 determines the amount of controls applied to go from the initial state towards the end state. 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 Eq. (4.3) are inputted in the optimization because derivatives of the vehicle states are needed. The time discretization used can be categorized as a direct method and because a multiple shooting approach is used at every time instance a new set of state optimization variables is introduced as is

explained in section 2.1.1. The path constrain vector $\boldsymbol{F}$ demarcates together with the equality constraint vector $\boldsymbol{H}$, the feasible space of the solutions for $\boldsymbol{X}$ and $\boldsymbol{U}$. An overview of the constraints used, is given by Eq. (4.9) and Eq. (4.10).

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

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

The constraints displayed in $\boldsymbol{H}$ make sure that at the end of the lane change the slip angles in the tires and the steer wheel angle are zero. From Eq. (4.3) this induces that the lateral velocity, acceleration and yaw acceleration also become zero and this marks the end of the lane change. y^N makes sure that the wanted lateral distance is covered. This distance can be calculated from the start position of the vehicle and the width of the lane in order to end up at the centre line of the desired lane.
At the start of the lane change straight driving at constant longitudinal speed is assumed. To achieve this, the constraints of Eq. (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 Eq. (4.3). In Eq. (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 for chosen weights.

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

The time limit constraint in Eq. (4.8) is needed in order to demarcate the optimization solution space. When set, it has to take two conflicting criteria into account. It has to be chosen large enough in order to have a minor influence on the lane change behaviour introduced by this constraint. Secondly it has to be taken small enough in order to preserve good conditions for the numerical integration performed inside $\boldsymbol{F}(\boldsymbol{X}, \boldsymbol{U}, T)$ as will be seen in in 4.2.2. This is because the number of optimization points $N + 1$ of the states is fixed, which means that a larger time limit will give a coarser time discretization. In this thesis the time limits used are 25 s and 30 s, which is a compromise between the two criteria. 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 s. This is not an realistic results because the objective will, as previously explained, not have good numerical properties.

In order to solve Eq. (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 Eq. (4.6) this would give a division by zero in the calculation of the slip angles.
To further enhance the solving speed of Eq. (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 because IPOPT is an interior point method. Therefore the initial guesses for $\boldsymbol{X}, \boldsymbol{U}$ *and* T are taken from the observed ideal data.

Another way to speed up the solving time of the IPOPT solver, is setting the initial guess of the lambda multipliers internally used, equal to the ones found during the previous call of Eq. (4.8) during the loop visualized by figure 4.2.
The time needed for the CPU to calculate the expected path for a certain set of weights and thus solving Eq. (4.8), takes around 5 s (python implementation) when a time limit of 30 seconds and N equal to 1000 is chosen.

As discussed above the solver used to calculate the states and control signals in Eq. (4.8), is 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]

4.2.2 Objective function

The objective function used in Eq. (4.8) has to represent 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 will be learned in. It can be expected that the linear relation of features given by $\boldsymbol{\theta}^T \boldsymbol{F}(\boldsymbol{r})$ will serve as an approximation for the real, more complex comfort objective of a human drive. The feature framework that is further discussed in this section, can be validated and adjusted based on an user study.

$$discomfort = \theta_1 \cdot f_1 + \theta_2 \cdot f_2 + \theta_3 \cdot f_3 + \theta_4 \cdot f_4 + \theta_5 \cdot f_5 + \theta_6 \cdot f_6 \tag{4.12}$$
$$f_i, \theta_i \in \mathbb{R} \quad i \in \mathbb{N}$$

Feature 1: longitudinal acceleration

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

Feature one is assessing the amount of discomfort by integrating the total longitudinal acceleration in the local axis. The local axis is 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 Eq. (4.3).

Feature 2: lateral acceleration

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

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 Eq. (4.3).

Feature 3: longitudinal jerk

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

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

Feature 4: lateral jerk

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

Feature four is given 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

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

L is a constant and set equal to the desired lateral distance. If the vehicle reaches its desired lateral displacement faster, this is perceived as a good response and is interpreted as comfort as is discussed in section 3.1.

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

$$\int_{f_j(t^n)}^{f_j(t^{n+1})} df_j = \int_{t^n}^{t^{n+1}} P(t) \cdot dt \tag{4.19a}$$

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

To summarize, the objective described by Eq. (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 Eq. (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 contributes to the amount of comfort that is experienced. 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 Eq. (4.12). This can be done if data of the position of other vehicles during the maneuver is available. Paper [11] gives some suggestions showed by Eq. (4.20) and Eq. (4.21).

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

With $[x_o, y_o]_k$ the position of the closest point of a different agent and NA the total amount of road agents in the nearby area.

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

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

The minimum desired following distance $\hat{d}$ can be calculated based on the distance needed to avoid collision during unexpected events when driving at 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 with a constraint time horizon can be planned for a specific driver.

Normalization factors

To reduce the effect of order of magnitude given by the units in the objective, a normalization of the features is done. 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 as can be seen in Eq. (4.22). Table 4.2 gives an overview of the lane change normalization factors used.

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.

Because of the normalization the relative weights defined as $\theta_{r,i} = \frac{\theta_{abs,i}}{norm_i}$, will quantify the trade-offs between different comfort features without disturbance of units used. Aside of this it allows to learn weights faster, because of the absence of big size differences induced by unit differences that are present in the absolute weights.

During the learning loop (Figure 4.2) the max weight update is set to a maximum of 1 in this thesis. When the absolute weights are learned instead of the relative ones, this would give a substantially higher amount of iterations when started from an initial guess an all-one vector.

4.3 Ideal data

As mentioned at the begin of this chapter the term 'ideal data' concerns data that is generated with a non-linear bicycle model and with a beforehand known objective function Eq. (4.12) which consist out of a linear combination of features and weights $\boldsymbol{\theta}$. This means that the algorithm that learns the weights can be validated by checking if it is able to find back the correct weights. First the generation of ideal data is presented in section 4.3.1 whereafter it is validated in section 4.3.2.

4.3.1 Generation

The relative weights chosen to generate the data 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 discussed in section 4.2.2 is taken into account. The objective that is used in Eq. (4.8) is given by Eq. (4.22). The different feature values are the ones as defined in section 4.2.2.

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

When multiple ideal datasets have to be generated in order to serve as observations, the initial speed V_0 and width of the lateral distance L 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 control points, use of linear tire model and a discussion on 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 Eq. (4.1) and Eq. (4.2). Variant one has only 6 states and the accelerations and jerks in the objective function Eq. (4.12) are calculated by making use of numerical differentiation described by Eq. (4.23) from the other vehicle states. Variant two on the other hand has the total accelerations as direct states in the vehicle model and uses an analytical formulation of the jerk described in appendix A.

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

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

In order to validate the two approaches the generated lateral jerk signals are compared in Figure 4.3.

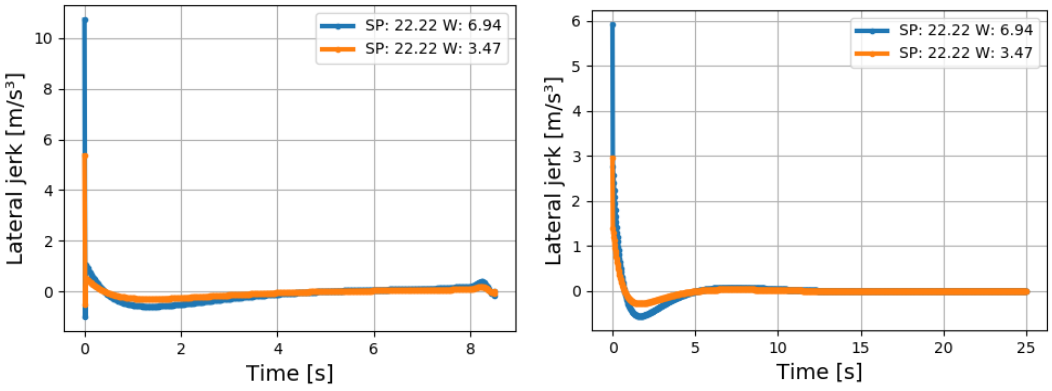


Figure 4.3: A comparison between the numerical jerk (left) based on Eq. (4.23) and Eq. (4.1) and the analytical jerk (right) based on appendix A and Eq. (4.2).

It is clear that the analytical formulation (right figure) of the jerk gives more smooth results and less high peaks which is in line with observations of the more complex 15 dof Amesim 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 i.e. the equations resulting from numerical differentiation are approximations of the analytical ones.

Initial guess

To determine if the generated data is a local solution, two different initial guesses $V0 : 22.22 - L3.47$ and $V0 : 25.00 - L6.94$ are used whereof the retrieved data feature values are summarized in table 4.3. V_0 is the initial speed and L the lateral distance of the lane change used as initial guess and are generated using the above described numerical formulation. In order to calculate the data features displayed, time limit is set on 30 s, N on 1000, $V_0 = 22.22\ \frac{m}{s}$ and $L = 3.47\ m$ as parameters in Eq. (4.8) and the analytical formulation is used. From the results it is suggested that the generated data feature values are not a local solution because they are found back from different start points of the optimization. During the learning of the weights as described in Figure 4.2 the initial guess is set equal to the observed data.

Time limit

In order to check the dependency of the generated data on the chosen T_{limit} constraint in Eq. (4.8), data is generated for a lane change with N, the amount of control points equal to 1000, initial velocity equal to 80 $\frac{km}{h}$ (V_0), a desired lateral displacement of

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 the two different initial guesses in Eq. (4.8).

3.47 m (L) and a varying T_{limit} constraint as indicated in table 4.4. Figures that show what the difference in feature values actually means for the different kinematic signals of the vehicle, 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 Eq. (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 which will result in an more unreliable discretization.

From the result of table 4.4 it can be concluded that the influence of the manually setting of the time limit in Eq. (4.8), can be neglected. The lateral features that are the most important ones as will be discussed in section 4.3.2 are very similar and also the kinematic signals for the lateral variables found in Appendix B show the same behaviour.

Amount of control points

The test carried out to investigate the dependence of the resulting feature values on the amount of control points N, 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 chosen equal to 1500. For a full overview of the different kinematic signals,

reference is made to Appendix B. Again the kinematic vehicle signals confirm the same conclusion as taken from table 4.5.

Feature Value	500	1000	1500
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 Eq. (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 ($a_y <= 4\frac{m}{s^2}$) and slip angles ($\alpha <= 5^o$) during the maneuver. Figure 4.4 gives the total lateral acceleration during a lane change maneuver that moves 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 embedded in Eq. (4.8). Both figures are generated with the complex vehicle model discussed in chapter 5 and make use of Eq. 4.6 in order to estimate the slip angle.

Feature values

In the above tables it can be seen that the first, third and fifth feature values concerning longitudinal behaviour of the vehicle are very small. This often means that during a lane change these features have a low influence. It can be expected that for small observed feature values, it becomes hard to accurately learn weights that generate feature values that match with these small values. The reason for this is because the feature value is negligible which means that any weight times zero stays zero in the objective function of Eq. 4.8. For this reason almost any weight can be used for the longitudinal features without observing a different lane change outputted by the optimization done in Eq. 4.8. This will be further discussed in section 4.4.1.

An other interesting observation about the generated data features is that when the initial speed of the lane change is varied and the desired lateral displacement stays the same, because of the small longitudinal feature values the data features found are almost identical.

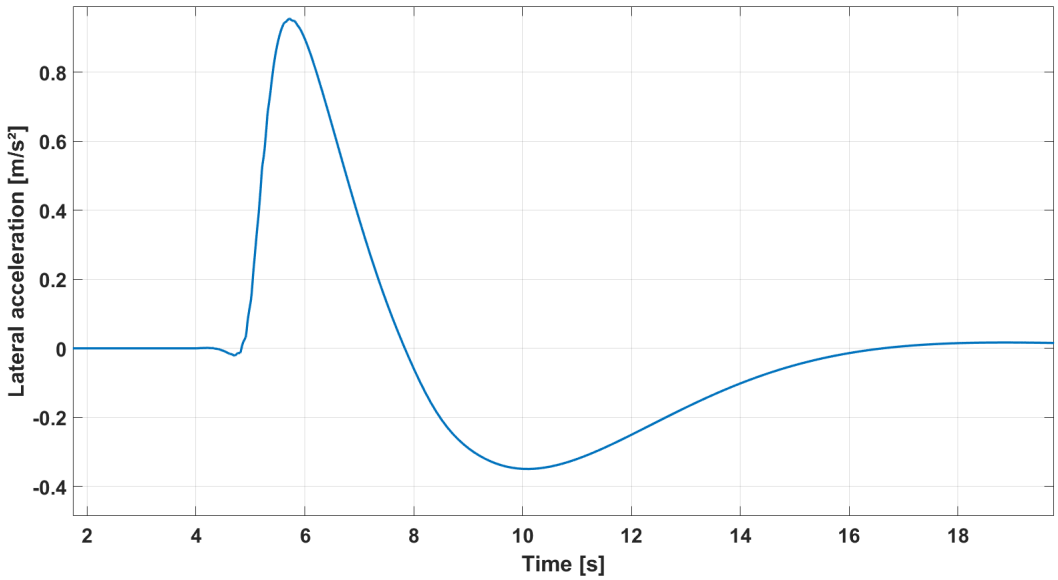


Figure 4.4: Lateral acceleration during a lane change $V_0 : 25.00\frac{m}{s}$ and $L : 6.94m$ generated with the 15 dof Amesim model.

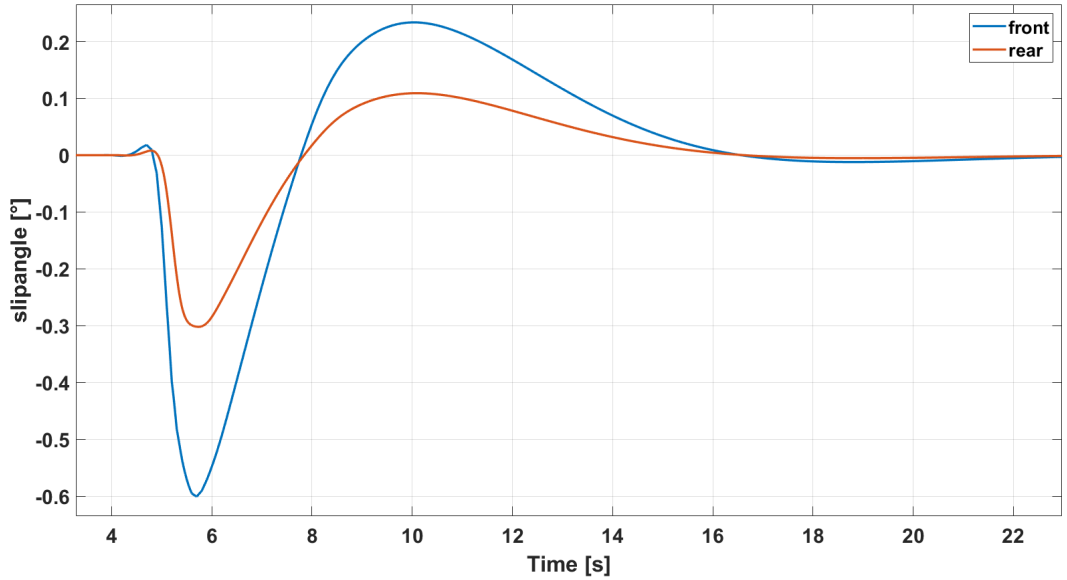


Figure 4.5: slip angle during lane change $V_0 : 25.00\frac{m}{s}$ and $L : 6.94m$ generated with the 15 dof Amesim model.(blue:front,red:rear)

Conclusion

In this section there has been looked into the correctness of the generated data making use of Eq. (4.2), that further will be used to learn from. The influence of different choices i.e. initial guess, the time limit and the amount of control points

has been checked. Out of this it followed that the same data features were found for different intial guesses and it is allowed to take the time limit equal to 30 s and the amount of control points to 1000 as is done further on in this thesis. Next it has been shown that it is justified to use a linearised tire model. Finally the resulting feature values of the ideal data were shortly look into. It was concluded that a worse learning is expected for the retrieved longitudinal features due to their small values.

4.4 Ideal data learning results

In this section the results of learning weights from ideal data is discussed. Section 4.4.1 concerns the learning of a single dataset $V_0 : 22.22\ \frac{m}{s}$, $L : 3.47\ m$. In reality one demonstrated demonstration doesn't capture perfectly the preference of a human driver. In order to do so, learning of multiple datasets is needed. Therefore the averaging method is presented in section 4.4.2 and the conflict method in section 4.4.3. Afterwards a comparison is made in section 4.4.4.

It is known beforehand that the weights used to generate the data feature values 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 loop as displayed in Figure 4.2, is when the maximum amount of iterations set to 300 is reached or if the data feature values are accurately regenerated with learned weights.
This is quantified by $f_{rel,i} = \frac{f_{learned,i}}{f_{obs,i}} \leq 10^{-3}$. $\boldsymbol{f}_{obs}$ equals the generated data feature vector retrieved according to section 4.3 and is constant during learning. $\boldsymbol{f}_{learned}$ is the feature vector calculated from the learned path and changes during every learning loop. Convergence is reached when the three lateral feature values 2, 4 and 6, that dominantly define the lane change, are accurately fitted for a certain set of weights. The simulations done in this thesis are performed on a notebook provided by Siemens with Intel Core i7-7920HQ CPU @ 3.10GHz and 32 GB of RAM memory.

4.4.1 Single dataset learning

The generated data features that have to be matched, can be seen in table 4.3. The resulting weight vector $\boldsymbol{\theta}$ and $\boldsymbol{f}_{rel}$ outputted at convergence on iteration 28 is displayed in table 4.6. The weight concerning the lateral acceleration (Nr.2) is taken as reference in order to compare the learned weights with the chosen ones. The results show that the lateral weights are found accurately back. Table 4.6 also shows the results when the algorithm is manually set to run for 121 iterations. A clear difference between the lateral features that have an increased matching of $f_{rel,i}$ with an accuracy of 10^{-6} is seen. The convergence of the longitudinal features $f_{rel,i}$ towards one, didn't increase much because they are constraint in the accuracy that they can be learned.

[2] The corresponding features of the chosen weights can be seen in section 4.2.2 and are summarized by $\left[f_{ax},\ f_{ay},\ f_{jx},\ f_{jy},\ f_{diffvx},\ f_{diffy}\right]$.

	$It.28 - \boldsymbol{\theta}$	$It.28 - \boldsymbol{f_{rel}}$	$It.121 - \boldsymbol{\theta}$	$It.121 - \boldsymbol{f_{rel}}$
Nr.1	14.469	0.9907	10.021	1.0004
Nr.2	5.000	0.9995	5.000	1.0000
Nr.3	3.045	1.0037	2.179	0.9976
Nr.4	5.977	1.0006	5.998	1.0000
Nr.5	3.689	0.9941	2.538	0.9892
Nr.6	1.998	1.0001	2.002	1.0000

Table 4.6: This table shows the weights learned from the dataset $V_0 : 22.22\frac{m}{s} - L : 3.47m$ and the associated $\boldsymbol{f}_{rel}$ at a two different amount of iterations.

As already suggested in section 4.3.2 this is because of the small size of the feature values of the longitudinal direction. Not very accurate learned longitudinal feature values will have a negligible influence on the overall behaviour of the planned path and as can be seen, a range of weights will give an acceptable match quantified by $f_{rel,i}$.
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 in defining the lane change, they have to be removed from the objective Eq. (4.12). This is however not an correct assumption. The longitudinal features are less dominant during a lane change and therefore different longitudinal 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 $f_{rel,i} = \frac{f_{learned,i}}{f_{obs,i}} \leq 10^{-3}$. However, instead of inputting a data feature vector $\boldsymbol{f}_{obs}$ based on a single dataset, an averaged one over multiple datasets is used.
To calculate the gradient used to update the weights $\boldsymbol{\theta}$, the difference between the averaged data feature vector and the averaged learned feature vector is taken. In order to obtain the averaged learned feature vector, m times the optimization Eq. (4.8) is called with m the amount of observed maneuvers. After solving the m resulting learned feature vectors they are averaged. Each distinct maneuver has a different initial longitudinal speed and desired lateral displacement.

The datasets chosen to perform the learning 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 weights 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 $\left[0.9981,\ 0.9996,\ 0.9829,\ 1.0000,\ 0.9946,\ 1.0001\right]$. This means that the lateral chosen weights are accurately found back. The $\boldsymbol{f}_{rel}$ of the individual datasets can be seen in table 4.7. The individual $\boldsymbol{f}_{rel,i}$ at convergence shows how good the match of feature values is when applying the learned weights at Eq. (4.8) with the same parameters for V_0 and L. It is shown that there is a good match between the learned feature vectors and the individual data feature vectors.

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 4.7: 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 brown. The finally learned paths can be seen in pink, grey and yellow. These paths lay exactly on the observed ones. From this it is concluded that matching of feature values which are scalars, give a good representation for the matching performance of the 2D kinematic vehicle signals. The rest of the kinematic signals originating from the non-linear bicycle model can be seen in Appendix C.

The progress towards convergence over the iterations is showed in Figure 4.7. Convergence is reached in 25 iterations.

Figure 4.8 gives a view of how the learned weights change over the iterations for the different features. The weights in these graphs differ with the ones shown at the beginning of this section by a scaling factor $\frac{5.0}{\theta_2}$. Both these weights will generate the exact same lane change when used in the objective of Eq. (4.8). During the learning process this degree of freedom can be removed by fixing the second weight equal to 5.000. Convergence is reached after 59 iterations and following weights are outputted $\left[15.143,\ 5.000,\ 3.222,\ 6.016,\ 3.880,\ 2.007\right]$ where the lateral weights again clearly are found back. It takes the algorithm longer to find equivalent weights because of the lost in the ability of the RPROP algorithm to update all the weights. Therefore the scaling degree of freedom is retained during the rest of this thesis.

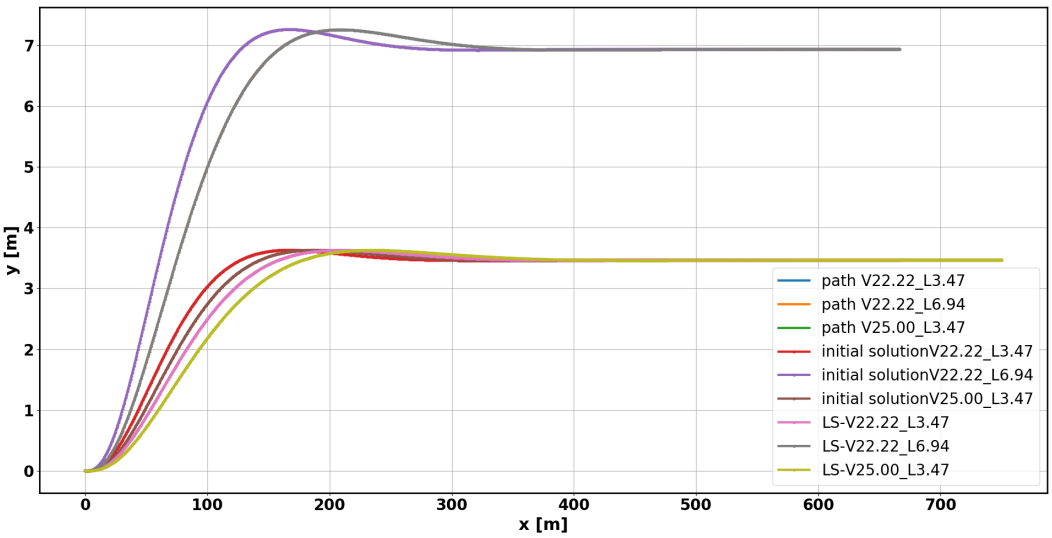


Figure 4.6: Observed, initial and learned paths for 3 different observed lane changes generated with common underlying objective function.

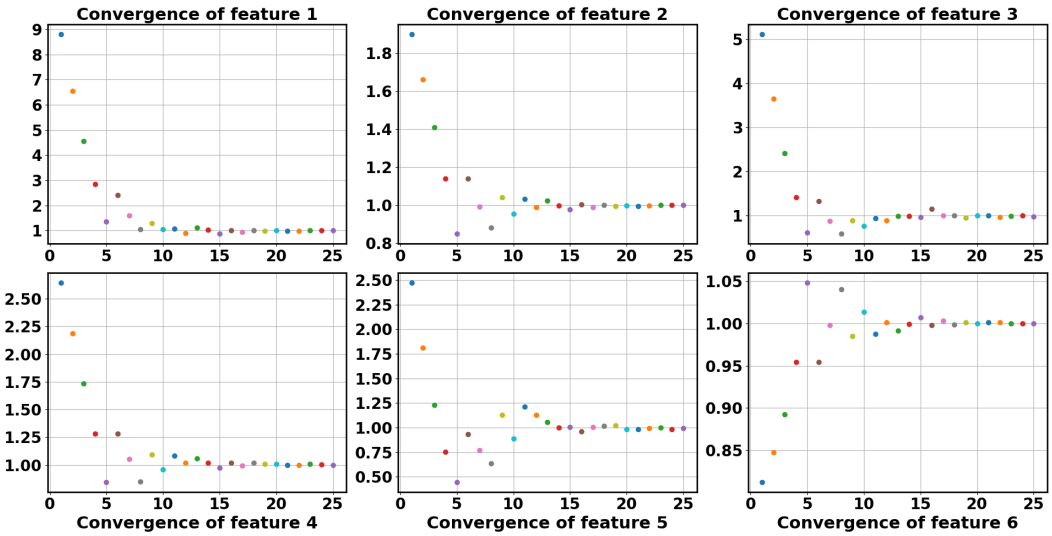


Figure 4.7: The evolvement of the average $\boldsymbol{f}_{rel}$ over the learning iterations.

4.4.3 Conflict method

The second method proposed to learn simultaneous of multiple datasets, is the conflict method. Its main idea is to only update a weight if the gradient, as given by Eq. (3.6b), of the different individual datasets point all in the same direction. If there are individual gradients with conflicting signs, the update direction is ambiguous and the concerning weight is not updated. The updating will be resumed when the conflict is solved by updating the other weights. Solving conflicts is possible because

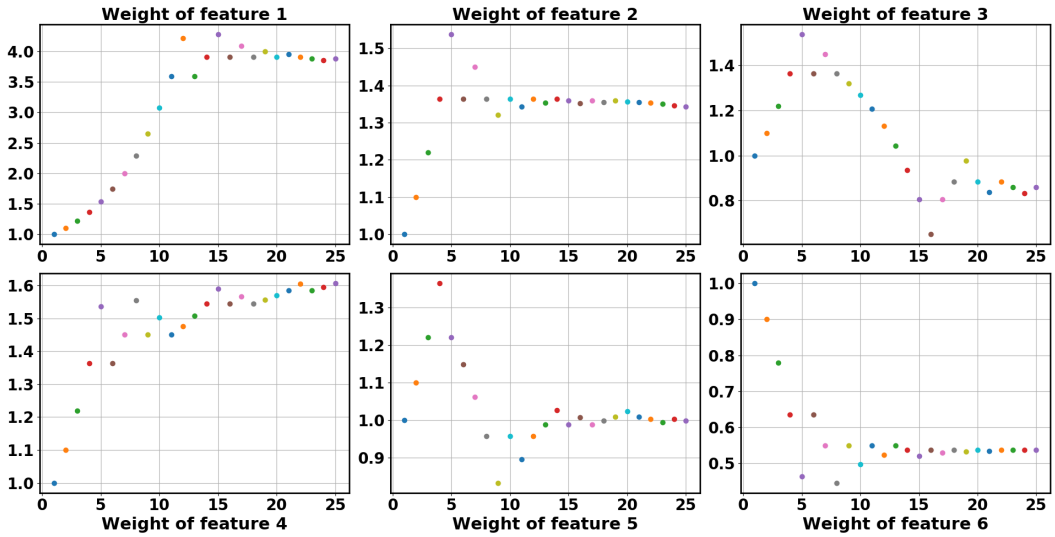


Figure 4.8: The different outputted weights over the learning iterations.

the features are not independent of each other. Figure 4.9 shows how the conflict check is integrated together with the RPROP algorithm in the 'update weights' box in the basic learning flow of Figure 4.2. The blue 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 (dw) and the exception value as discussed in section 3.2.2. Inside the conflict block, it is checked if all the signs of the current gradients are consistent. This boils down to verifying on which entry in the m different gradients there is a sign difference. Here is m the amount of observations and an individual gradient is calculated as $\boldsymbol{F}_{obs,i} - \boldsymbol{F}(\boldsymbol{r}_{expected,i})$ with $m \in \mathbb{N}_{[1...ND]}$.

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). 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, because no conflict was spotted in the gradients during the previous iteration. When a conflict is resolved, the next case will be automatically equal to RPROP case 3 because no decision can be made to either increase or decrease 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

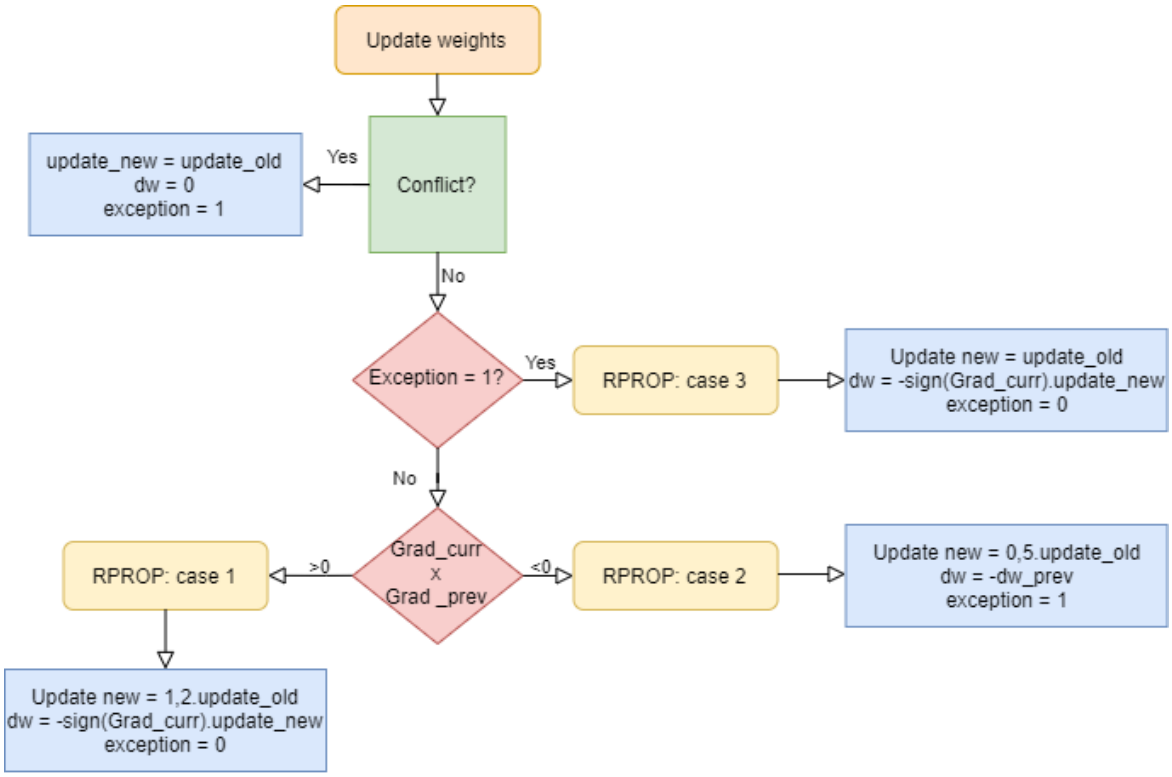


Figure 4.9: Flow of the conflict method as part of the basic flow diagram of Figure 4.2

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. It can be concluded that the lateral weights are found back. The $\boldsymbol{f}_{rel}$ of the individual datasets can be seen in table 4.8. It is shown in the table that there is a good match between the learned feature vectors and the individual data feature vectors. Figure 4.10 presents 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 for the other datasets are very similar.

As can be seen in table 4.8 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 real driver data is used where the human driver is less consistent in producing observations that resemble its underlying objective, the algorithm is stopped earlier. This will naturally constrict the algorithm in its feature matching accuracy due to no available direction of improvement. This can already

$\boldsymbol{f}_{rel}$	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 4.8: This table shows the $\boldsymbol{f_{rel}}$ for each individual dataset at convergence using the conflict method.

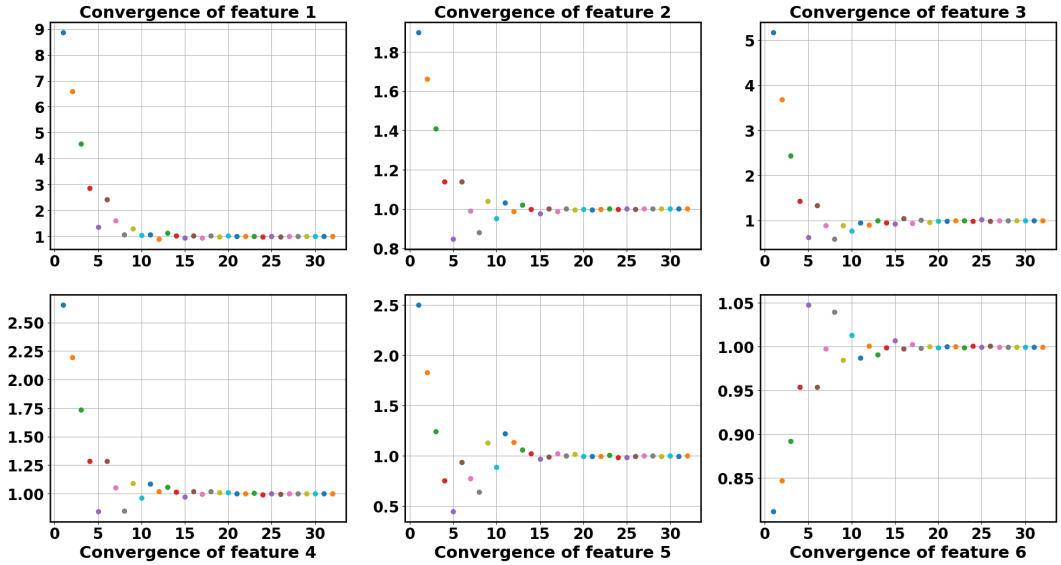


Figure 4.10: The evolvement of $\boldsymbol{f}_{rel}$ of dataset V_0 : 22.22 $\frac{m}{s}$, L : 3.47 m over the learning iterations.

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 improvement. 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 $\boldsymbol{f}_{rel,i}$ vector makes with respect to convergence to one for the three important lateral features. The three important lateral features shown in Figure 4.11 are respectively the remaining lateral distance, lateral acceleration and lateral jerk. ND stands for number of datasets used.

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

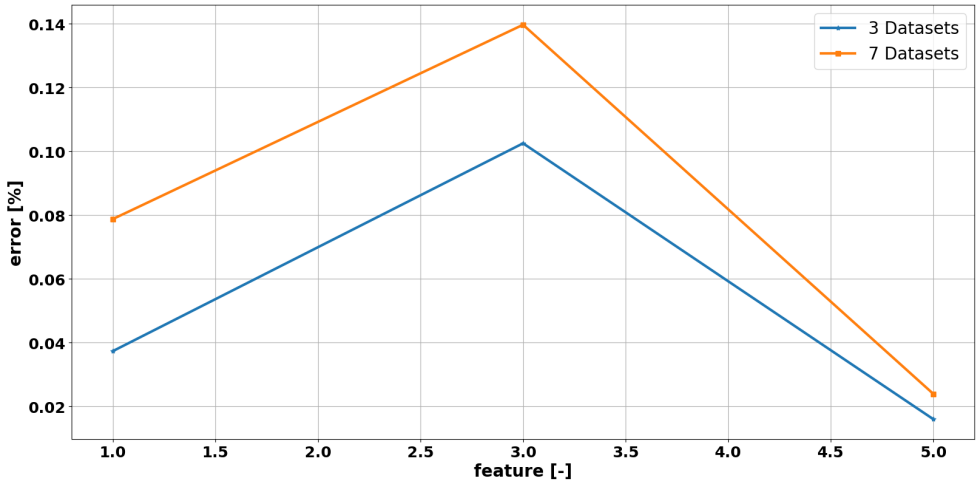


Figure 4.11: This figure shows the averaged, relative error given by $\frac{100 \cdot \sum_{n=1}^{ND} |1 - f_{rel,i}|}{ND}$ of the matching of the feature values for the three lateral features. (3 datasets: blue, 7 datasets: orange)

4.4.4 Comparison of methods

In this section a comparison is made between the averaging and conflict method of combining multiple datasets. Figure 4.12 shows that the averaging method learns weights that can better explain the individual data feature vectors of the different datasets. Therefore this will be the method further used during this thesis.

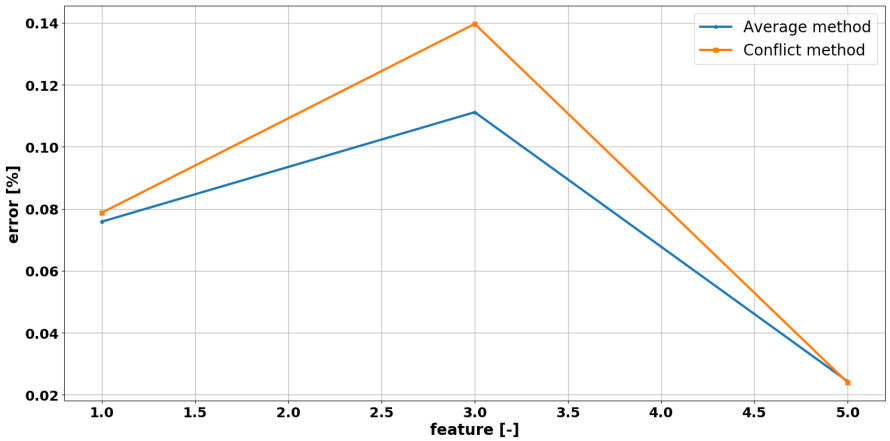


Figure 4.12: This figure shows the averaged, relative error given by $\frac{100 \cdot \sum_{n=1}^{ND} |1 - f_{rel,i}|}{ND}$ of the matching of the feature values of the three lateral features. (Averaging method: blue, conflict method: orange)

4.5 Conclusion

In this chapter the implementation of learning from ideal data was analysed. Data is considered ideal when it is produced with the same vehicle model as is used during the learning loop which calls the optimization Eq. (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 attained during a lane change. Contrary to the lateral weights, the longitudinal chosen weights were not accurately found back. Because the conflict method gives for the same amount of datasets a slightly larger error on the ability to explain the individual 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 $\frac{\partial \boldsymbol{F}_{diff}}{\partial \boldsymbol{\theta}}$ 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 with 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 test car they are currently using. The model has as inputs the amount of throttle, braking and steerwheelangle.

First it will be explained in section 5.1 how the Amesim model will be integrated into the learning of the different weights. Next, a tracking MPC needed for the implementation will be discussed in section 5.2. Afterwards the learning results will be discussed in section 5.3.

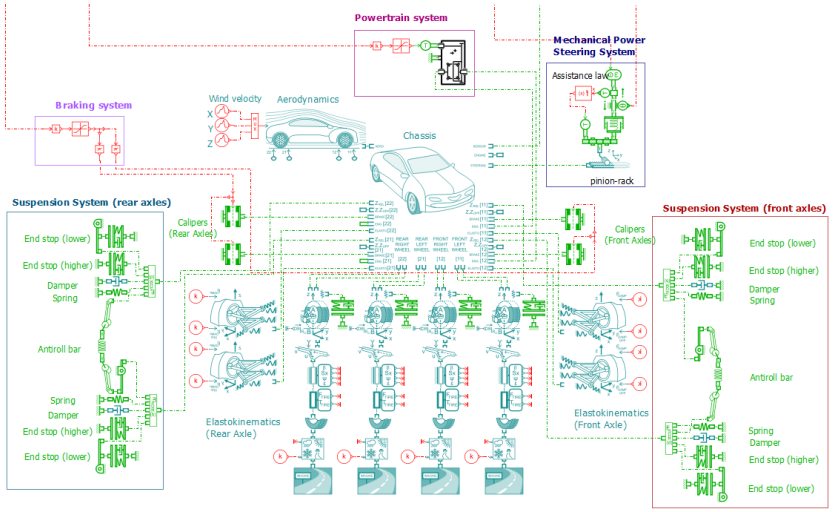


Figure 5.1: The 15 dof amesim model with as inputs the amount of throttle, braking and steerwheelangle.

5.1 Flow of the algorithm

The Amesim model serves as a black box and no explicit dynamic equations are available to include in Eq. (4.8) as was done with the non-linear bicycle model Eq. (4.3). Therefore a path is planned with the simpler non-linear bicycle model and afterwards a model predictive control tracking algorithm is applied with the Amesim model in order to follow the calculated reference. The working principles of a MPC is discussed in 2.2. Diagram 5.2 shows the flow of handlings that is done during the learning of the weights with the Amesim model.

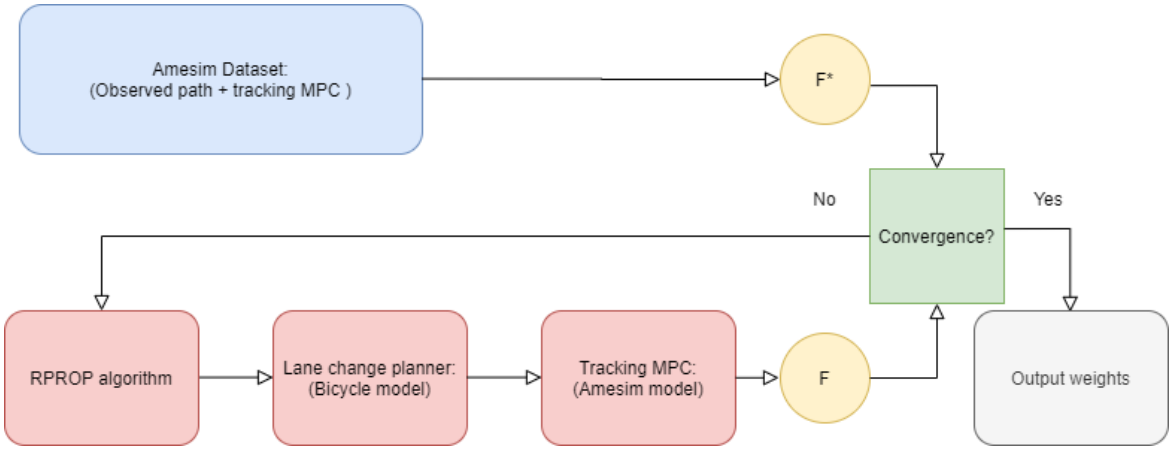


Figure 5.2: The flow of learning with the Amesim model.

As is described in Diagram 5.2 the feature vector $\boldsymbol{F}(\boldsymbol{r})$ is calculated based on a path tracked by the Amesim model and it is used in the convergence block to estimate the gradient $\frac{\partial \boldsymbol{F}_{diff}}{\partial \boldsymbol{\theta}}$. The estimation is calculated by $\boldsymbol{F}^* - \boldsymbol{F}(\boldsymbol{r}_{expected})$. The reason why the path planned by the bicycle model has to be tracked is to avoid including vehicle mismatch error in the learning process. It is otherwise possible that even when the algorithm is able 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 that were 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 are reasonably preserved when a trajectory is tracked by the Amesim model. However although the feature values are not that far apart, not all kinematic signals of the bicycle and Amesim model will match accurately as can be seen in Appendix D. This will further be discussed in section 5.2.

[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.

This chapter also gives an answer on the question if the estimate of the gradient $\frac{\partial \boldsymbol{F}_{diff}}{\partial \boldsymbol{\theta}}$ by $\boldsymbol{F}^*(\boldsymbol{r}) - \boldsymbol{F}(\boldsymbol{r}_{expected})$ is still sufficient in order to match the learned and observed feature values in section 5.3.

5.2 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 $j_y(t)$. The structure of this section is build up as follows. First the MPC formulation is discussed in section 5.2.1 and the tracking results are discussed in section 5.2.2.

5.2.1 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 Eq. (4.2), inside the OCP formulation given by Eq. (5.1). Eq. (5.1) uses as objective an error function with respect to the reference what will assure a 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 table 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 of the bicycle model. The path constraints can be seen in Eq. (5.2). These 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.

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

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

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.

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

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

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 Eq. (5.1).

The weights displayed were attained by trail and error but there is an intuitive explanation for why these states were used and which order of magnitude they 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 was 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 5.2.2 the Amesim model made a deceleration in the longitudinal direction at the begin of the maneuver. 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 Eq. (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 Eq. (4.6). When the OCP is defined and implemented in CasADi, the opti environment is saved as a function that is called during the MPC. To improve the solving time of Eq. (5.1), the solver is switched to a SQP method using an active-set QP solver instead of IPOPT as was used in Eq. (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 of the different simulation components, 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 Eq. (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 was build.

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 Eq. (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 Eq. (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

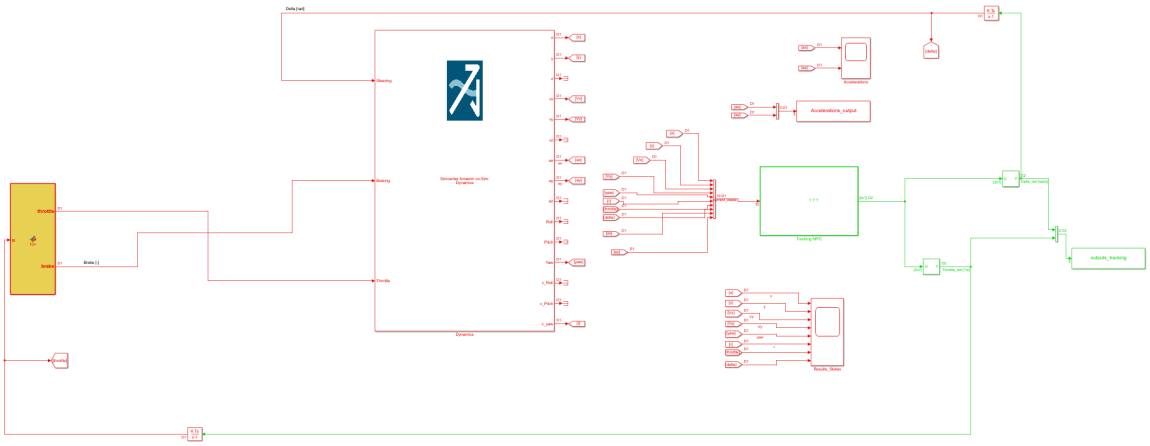


Figure 5.3: This figure shows hows the tracking MPC is connected with the Amesim model using different sampling times. (red: $T_s : 0.01\ s$ and green: $T_{MPC} : 0.1\ s$)

were separated and feeded to the Amesim model as throttling (positive) or braking (negative).

5.2.2 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 Amesim model (blue) can follow the planned reference path (red) with an accuracy of 10^{-3}. During analysing 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. This could also be seen in table 5.1 that gave similar feature values for the Amesim model and it's reference. However there are 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 Eq. (5.1),. 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 Eq. (4.4) used by the bicycle model. Another observation is that the lateral jerk and first derivative of the steerwheelangle display less high peaks. 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 Eq. (4.8) which will not be a constant anymore. 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

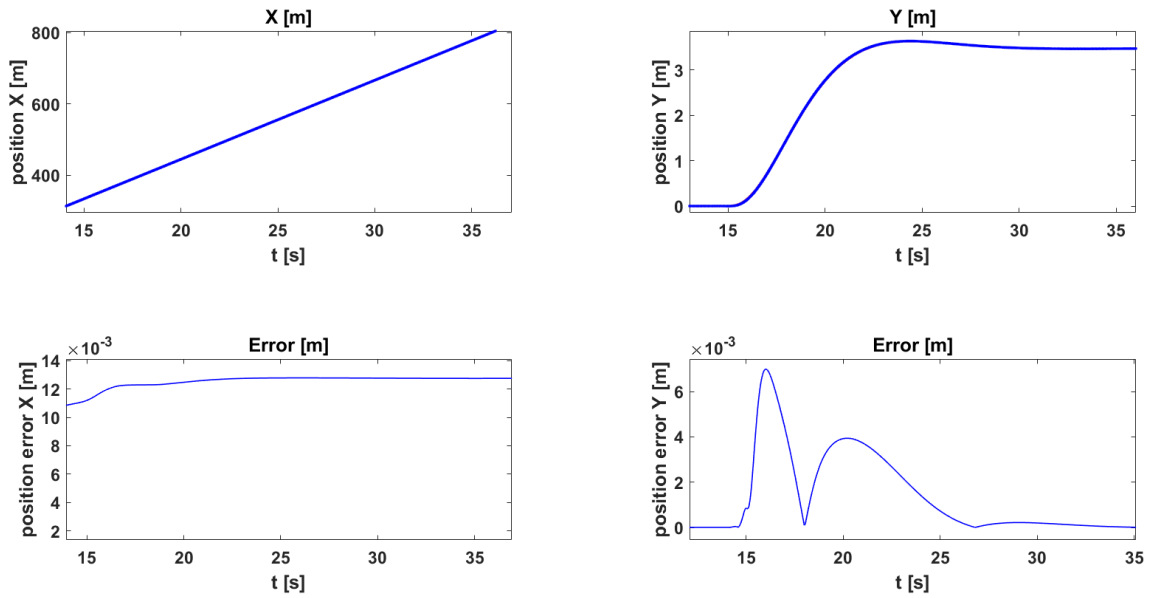


Figure 5.4: This figure shows the tracking result of the reference (red) by the Amesim model (blue).

numerical differentiation.

To be able to generate adequate 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 Eq. (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]

5.3 Learning results with the Amesim model

Diagram 5.2 shows in blue the observed path that is generated out of Eq. (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 Eq. (4.8) is called with an initial guess of the weights an all-one vector. After

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

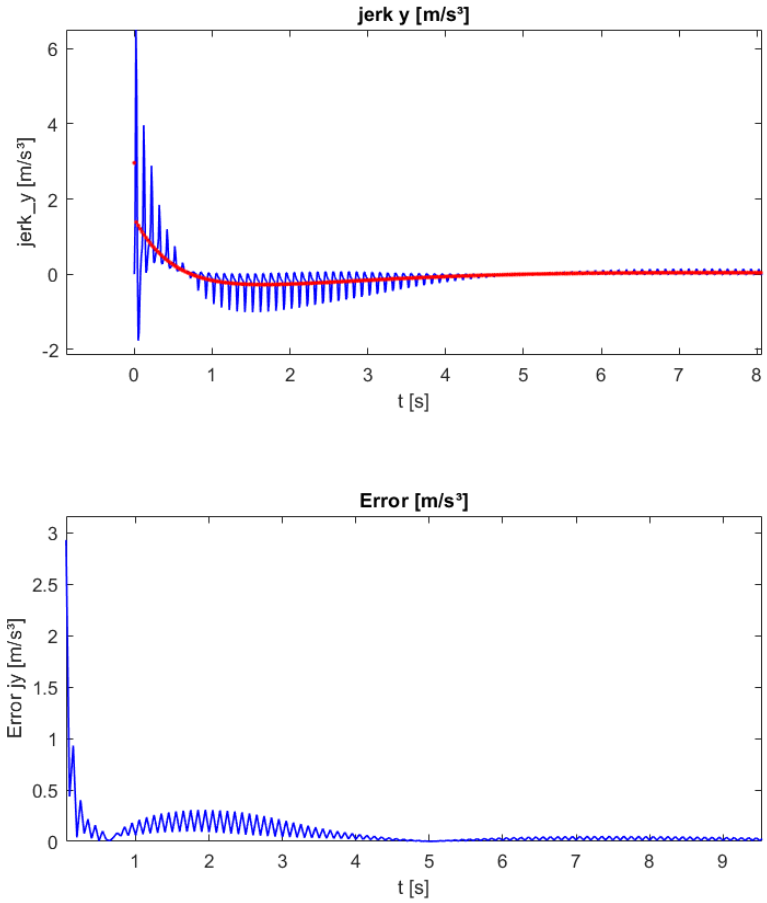


Figure 5.5: This figure shows the obtained oscillating signals for the jerk when a piecewise signal of δ_s and t_r is applied.

the planned path is outputted, tracked[3] and the associated feature vector $\boldsymbol{F}(\boldsymbol{r})$ is calculated, it is checked if the lateral feature values match accurate enough in the convergence block. If this is not the case, the difference of the features is taken as an estimate for $\frac{\partial \boldsymbol{F}_{diff}}{\partial \boldsymbol{\theta}}$ and used in the RPROP algorithm in order to generate a new planned path by calling Eq. (4.8).

Four simulations are conducted with respectively one, two, three and five 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 changed, this gives a clear

[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.

difference in the lateral feature values. An overview of the different features of the individual datasets is given in table 5.4. [4].

1 Dataset	2 Datasets	3 Datasets	5 Datasets
$V0 : 22.22 - L : 3.47$	$V0 : 22.22 - L : 3.47$	$V0 : 22.22 - L : 3.47$	$V0 : 22.22 - L : 3.47$
	$V0 : 25.00 - L : 6.94$	$V0 : 25.00 - L : 6.94$	$V0 : 25.00 - L : 6.94$
		$V0 : 27.78 - L : 3.47$	$V0 : 27.78 - L : 3.47$
			$V0 : 22.22 - L : 6.94$
			$V0 : 25.00 - L : 3.47$

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

Feature	V022.22 - L3.47	V022.22 - L6.94	V025.00 - L3.47	V025.00 - L6.94	V027.78 - L3.47
Nr.1	5.18e-07	7.20e-06	3.88e-07	5.97e-06	4.01e-07
Nr.2	0.38	1.50	0.38	1.51	0.38
Nr.3	12.29e-05	14.53e-05	1.67e-05	7.23e-05	5.37e-06
Nr.4	0.51	2.06	0.52	2.09	0.52
Nr.5	9.50e-06	4.42e-05	1.66e-05	7.08e-05	3.71e-05
Nr.6	91.25	364.99	91.25	364.97	91.25

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

The weights learned for the four different simulations is presented in table 5.5[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 averaged data feature vector. Convergence for the first three simulations is reached after 28 iterations and for 5 datasets after 26 iterations. The convergence criteria used, is matching the lateral features (2,4,6) with a tolerance of 10^{-3}.

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 in order of magnitude of 10^{-4}. All the resulting kinematic signals are displayed in Appendix E.

[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 than seen earlier in table 5.1

[5] The weights that come from the simulations are scaled so that the second weight equals 5.0.

Weight	1 Dataset	2 Datasets	3 Datasets	5 Datasets
Nr.1	33.4230	5.2294	3.9036	4.2995
Nr.2	5.0000	5.0000	5.0000	5.0000
Nr.3	2.8662e-06	1.0636	1.9803	3.4930
Nr.4	6.0018	6.0018	6.0018	6.0278
Nr.5	4.4445	1.2705	1.1869	1.5694
Nr.6	2.0011	2.0011	2.0011	2.0044

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	5 Datasets
Nr.1	1.1315	1.0097	0.9643	0.9677
Nr.2	1.0000	1.0002	1.0003	1.0005
Nr.3	0.2865	0.4496	0.4867	0.9957
Nr.4	1.0000	1.0002	1.0002	0.9993
Nr.5	1.0718	1.0030	0.9937	0.9772
Nr.6	1.0000	1.0000	1.0000	1.0000

Table 5.6: This table shows $\boldsymbol{f}_{rel}$ at convergence.

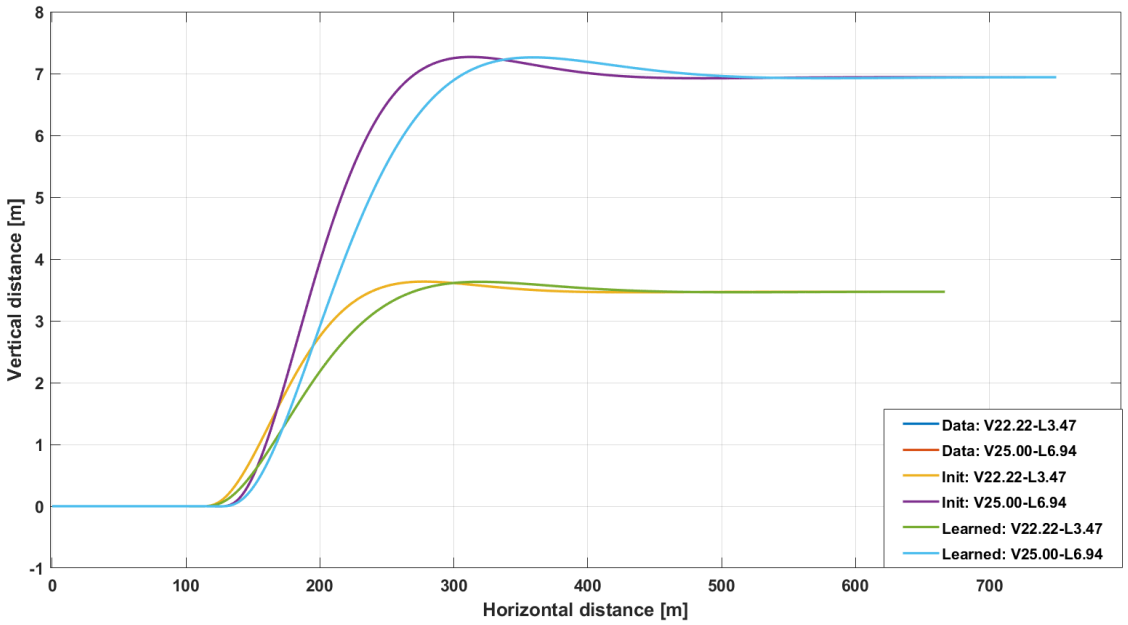


Figure 5.6: This Figure shows the observed paths, the initial solution retrieved with as weights an all one vector and the learned solution.

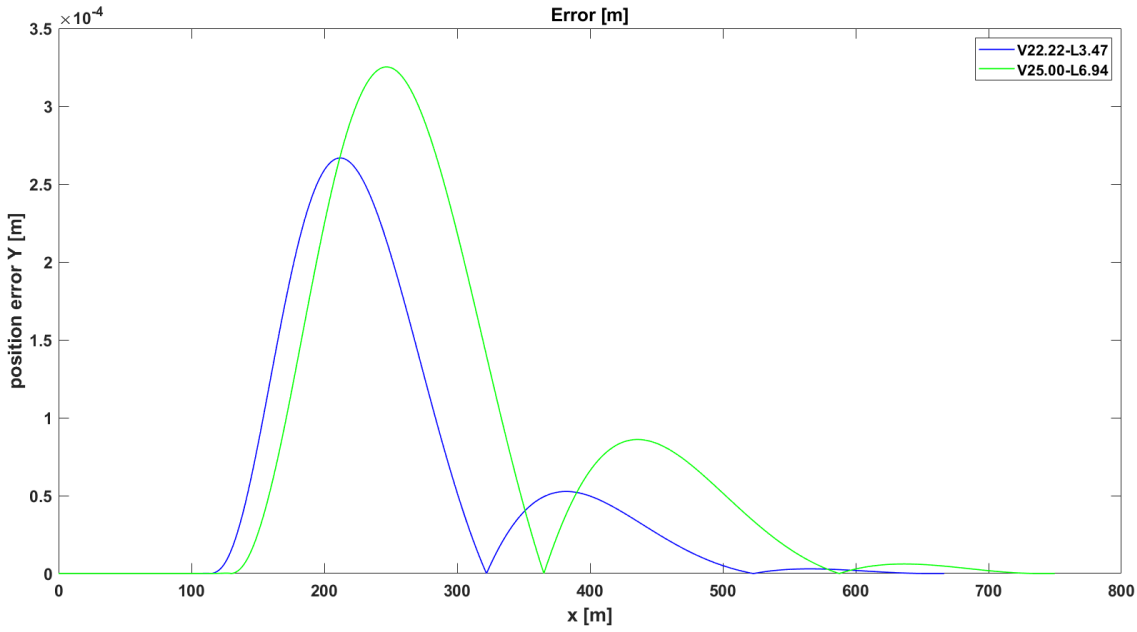


Figure 5.7: This Figure shows the error made between the observed and learned paths.

Figure 5.8 shows the convergence of $\boldsymbol{f}_{rel}$ towards one during the learning iterations when using 2 datasets. Except for the longitudinal jerk (feature 3) all the learned feature values converge towards the observed ones.
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 at convergence when three datasets are used) A reason for this is as discussed in 5.2.2, bad quality of the jerk data which contributes to a deteriorated 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.

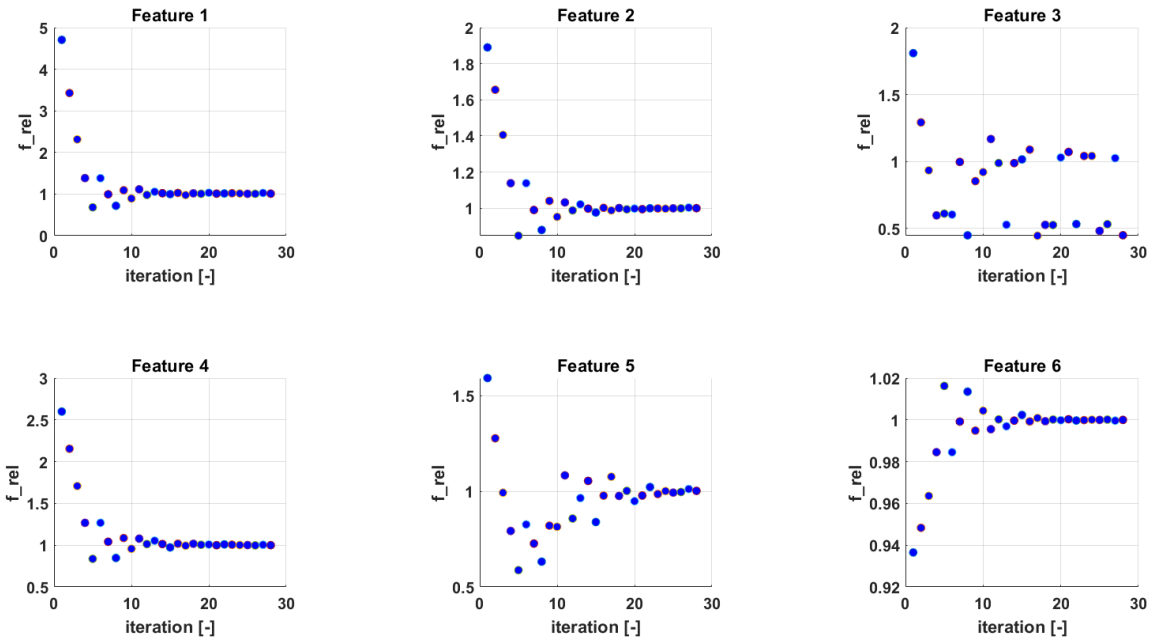


Figure 5.8: This Figure shows $\boldsymbol{f}_{rel}$ during the learning iterations.

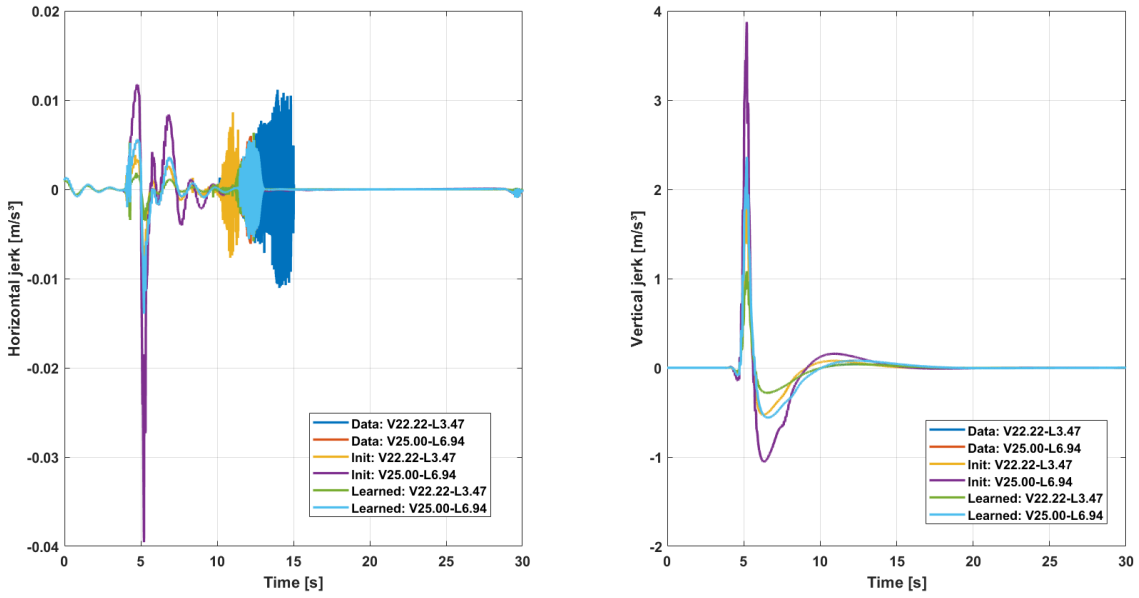


Figure 5.9: This Figure shows the observed jerk signals, the initial solution retrieved with as weights an all-one vector and the learned solution.

5.4 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 is embedded, was looked into. The question posed at the beginning was: "Is it possible to find back the underlying comfort weights when the gradient $\frac{\partial \boldsymbol{F}_{diff}}{\partial \boldsymbol{\theta}}$ is estimated by $\boldsymbol{F}^* - \boldsymbol{F}(\boldsymbol{r}_{expected})$". It was shown that this estimation is sufficient in order to learn weights that allow matching between an averaged learned feature vector and an averaged data feature vector.

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.

After the tracking MPC was validated the learning results with the Amesim model were discussed. Four simulations were conducted with a different amount of observed lane changes. Out of the results it could be concluded that for the lateral features there is a good convergence found for $f_{rel,i}$ and the chosen lateral weights are every time found back. When more observed lane changes are included during the learning also the results for the longitudinal weights and the feature matching of the longitudinal feature values improves. 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 from.

Chapter 6

Enhancement of the weight update

Bespreek de validatie van de methode. Implementeer similaties in prescan. Bespreek de verschillende software tools bij Siemens –> Amesim, simulink, prescan. Hoe werken ze samen en hoe wordt de validatie precies gedaan? Wat zijn de resultaten?

6.1 The First Topic of this Chapter

6.1.1 Item 1

Sub-item 1

Sub-item 2

6.1.2 Item 2

6.2 The Second Topic

6.3 Conclusion

Chapter 7

Conclusion

The contribution made with this thesis is the development of an algorithm based on inverse reinforced learning, that can learn driver specific experiences of comfort during a lane change captured in weights of an objective function.
First a literature study was conducted in order to identify comfort during driving. Here it was found that smooth paths play an important role in contributing to the retrieved amount of comfort of a human driver. Next, it was looked at the theoretical idea of feature based reinforcement learning. This was the starting point of this thesis and the goal was to implement this theoretical idea for in practise useable models e.g. 15 dof Amesim model.
In the following chapter, learning from ideal data was discussed. Here it was presented how ideal data was generated and validated and it was shown that the developed algorithm was able to accurately learn the chosen lateral weights that define the lane change maneuver. Thereby it was displayed that the estimation of the gradient $\frac{\partial \boldsymbol{F}_{diff}}{\partial \boldsymbol{\theta}}$ by $\tilde{\boldsymbol{F}} - \boldsymbol{F}(\boldsymbol{r}_{expected})$ can be used in order to converge towards the data feature vectors used.
Further, learning with a 15 degrees of freedom vehicle model was discussed. For this a tracking MPC formulation was developed and it was shown that also here the algorithm was able to accurately learn the chosen lateral weights while the learned feature values converge towards the data feature values.
As last the theoretical idea was discussed how to replace the RPROP algorithm for updating the weight vector $\boldsymbol{\theta}$ by an update based on an optimization.

This work has its practical usage in the avoidance of manual tuning of a comfort objective which when identified, can be used for path planning. Because this research is conducted together with Siemens, it is in line with their research and this thesis can be used to build further on.
The maneuver discussed is a lane change maneuver. The only way that this is defined in Eq. 4.8 is by setting constraints on the end state so that a lane change is performed. It is thereby possible to change these end constraints so that also other maneuvers are learned e.g. an acceleration maneuver.
During this thesis the environment is not assessed although this is a necessary

condition in order to take comfort into account as followed out of section 3.1. As discussed in section 4.2.2 it is possible to add features that encapsulate notions about the environment of the autonomous vehicle so that the feeling of safe driving can be assured. Therefore a suggestion for future work is to perform the developed algorithms on real driver data which also contains information about the vehicle its surroundings.

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.

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

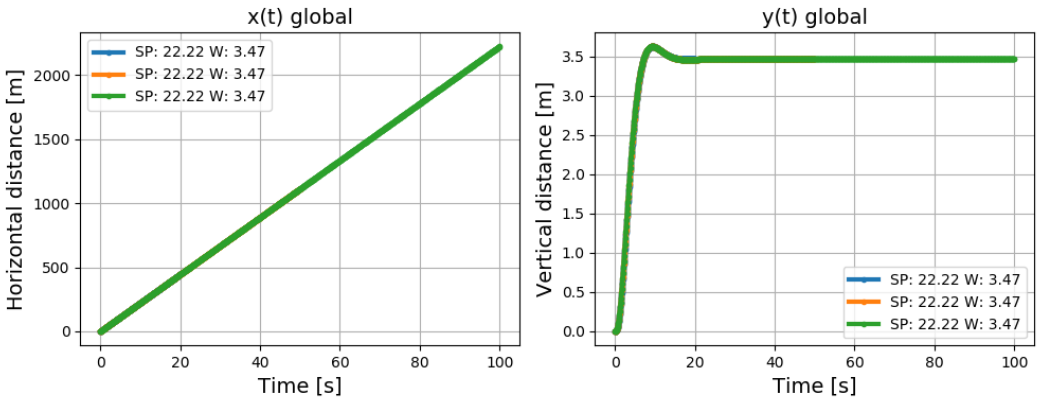
$$\begin{aligned} j_{y,t} = \frac{1}{M} \cdot (&cos\delta t_r c\dot{\delta} + sin\delta \dot{t_r} c + sin\delta 2K_{y,f} arctan(\frac{v_y + \omega a}{v_x})\dot{\delta} \\ &- cos\delta 2K_{y,f} \frac{v_x a_{t,y} + v_x \dot{\omega} a - a_{t,x} v_y - a_{t,x} \omega a}{v_x^2 + v_y^2 + 2v_y \omega a + \omega^2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

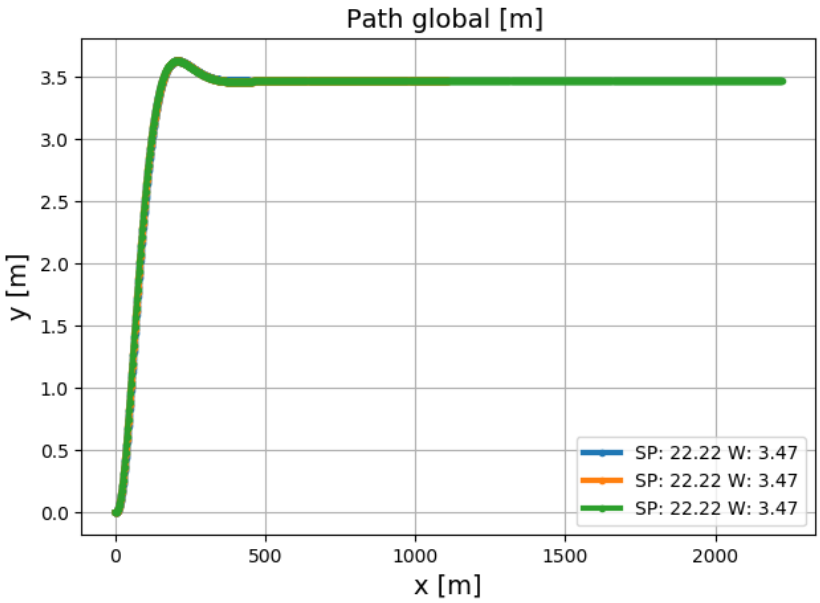
Appendix B

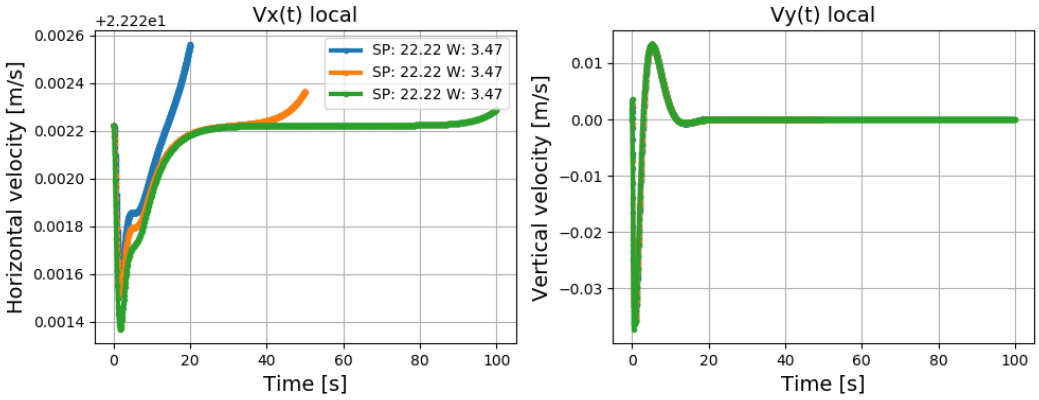
Results of the ideal data validation

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.

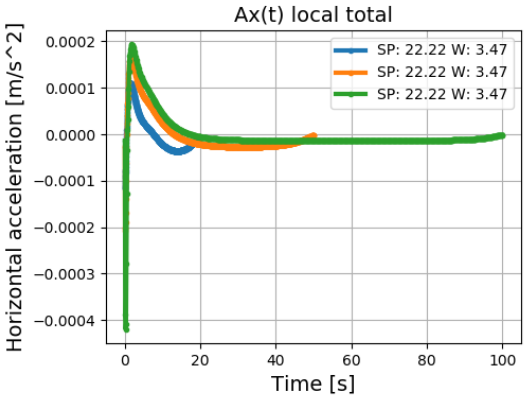
B.1 Time limit

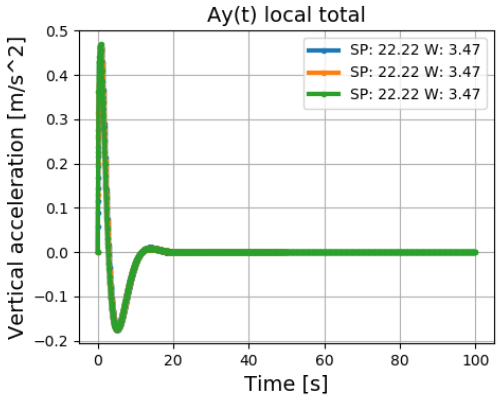


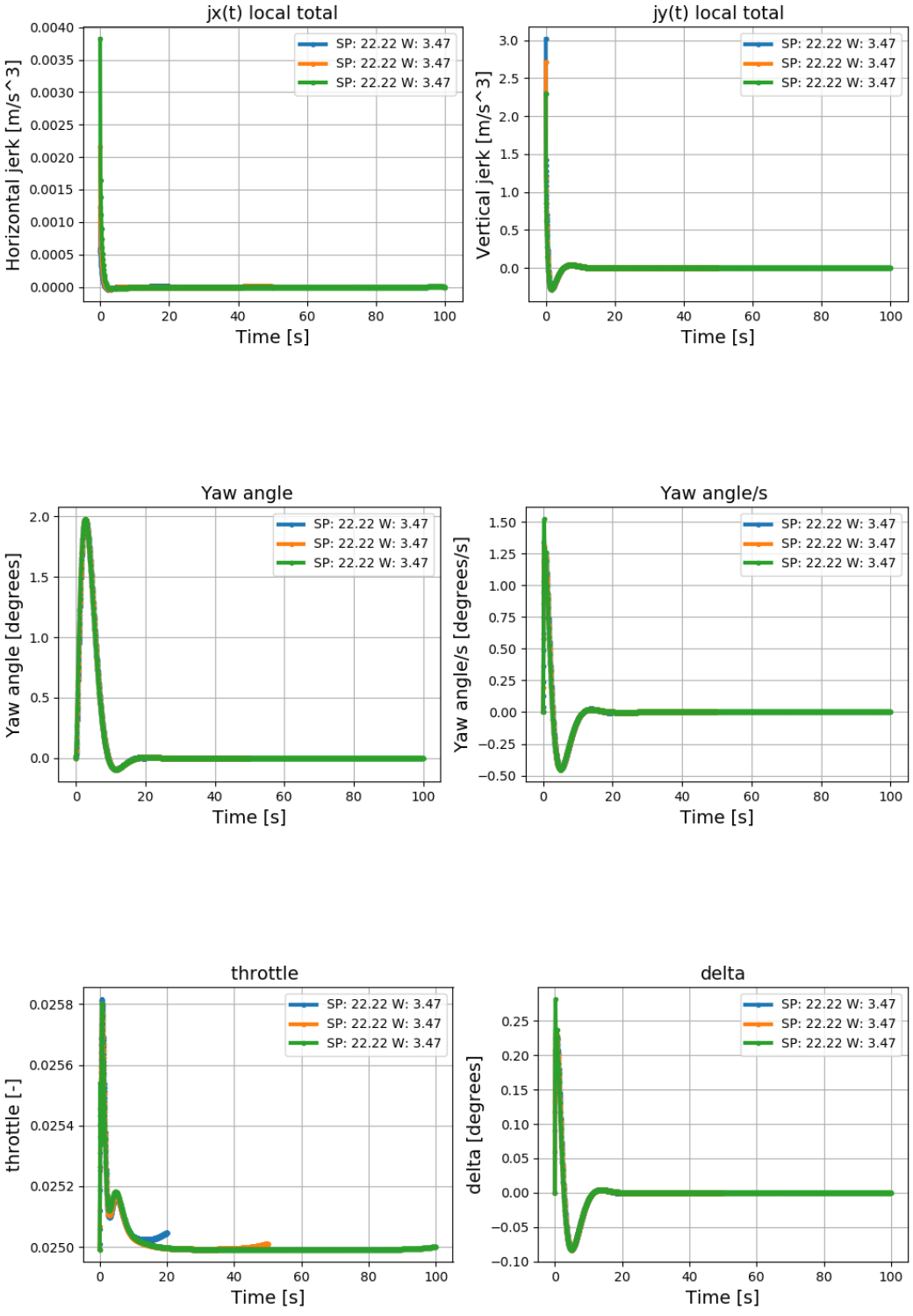

Path global [m]
y [m]
x [m]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47

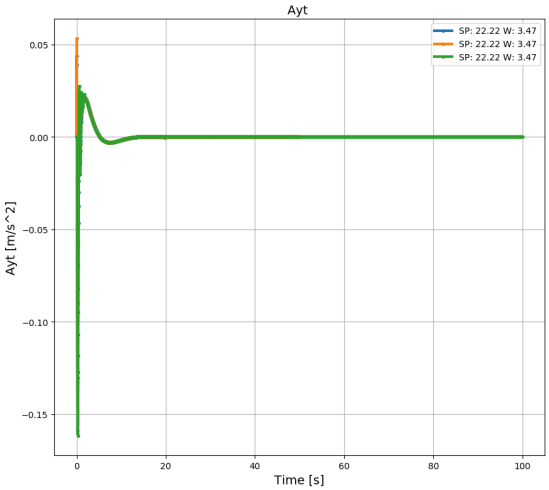

Vx(t) local
+2.222e1
Horizontal velocity [m/s]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47

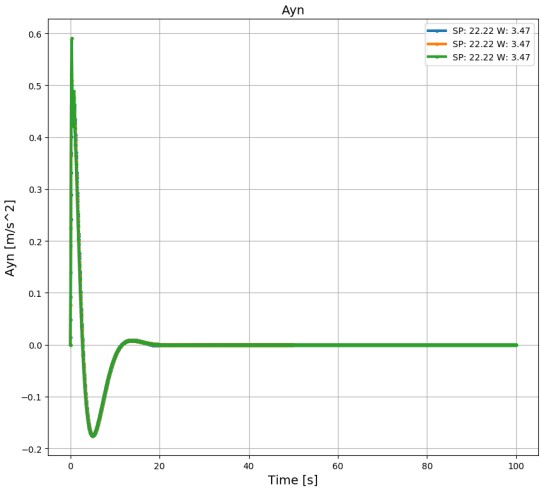
Vy(t) local
Vertical velocity [m/s]
Time [s]

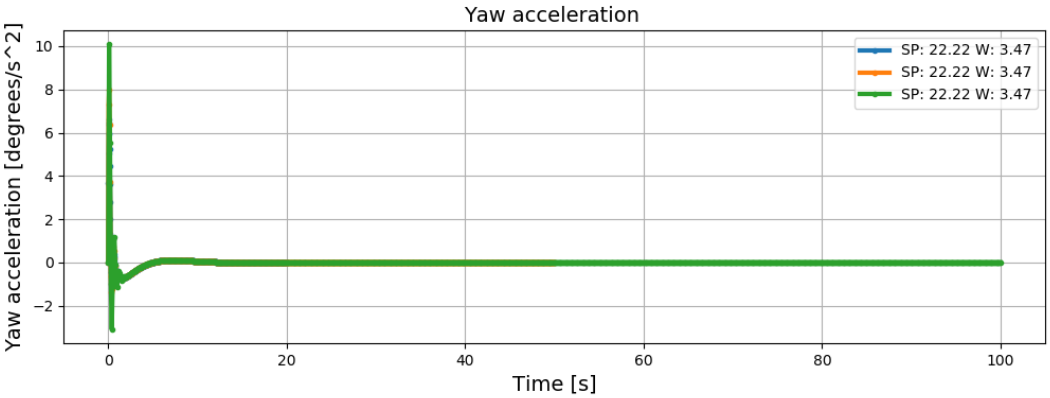

Ax(t) local total
Horizontal acceleration [m/s^2]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47

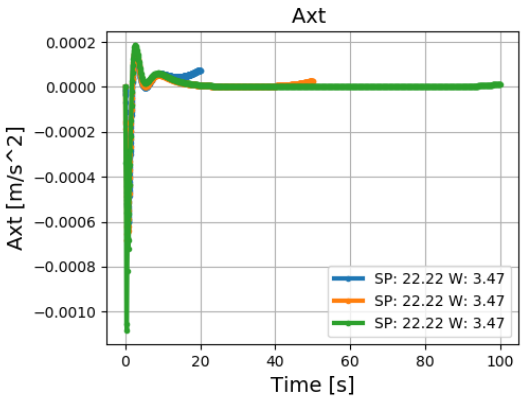

Ay(t) local total
Vertical acceleration [m/s^2]
Time [s]

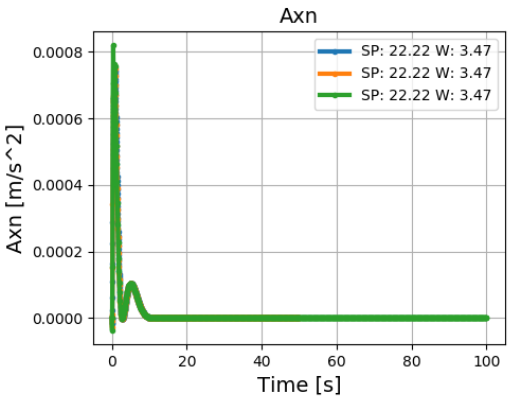

jx(t) local total
Horizontal jerk [m/s^3]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
jy(t) local total
Vertical jerk [m/s^3]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Yaw angle
Yaw angle [degrees]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47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

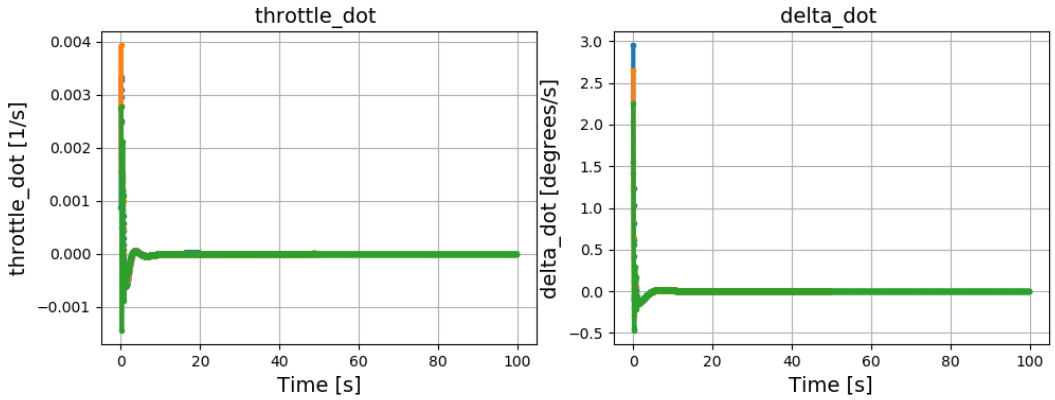

Ayt
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Ayt [m/s^2]
Time [s]


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]

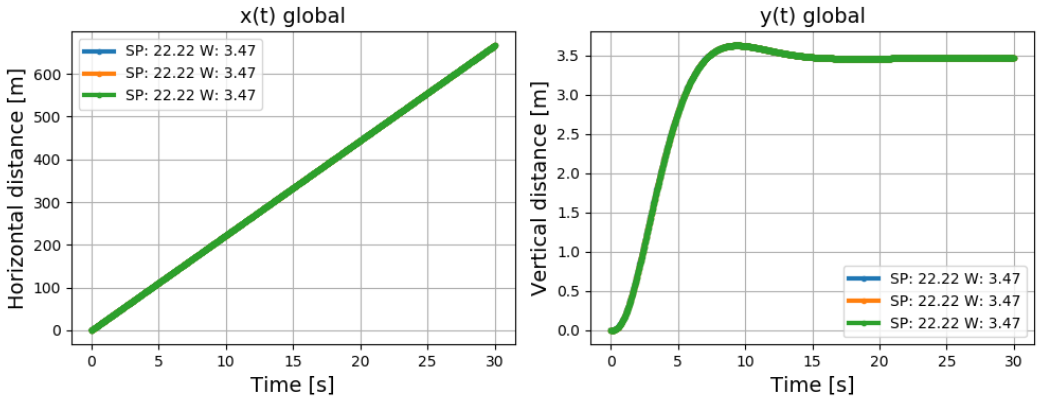

Yaw acceleration
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Yaw acceleration [degrees/s^2]
Time [s]

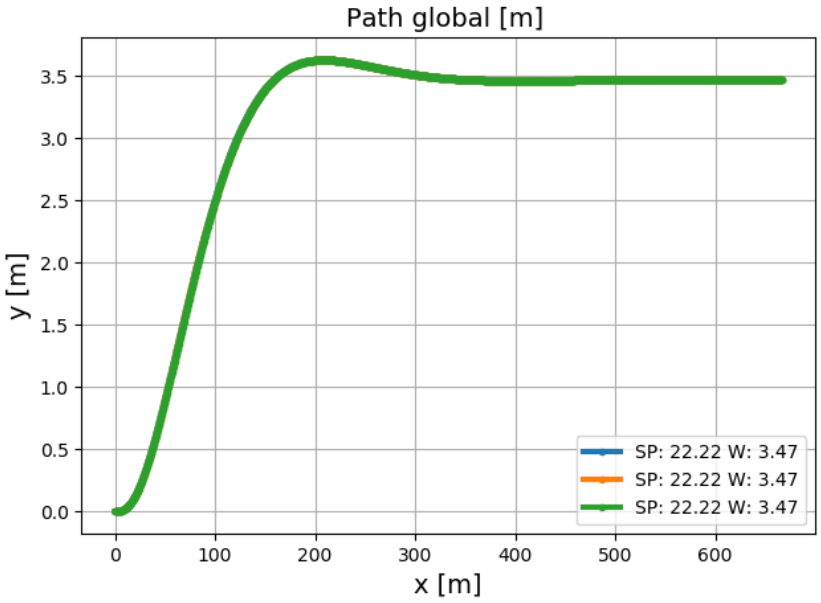

Axt
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Axt [m/s^2]
Time [s]

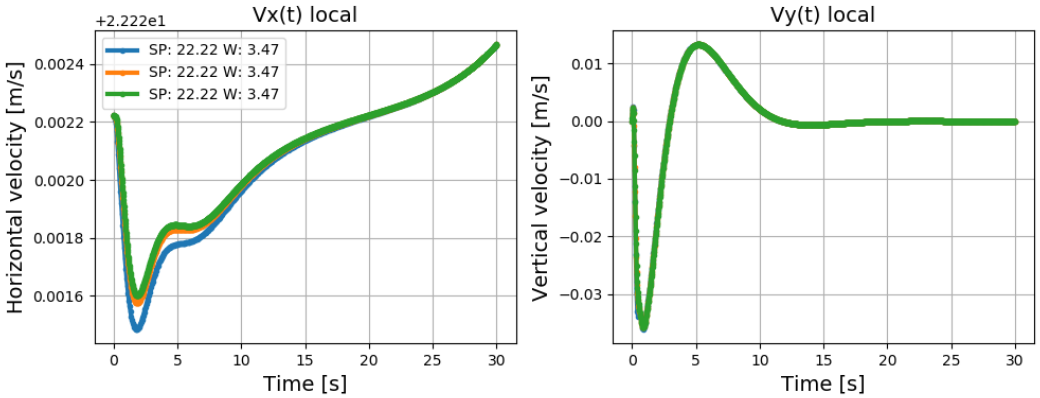

Axn
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Axn [m/s^2]
Time [s]


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]

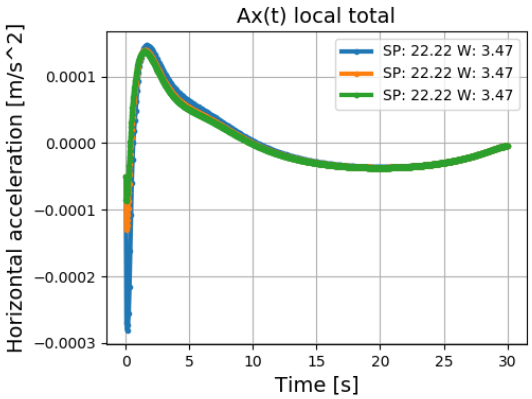
B.2 Amount of optimization points

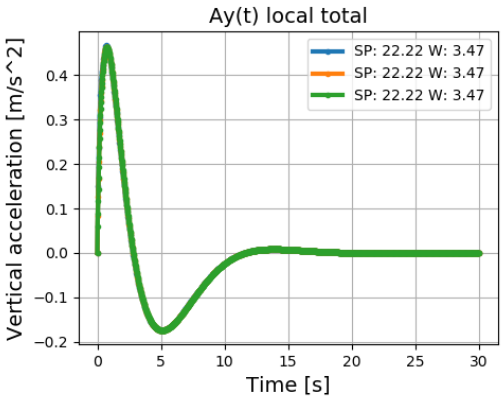


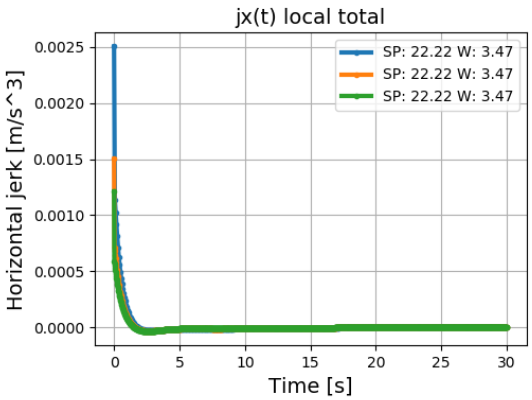


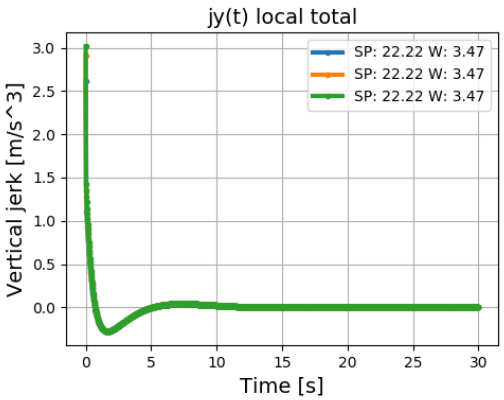

Vx(t) local
+2.222e1
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Horizontal velocity [m/s]
Time [s]

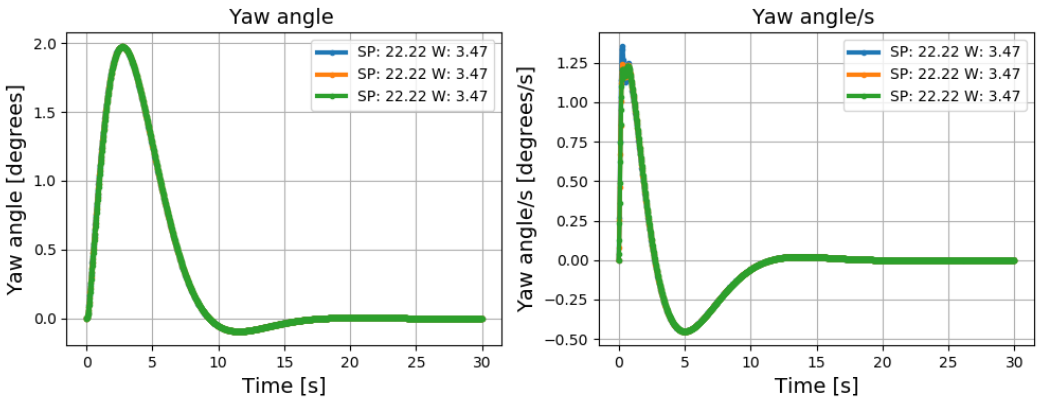
Vy(t) local
Vertical velocity [m/s]
Time [s]


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]

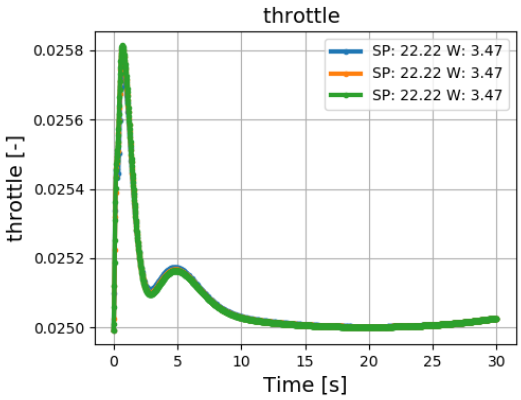

Ay(t) local total
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Vertical acceleration [m/s^2]
Time [s]

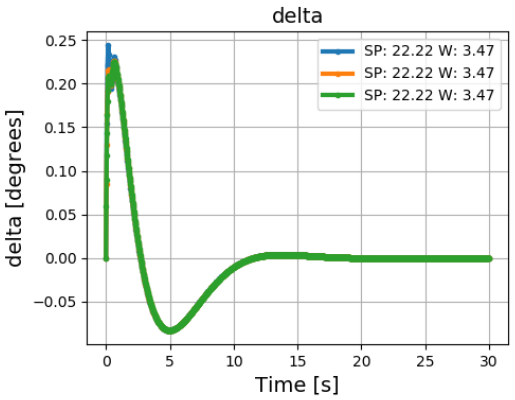

jx(t) local total
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Horizontal jerk [m/s^3]
Time [s]

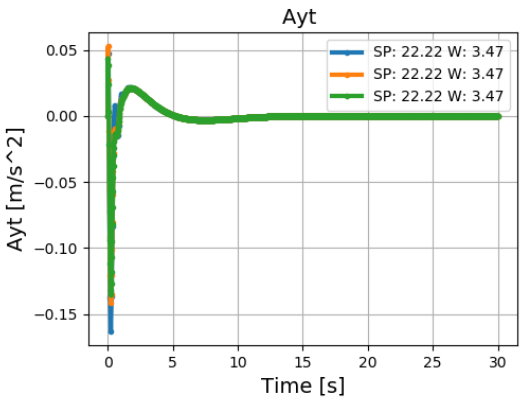

jy(t) local total
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
Vertical jerk [m/s^3]
Time [s]

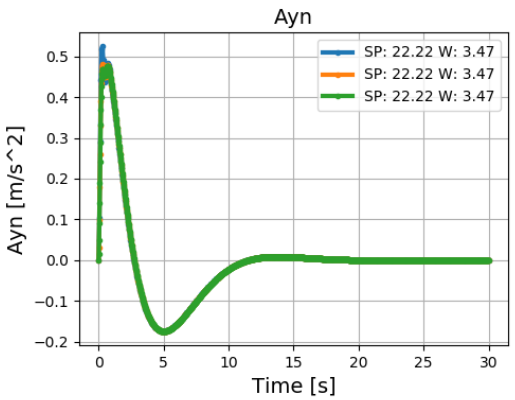

Yaw angle
Yaw angle [degrees]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47

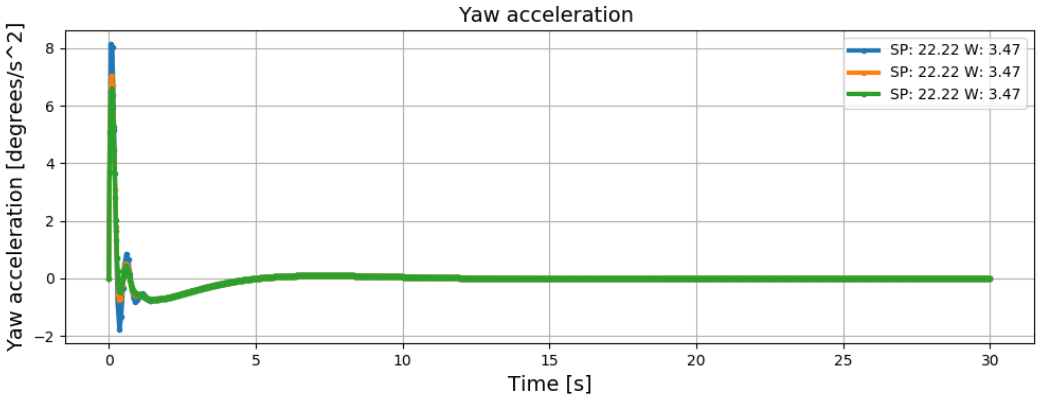
Yaw angle/s
Yaw angle/s [degrees/s]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47

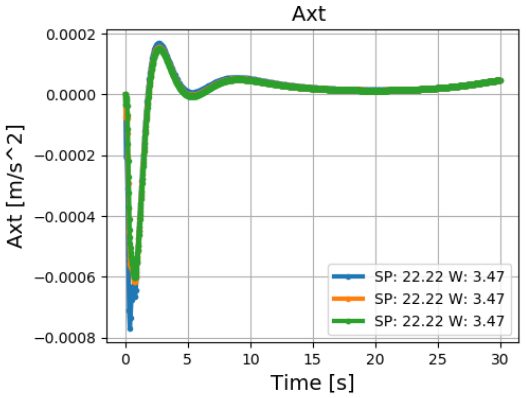

throttle
throttle [-]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47

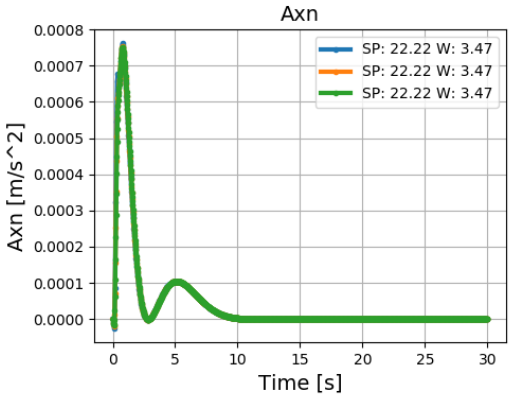

delta
delta [degrees]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47

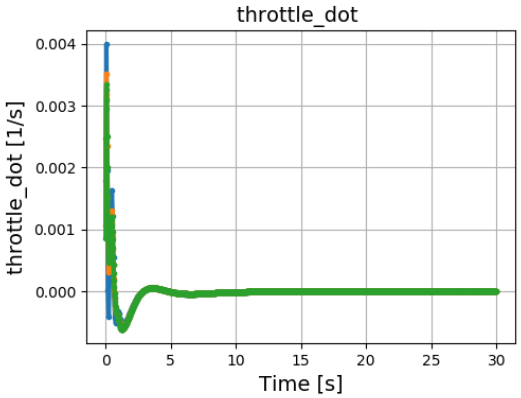

Ayt
Ayt [m/s^2]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47

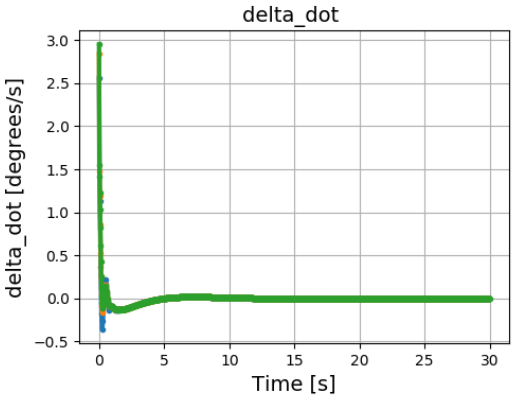

Ayn
Ayn [m/s^2]
Time [s]
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47
SP: 22.22 W: 3.47


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

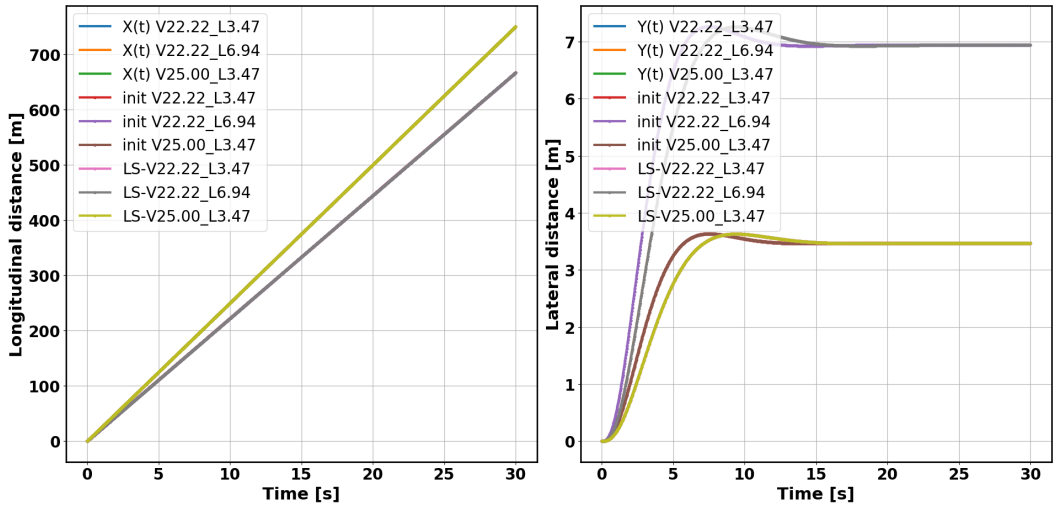

throttle_dot
throttle_dot [1/s]
Time [s]

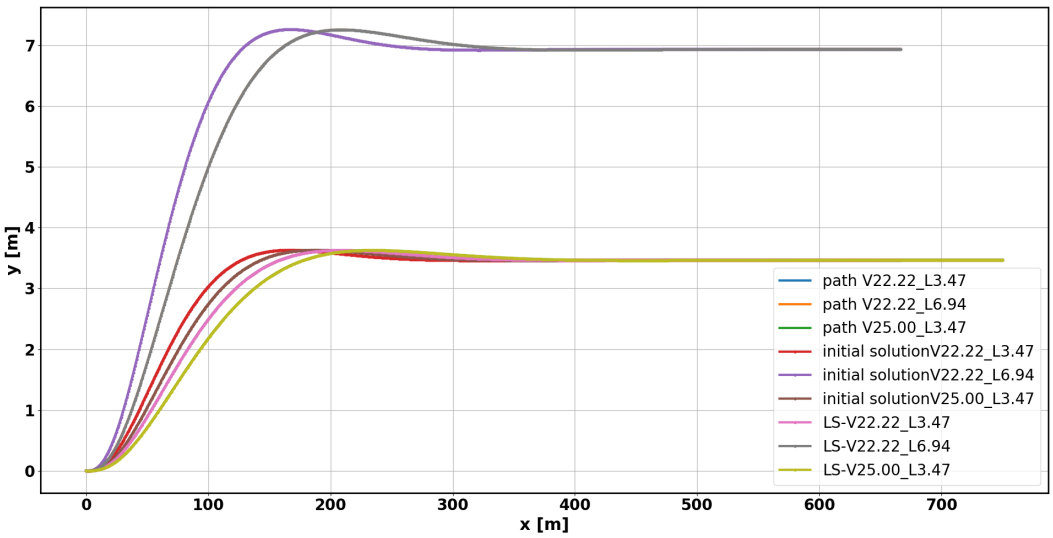

delta_dot
delta_dot [degrees/s]
Time [s]

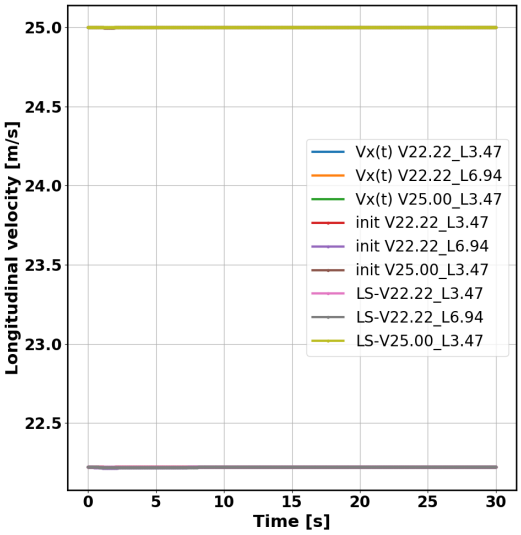
Appendix C

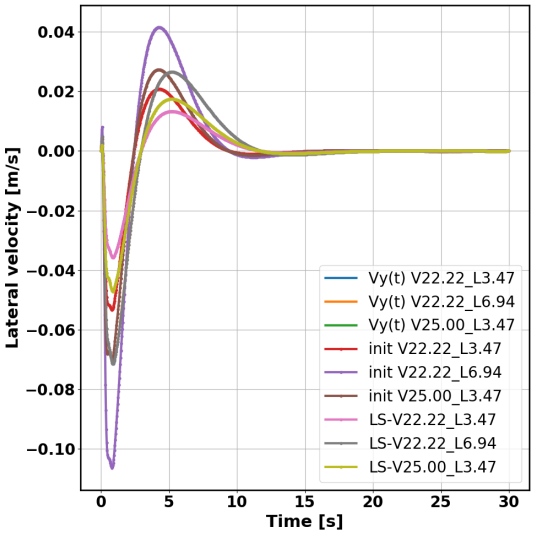
Results of the averaging learning on 3 ideal datasets

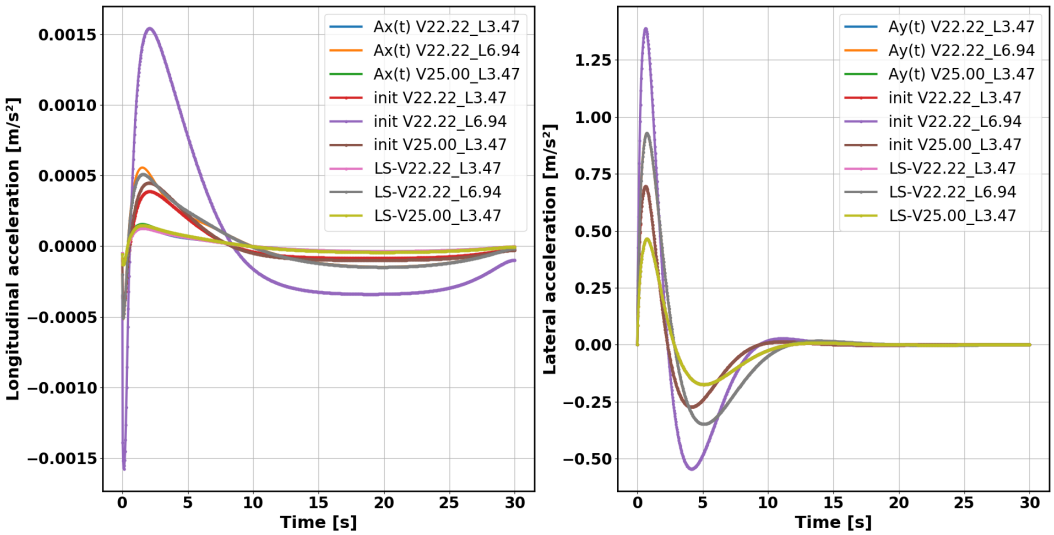
The figures shown in this appendix serve as a completion on the figures that were presented in the discussion of section 4.4.2. It concerns the results of the learning algorithm that learns from three ideal datasets with the amount of control points N equal to 1000 and time limit set to 30 s.
First the 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.

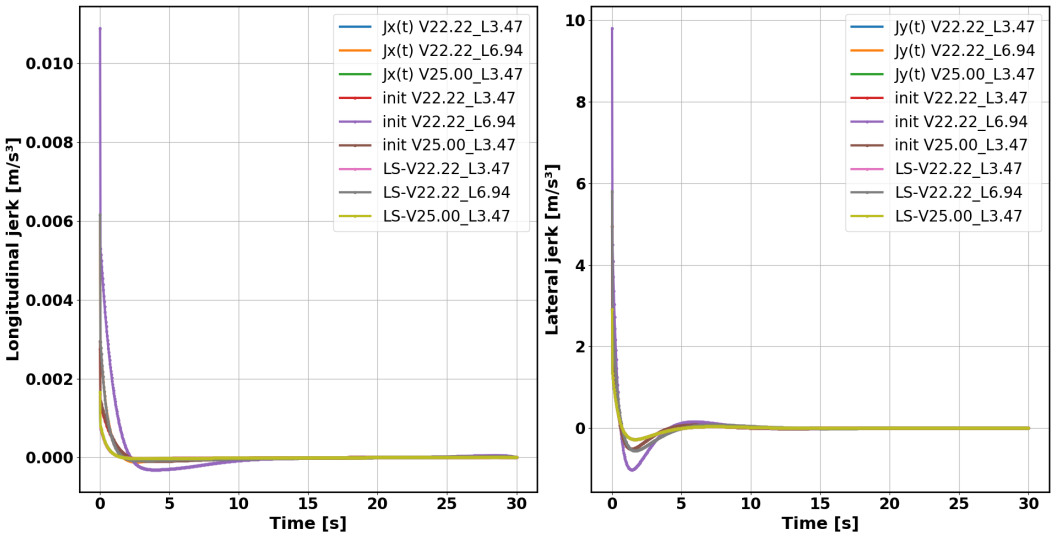



y [m]
x [m]
path V22.22_L3.47
path V22.22_L6.94
path V25.00_L3.47
initial solutionV22.22_L3.47
initial solutionV22.22_L6.94
initial solutionV25.00_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.00_L3.47


Longitudinal velocity [m/s]
Time [s]
Vx(t) V22.22_L3.47
Vx(t) V22.22_L6.94
Vx(t) V25.00_L3.47
init V22.22_L3.47
init V22.22_L6.94
init V25.00_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.00_L3.47


Lateral velocity [m/s]
Time [s]
Vy(t) V22.22_L3.47
Vy(t) V22.22_L6.94
Vy(t) V25.00_L3.47
init V22.22_L3.47
init V22.22_L6.94
init V25.00_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.00_L3.47


Longitudinal acceleration [m/s²]
Time [s]
Ax(t) V22.22_L3.47
Ax(t) V22.22_L6.94
Ax(t) V25.00_L3.47
init V22.22_L3.47
init V22.22_L6.94
init V25.00_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.00_L3.47
Lateral acceleration [m/s²]
Time [s]
Ay(t) V22.22_L3.47
Ay(t) V22.22_L6.94
Ay(t) V25.00_L3.47
init V22.22_L3.47
init V22.22_L6.94
init V25.00_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.00_L3.47


Longitudinal jerk [m/s³]
Time [s]
Jx(t) V22.22_L3.47
Jx(t) V22.22_L6.94
Jx(t) V25.00_L3.47
init V22.22_L3.47
init V22.22_L6.94
init V25.00_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.00_L3.47
Lateral jerk [m/s³]
Time [s]
Jy(t) V22.22_L3.47
Jy(t) V22.22_L6.94
Jy(t) V25.00_L3.47
init V22.22_L3.47
init V22.22_L6.94
init V25.00_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.00_L3.47

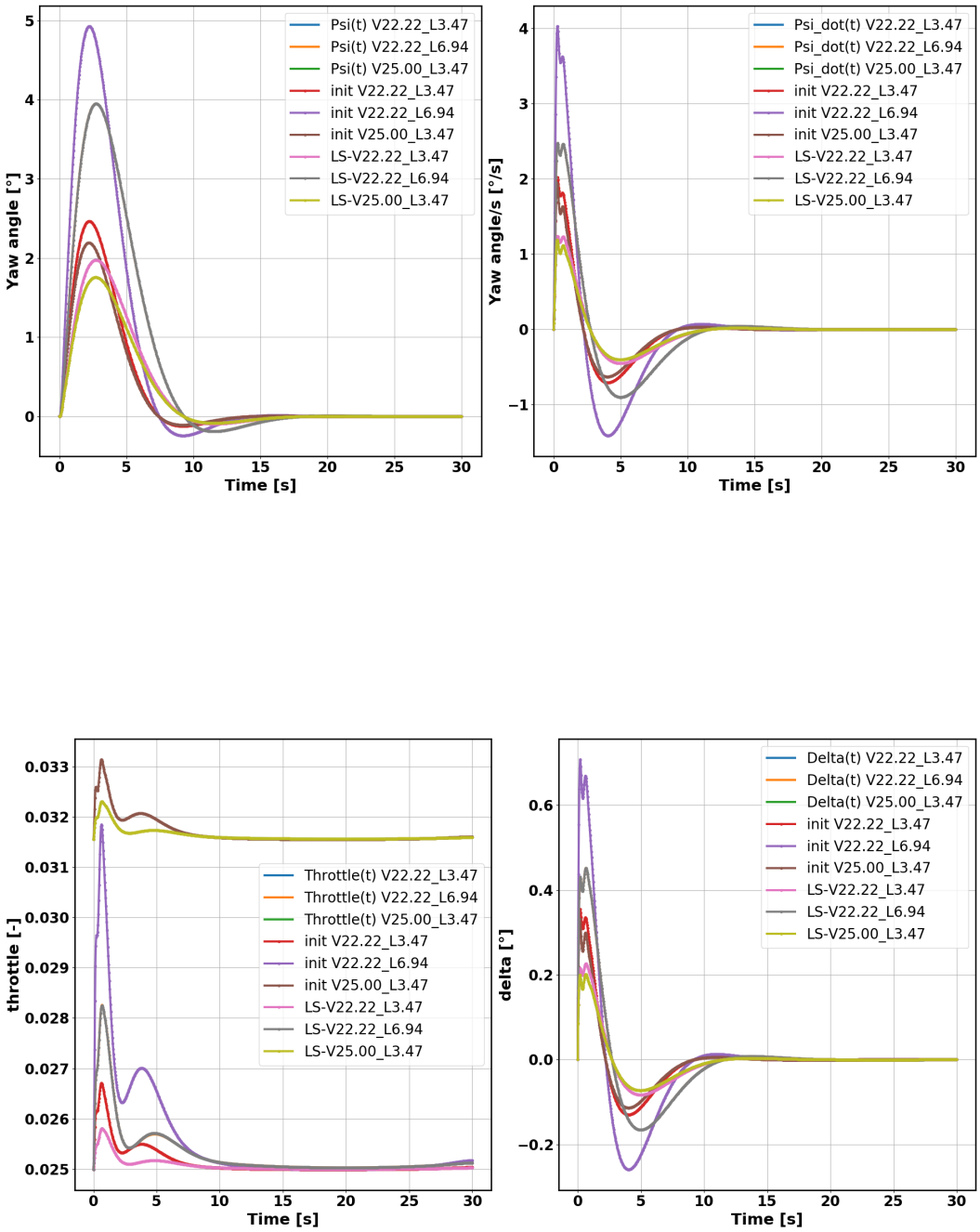
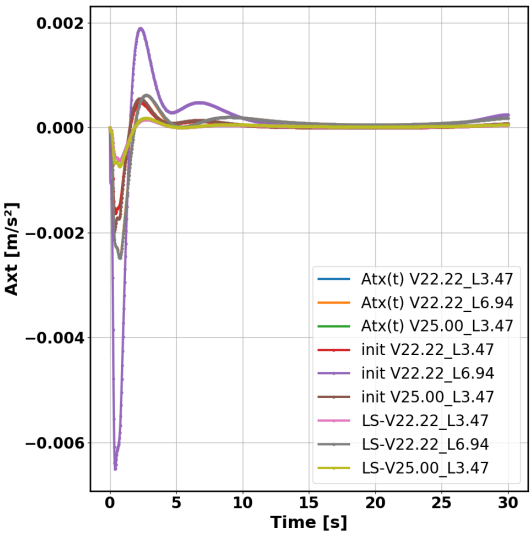
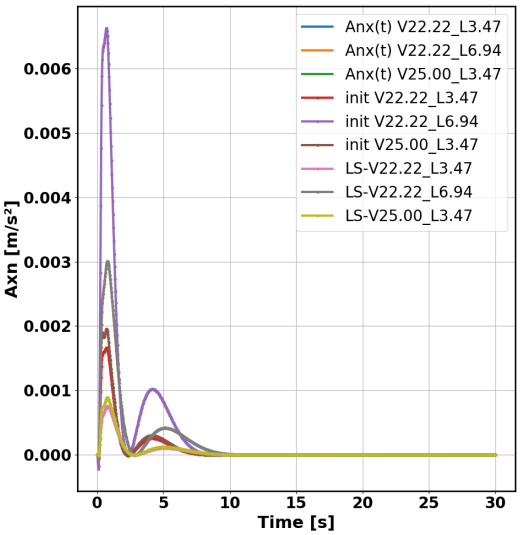
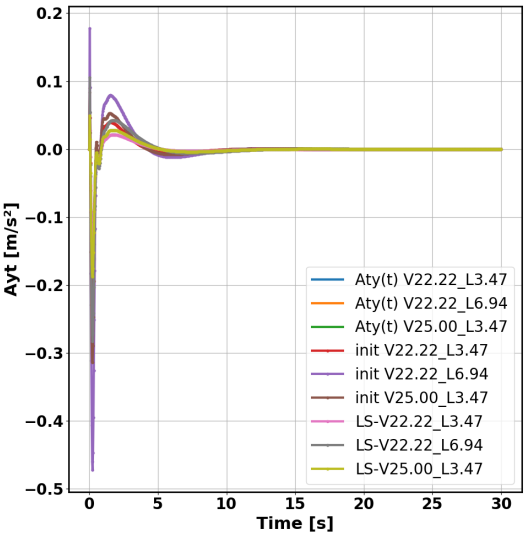
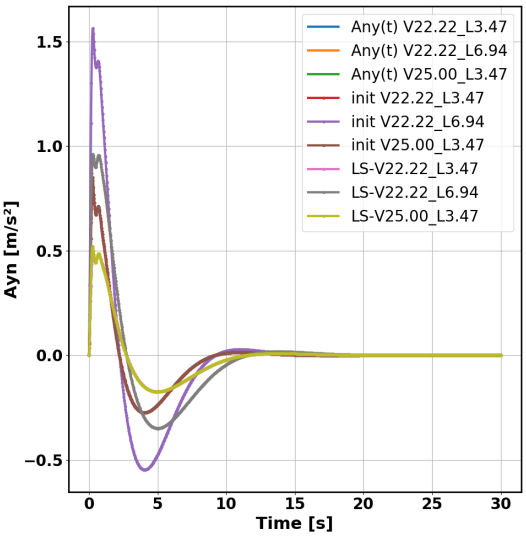


Figure C.1: This figure shows the amount of throttle and the angle of the front wheel of the bicycle model during the lane change maneuvers.


Axt [m/s²]
Time [s]
Atx(t) V22.22_L3.47
Atx(t) V22.22_L6.94
Atx(t) V25.00_L3.47
init V22.22_L3.47
init V22.22_L6.94
init V25.00_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.00_L3.47


Axn [m/s²]
Time [s]
Anx(t) V22.22_L3.47
Anx(t) V22.22_L6.94
Anx(t) V25.00_L3.47
init V22.22_L3.47
init V22.22_L6.94
init V25.00_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.00_L3.47


Ayt [m/s²]
Time [s]
Aty(t) V22.22_L3.47
Aty(t) V22.22_L6.94
Aty(t) V25.00_L3.47
init V22.22_L3.47
init V22.22_L6.94
init V25.00_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.00_L3.47


Ayn [m/s²]
Time [s]
Any(t) V22.22_L3.47
Any(t) V22.22_L6.94
Any(t) V25.00_L3.47
init V22.22_L3.47
init V22.22_L6.94
init V25.00_L3.47
LS-V22.22_L3.47
LS-V22.22_L6.94
LS-V25.00_L3.47

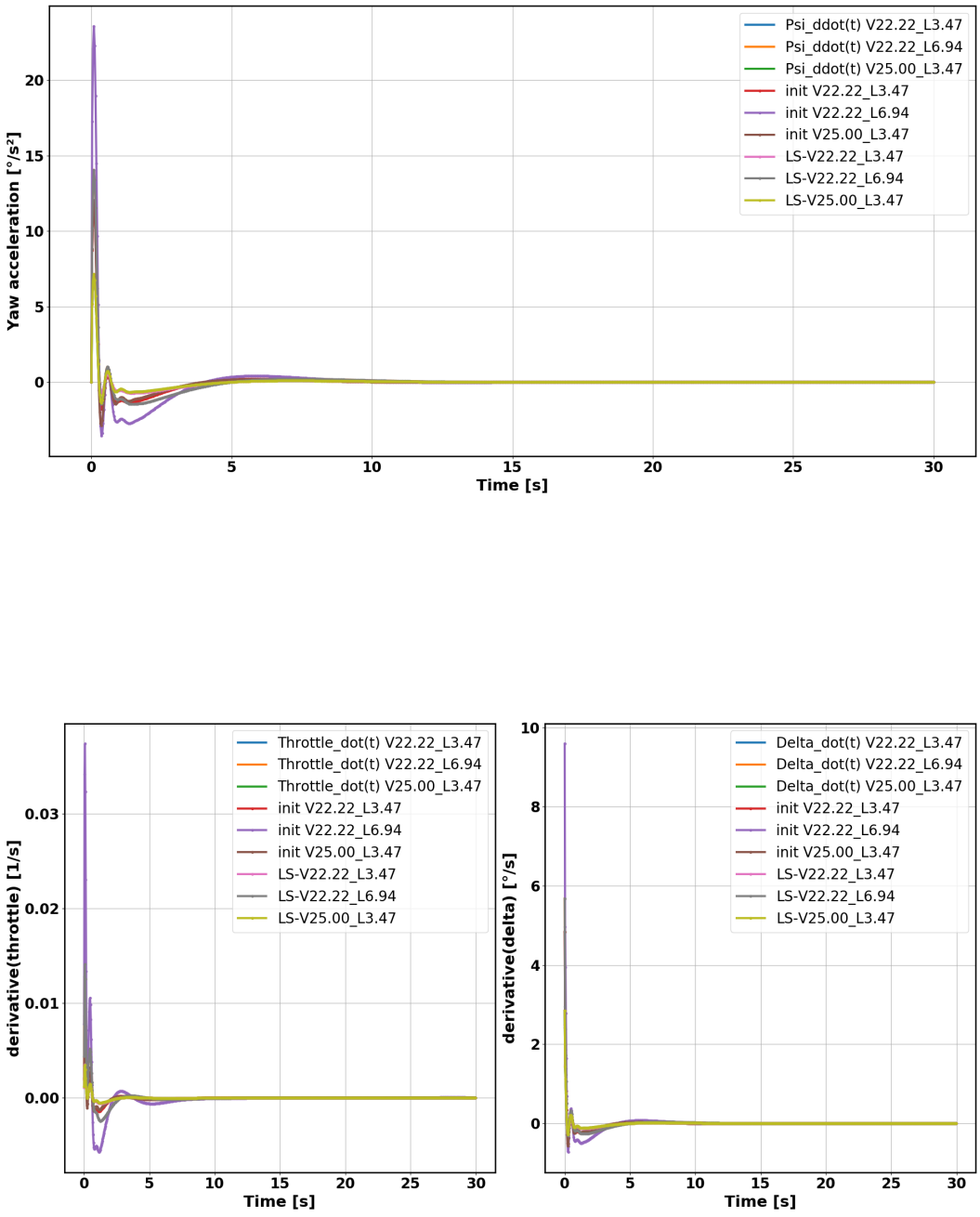


Figure C.2: This figure shows the amount of first derivative of throttle and the first derivative of the angle of the front wheel of the bicycle model is given as input during the lane change maneuvers.

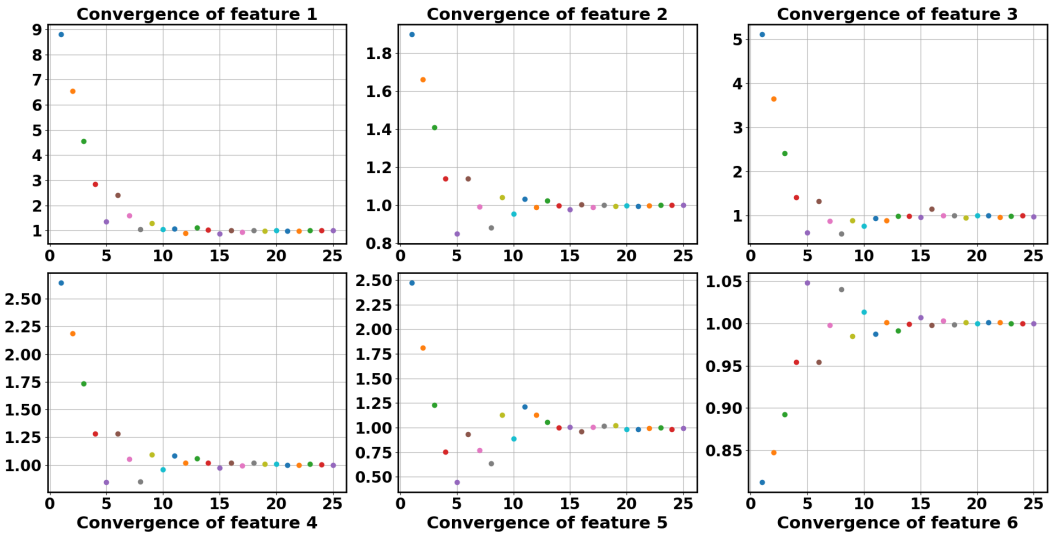


Figure C.3: The convergence of $\boldsymbol{f}_{rel}$ towards one during the learning iterations.

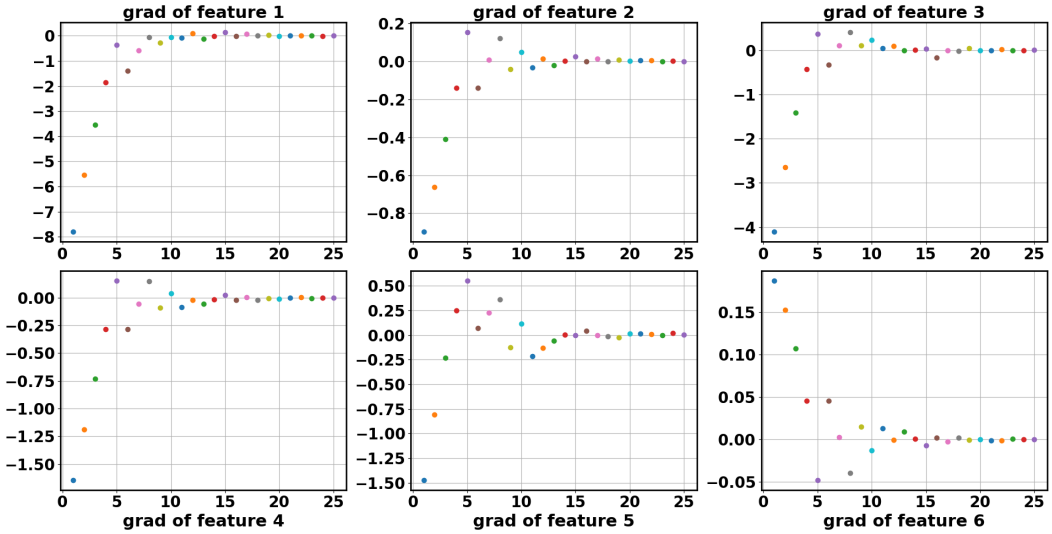


Figure C.4: The gradient $\frac{\partial \boldsymbol{F}_{diff}}{\partial \boldsymbol{\theta}}$ estimated by $\boldsymbol{f}_{obs} - \boldsymbol{f}_{learned}$ towards zero during the different learning iterations.

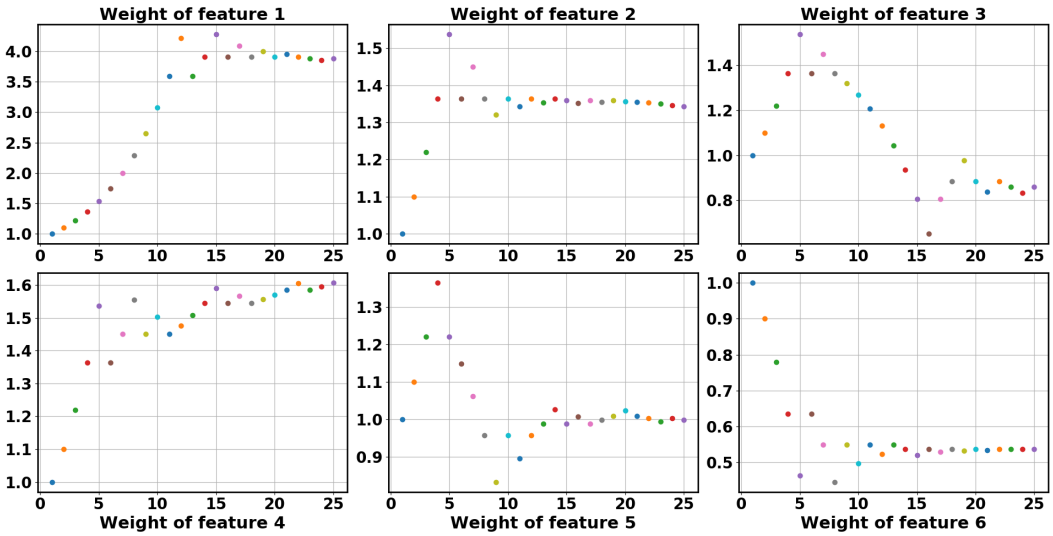


Figure C.5: The learned weights during the different learning iterations.

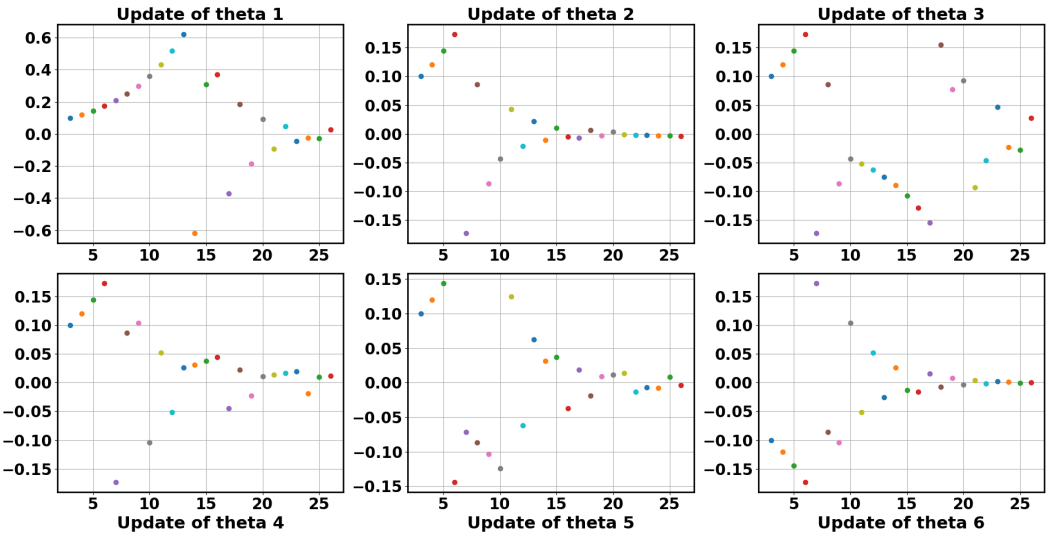


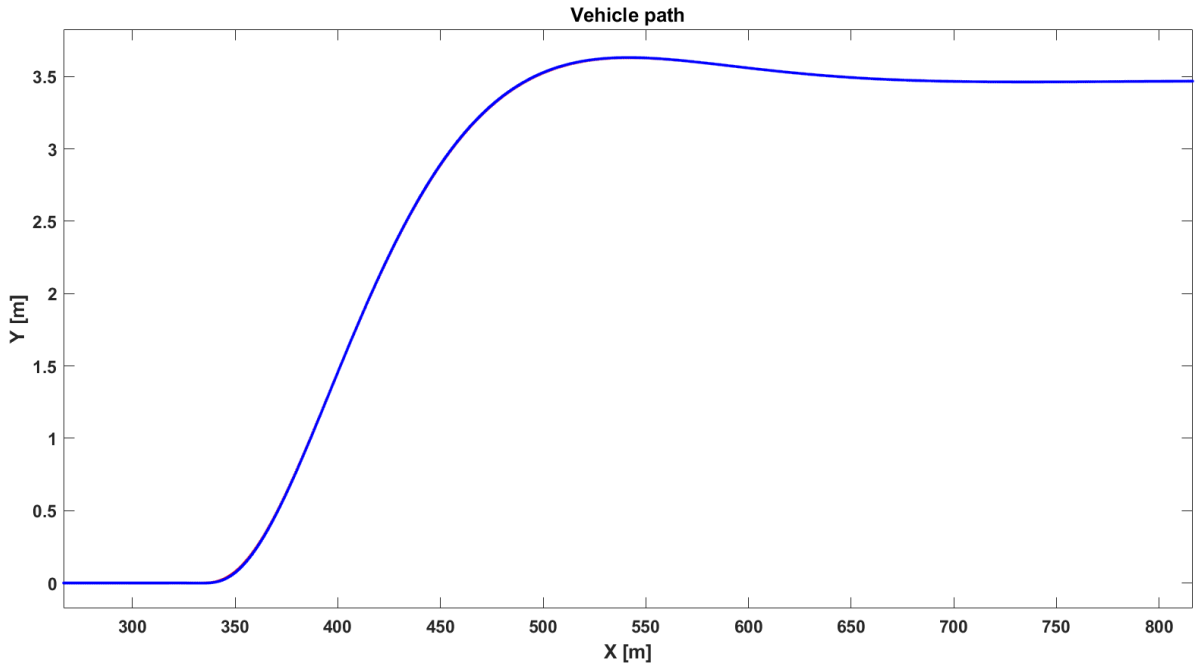
Figure C.6: The difference of $\boldsymbol{\theta}$ with respect to one used in the previous iteration.

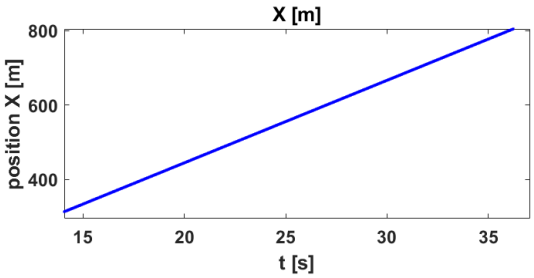
Figure C.7: This figure shows which case of the three available in the RPROP algorithm is chosen during the learning iterations.

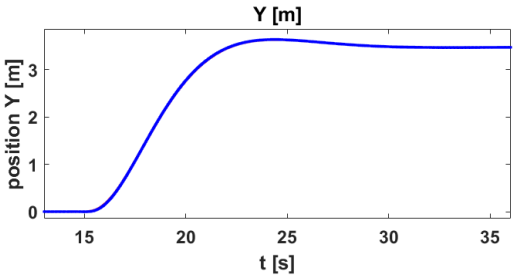
Appendix D

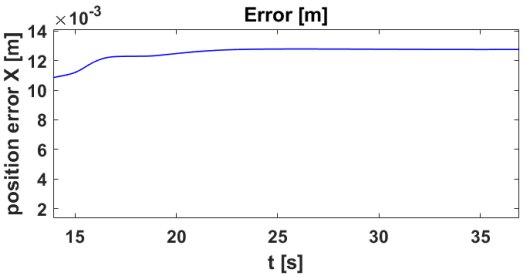
Accuracy results of the tracking MPC

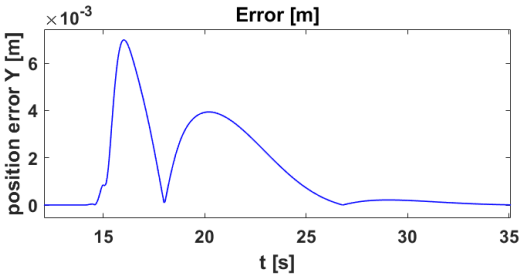
This appendix shows the results that are discussed in 5.2.2. In red the reference is shown that was derived from 4.8 with using the bicycle model and in blue the trajectory completed by the Amesim model. During the learning process with the Amesim model (section 5.3) the first 10 s of the Amesim trajectory is thrown away in order to remove the unstable longitudinal behaviour at the start of the simulation.

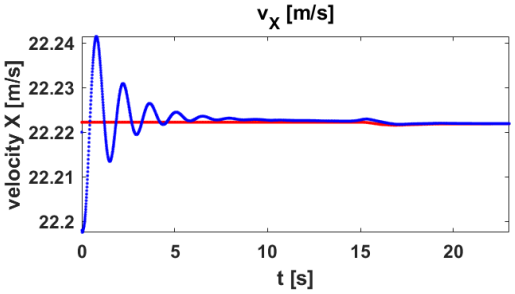


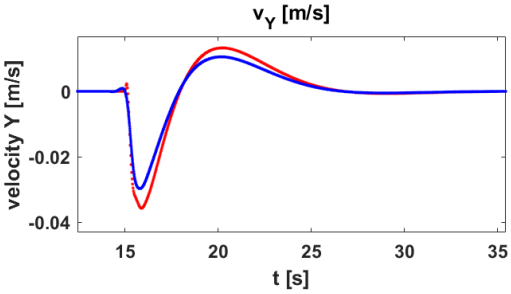

X [m]
position X [m]
800
600
400
15
20
25
30
35
t [s]

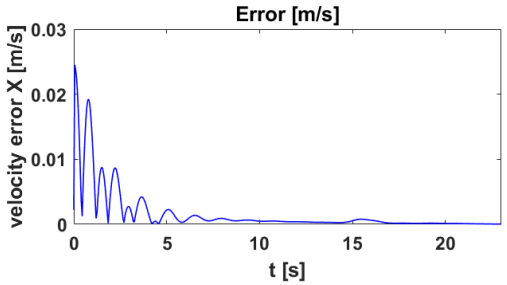

Y [m]
position Y [m]
3
2
1
0
15
20
25
30
35
t [s]

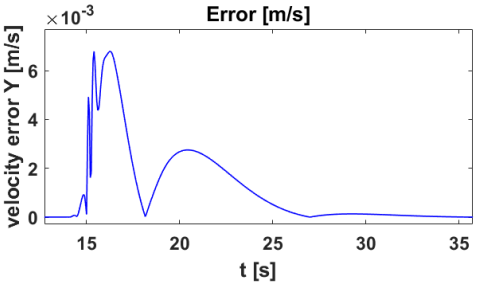

Error [m]
×10-3
position error X [m]
14
12
10
8
6
4
2
15
20
25
30
35
t [s]

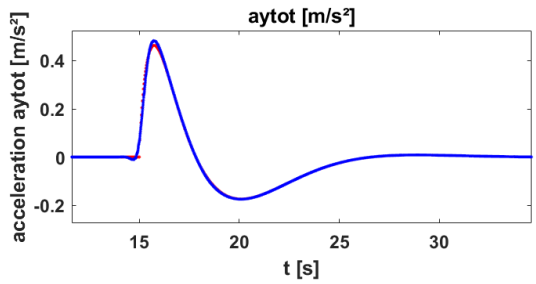

Error [m]
×10-3
position error Y [m]
6
4
2
0
15
20
25
30
35
t [s]

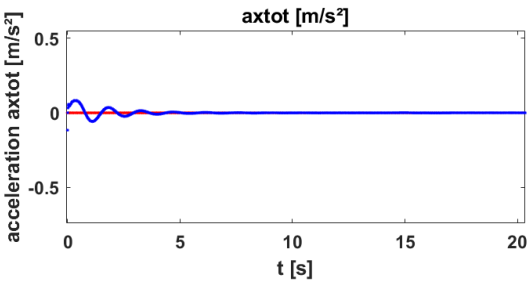

vX [m/s]
velocity X [m/s]
22.24
22.23
22.22
22.21
22.2
0
5
10
15
20
t [s]

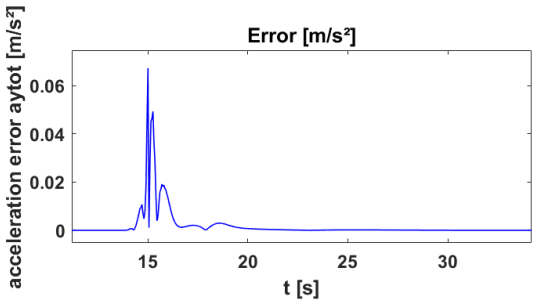

vY [m/s]
velocity Y [m/s]
0
-0.02
-0.04
15
20
25
30
35
t [s]

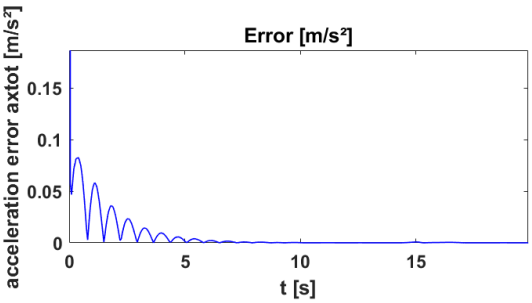

Error [m/s]
velocity error X [m/s]
0.03
0.02
0.01
0
0
5
10
15
20
t [s]

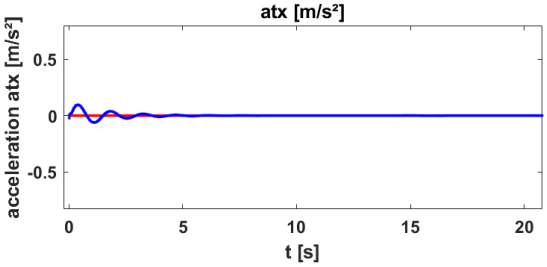

Error [m/s]
×10-3
velocity error Y [m/s]
6
4
2
0
15
20
25
30
35
t [s]

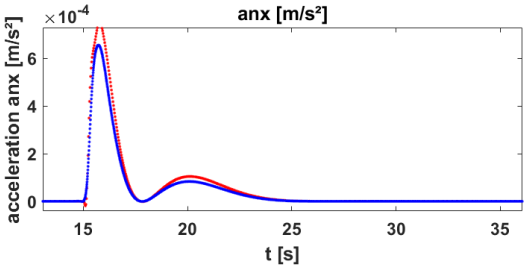

aytot [m/s²]
acceleration aytot [m/s²]
0.4
0.2
0
-0.2
15
20
25
30
t [s]

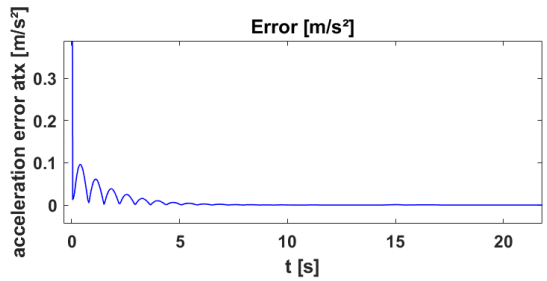

axtot [m/s²]
acceleration axtot [m/s²]
0.5
0
-0.5
0
5
10
15
20
t [s]

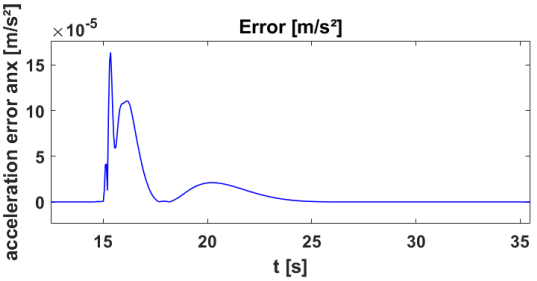

Error [m/s²]
acceleration error aytot [m/s²]
0.06
0.04
0.02
0
15
20
25
30
t [s]

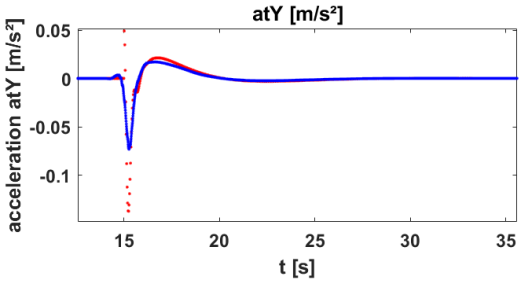

Error [m/s²]
acceleration error axtot [m/s²]
0.15
0.1
0.05
0
0
5
10
15
t [s]

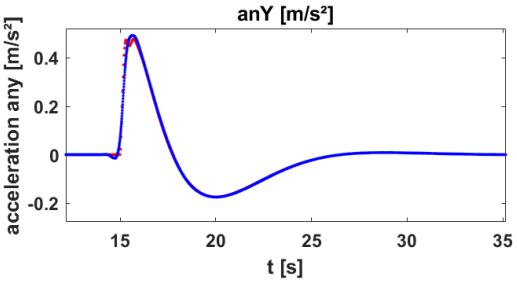

atx [m/s²]
acceleration atx [m/s²]
0.5
0
-0.5
0
5
10
15
20
t [s]

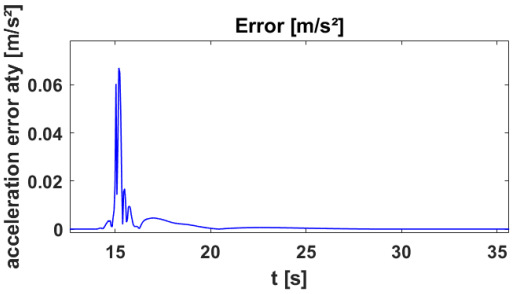

×10-4
anx [m/s²]
acceleration anx [m/s²]
6
4
2
0
15
20
25
30
35
t [s]

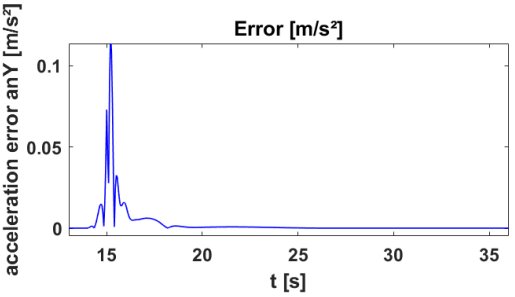

Error [m/s²]
acceleration error atx [m/s²]
0.3
0.2
0.1
0
0
5
10
15
20
t [s]

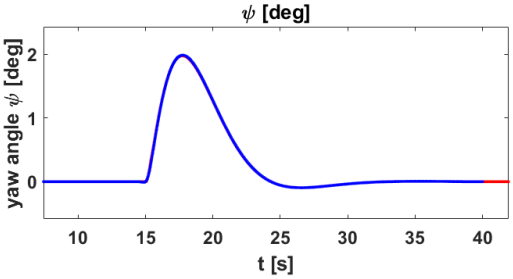

×10-5
Error [m/s²]
acceleration error anx [m/s²]
15
10
5
0
15
20
25
30
35
t [s]

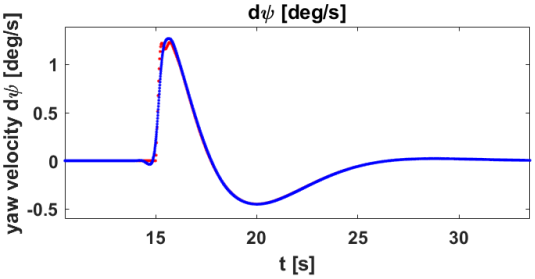

atY [m/s²]
acceleration atY [m/s²]
0.05
0
-0.05
-0.1
15
20
25
30
35
t [s]

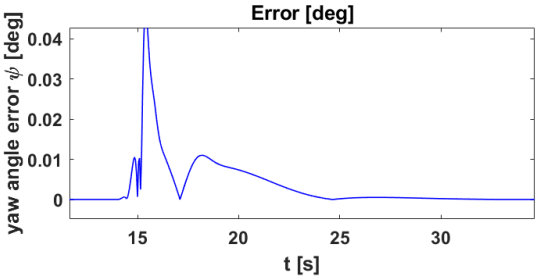

anY [m/s²]
acceleration any [m/s²]
0.4
0.2
0
-0.2
15
20
25
30
35
t [s]

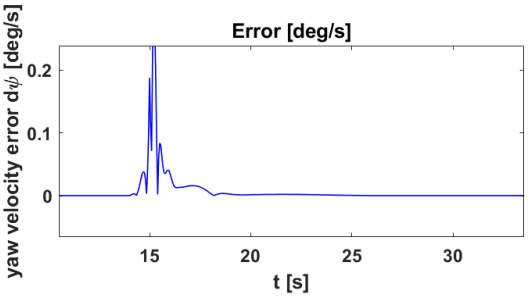

Error [m/s²]
acceleration error aty [m/s²]
0.06
0.04
0.02
0
15
20
25
30
35
t [s]

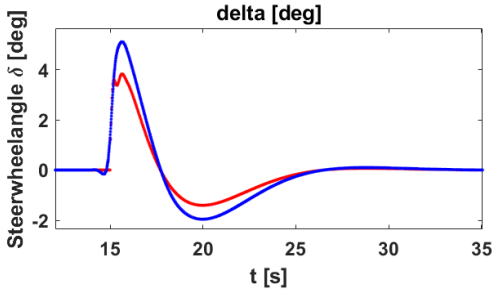

Error [m/s²]
acceleration error anY [m/s²]
0.1
0.05
0
15
20
25
30
35
t [s]

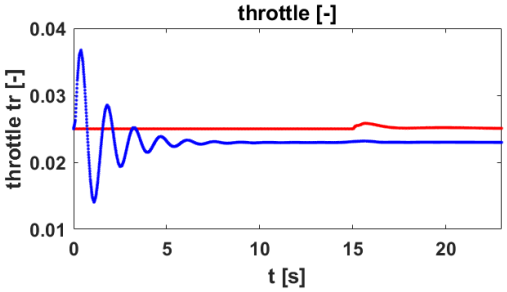

ψ [deg]
yaw angle ψ [deg]
2
1
0
10
15
20
25
30
35
40
t [s]

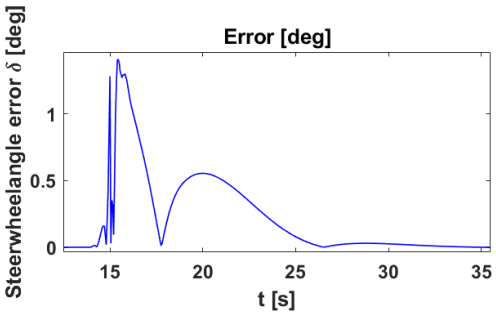

dψ [deg/s]
yaw velocity dψ [deg/s]
1
0.5
0
-0.5
15
20
25
30
t [s]

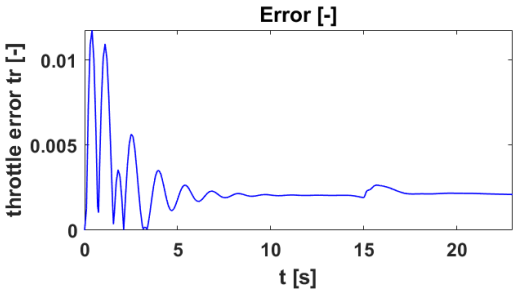

Error [deg]
yaw angle error ψ [deg]
0.04
0.03
0.02
0.01
0
15
20
25
30
t [s]

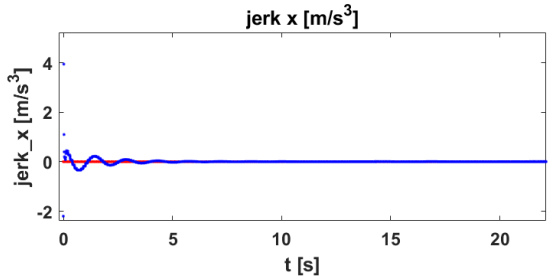

Error [deg/s]
yaw velocity error dψ [deg/s]
0.2
0.1
0
15
20
25
30
t [s]

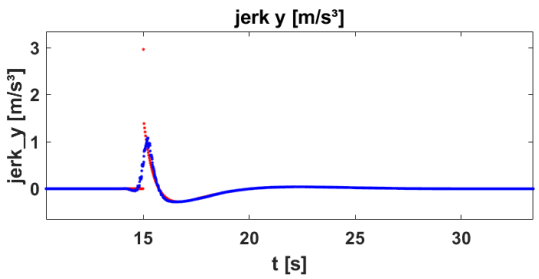

delta [deg]
Steerwheelangle δ [deg]
t [s]

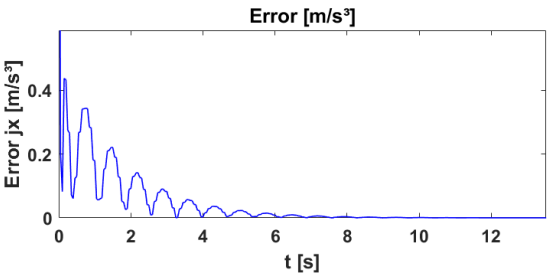

throttle [-]
throttle tr [-]
t [s]

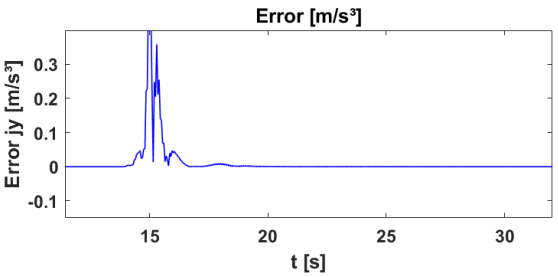

Error [deg]
Steerwheelangle error δ [deg]
t [s]

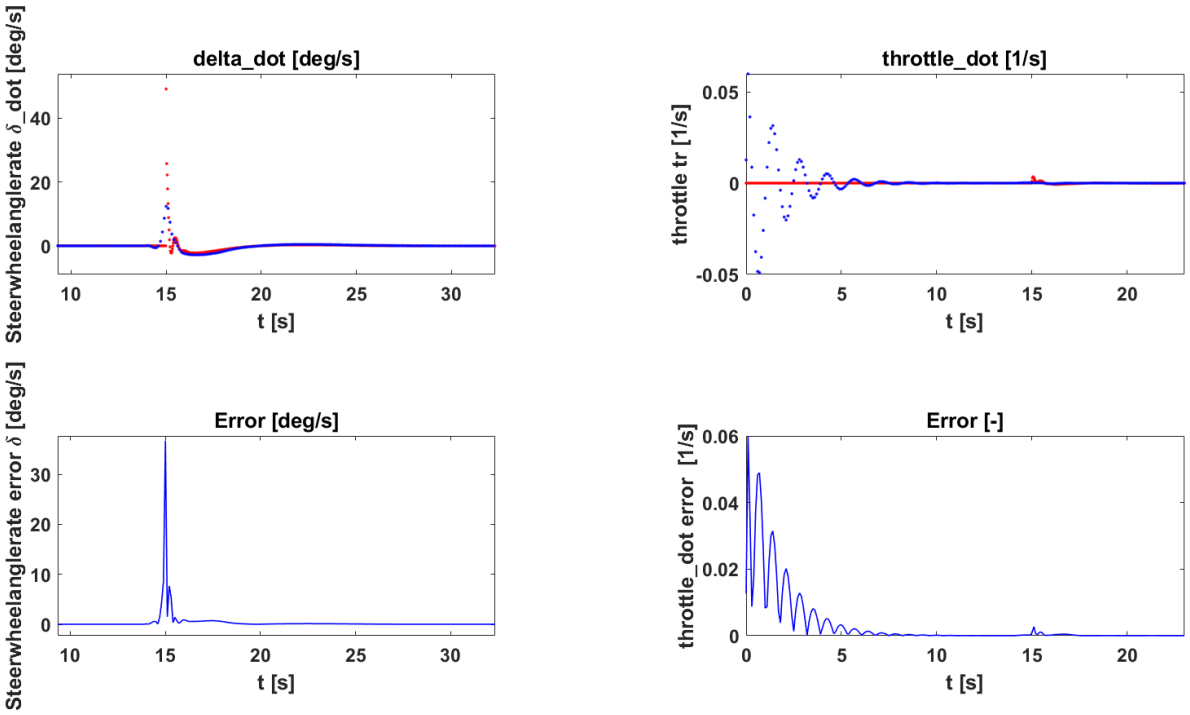

Error [-]
throttle error tr [-]
t [s]


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]

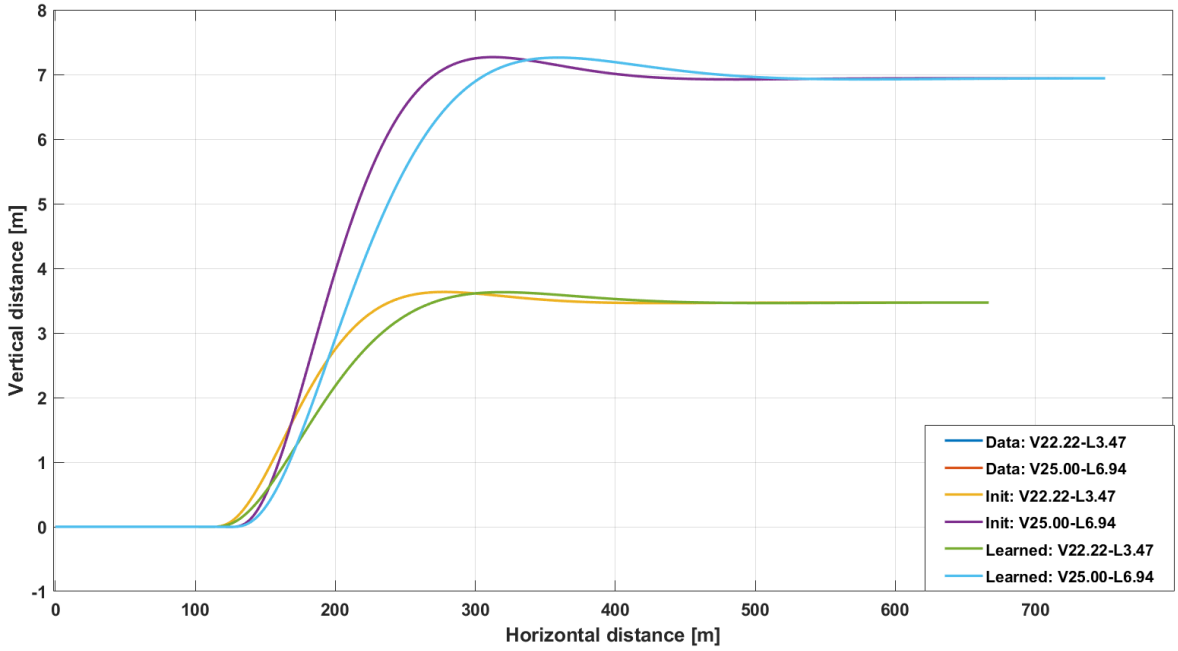

Error [m/s³]
Error jy [m/s³]
t [s]

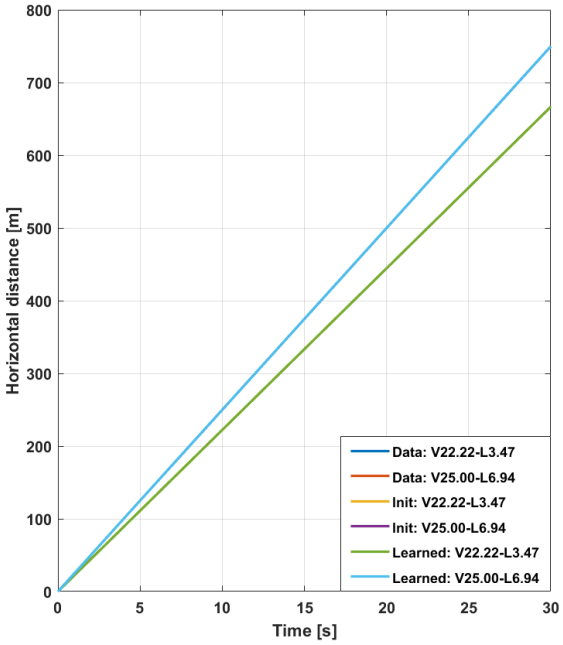

delta_dot [deg/s]
Steerwheelanglerate δ_dot [deg/s]
t [s]
throttle_dot [1/s]
throttle tr [1/s]
t [s]
Error [deg/s]
Steerwheelanglerate error δ [deg/s]
t [s]
Error [-]
throttle_dot error [1/s]
t [s]

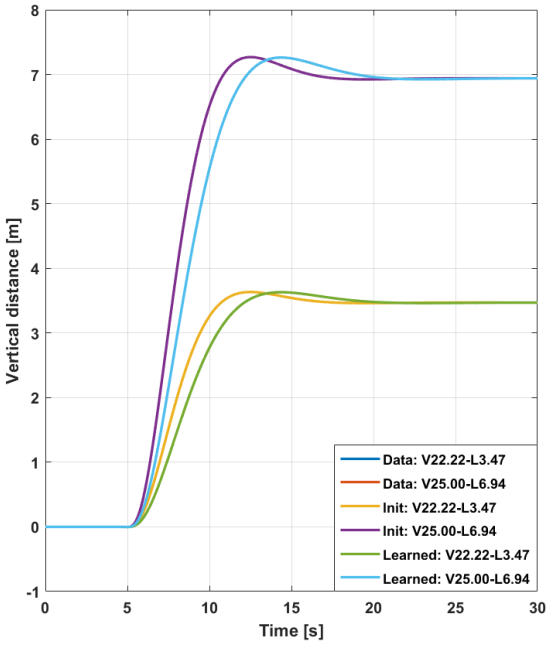
Appendix E

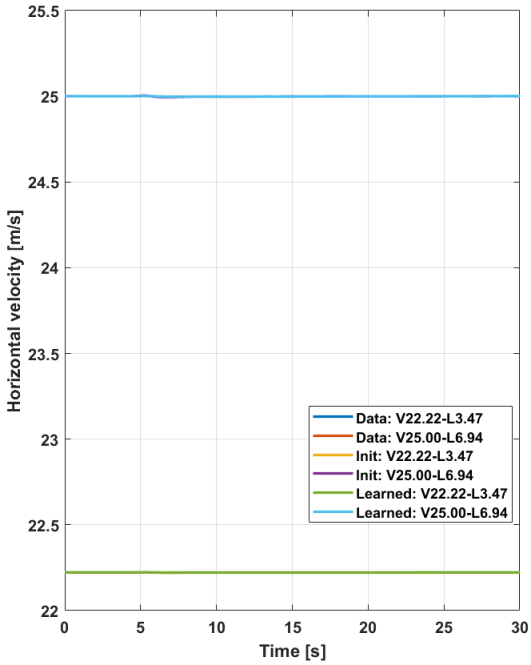
Results of learning with the 15 dof Amesim model

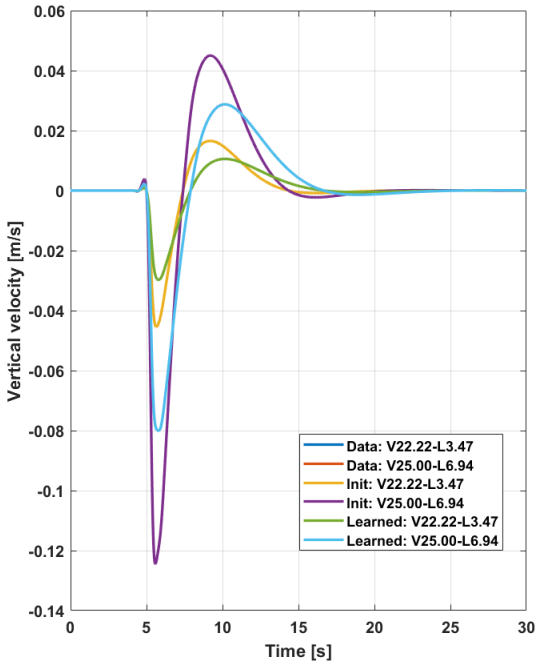
This appendix shows the learning results of the 15 dof Amesim model on two distinct as possible datasets $V022.22-L3.47$ and $V025.00-L6.94$. Except for the longitudinal total acceleration and jerk, all the learned kinematic signals match closely with the observed one. The most important conclusions about the results can be find in section 5.3.
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.

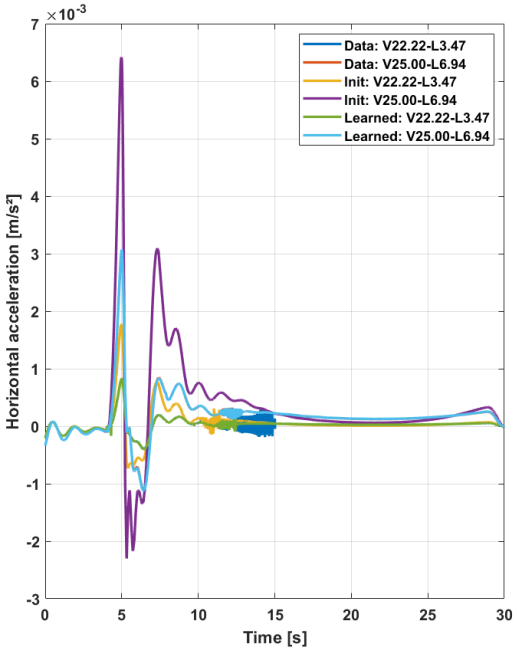

Vertical distance [m]
Horizontal distance [m]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94

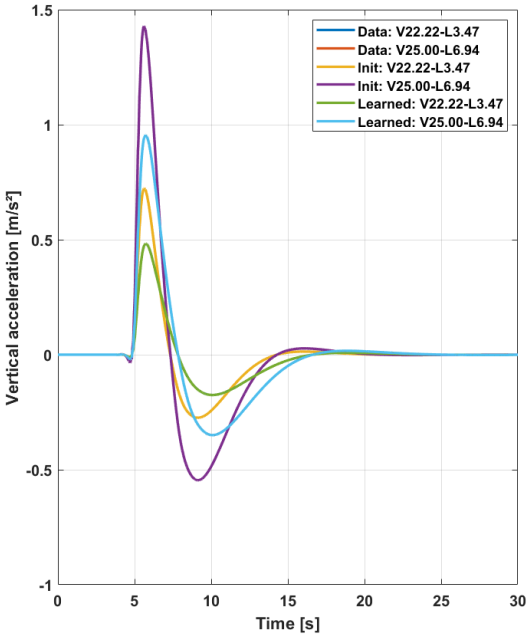

Horizontal distance [m]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94

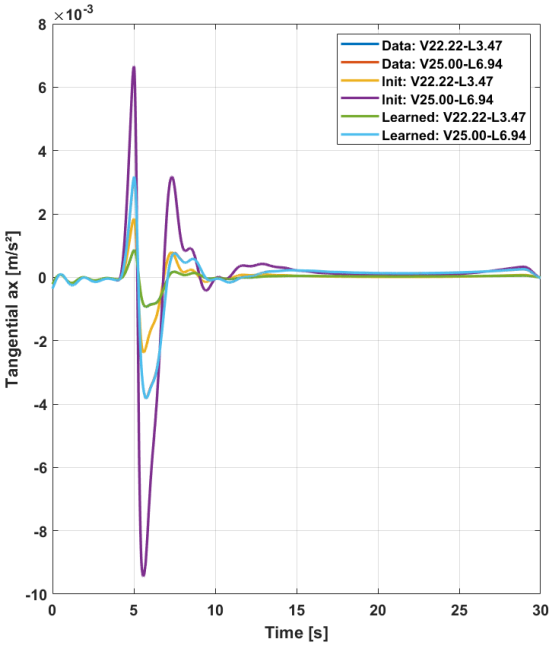

Vertical distance [m]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94

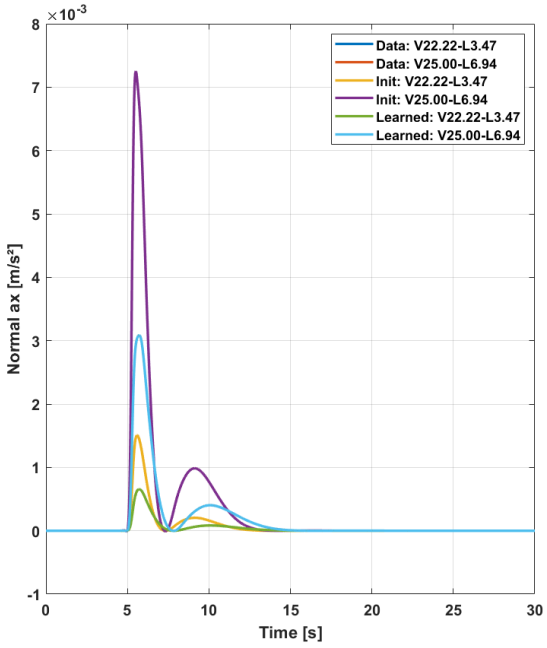

Horizontal velocity [m/s]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94

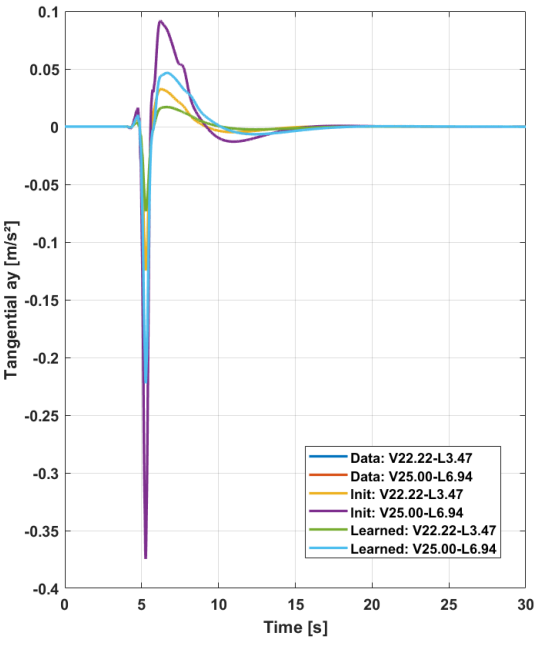

Vertical velocity [m/s]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94

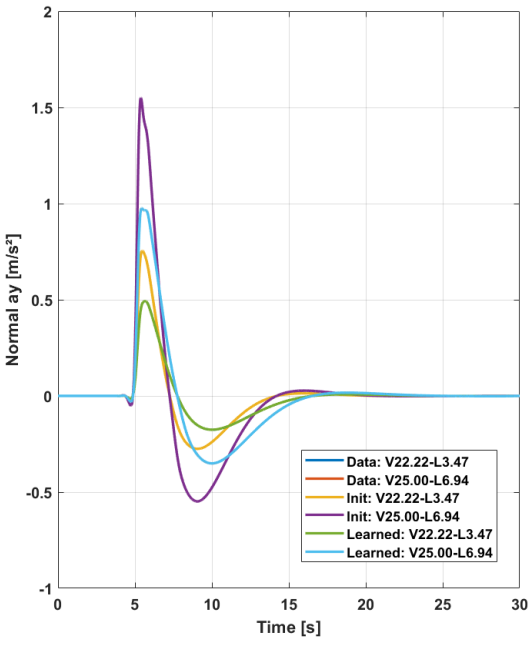

×10-3
Horizontal acceleration [m/s²]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94

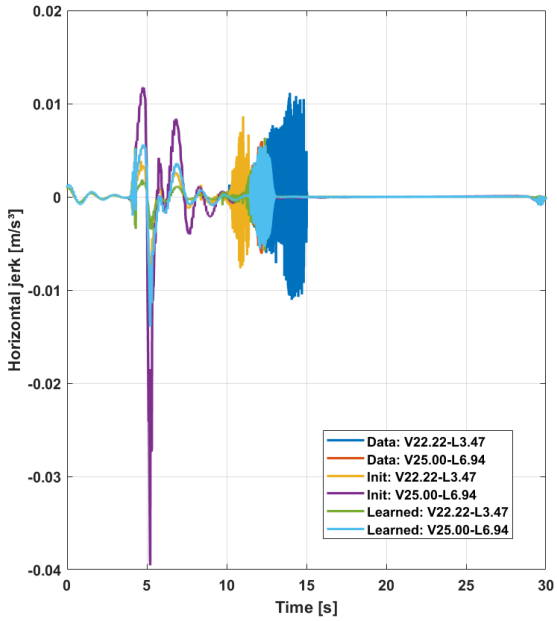

Vertical acceleration [m/s²]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94

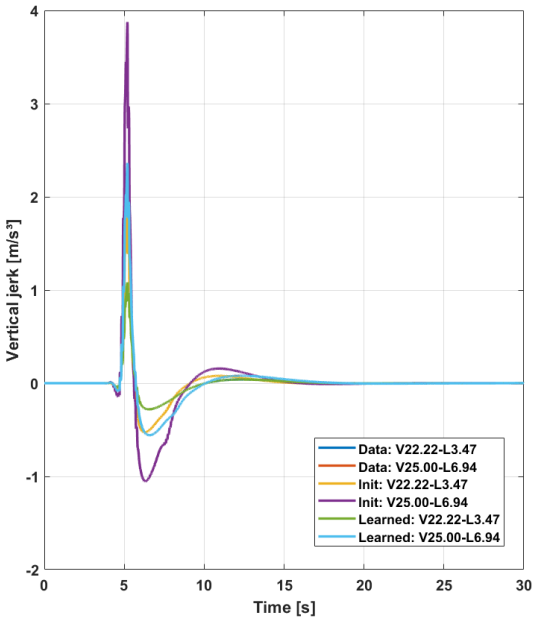

×10^-3
Tangential ax [m/s²]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94

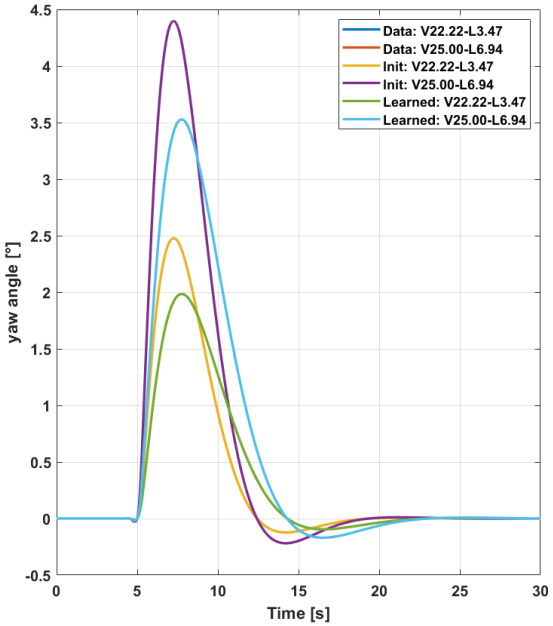

×10^-3
Normal ax [m/s²]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94

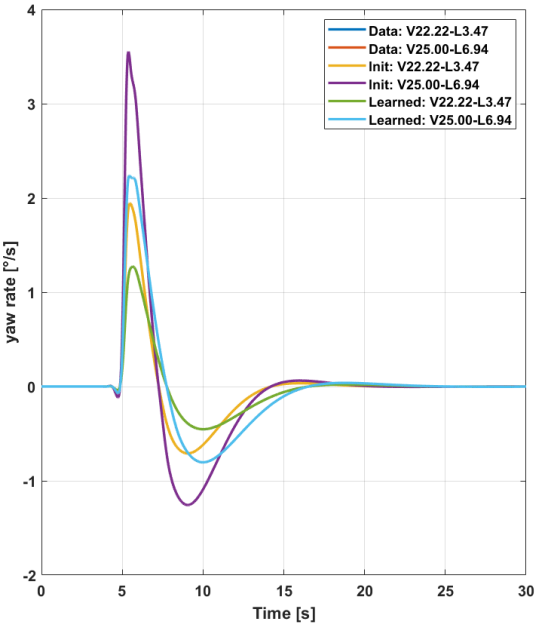

Tangential ay [m/s²]
Time [s]
Data: V22.22-L3.47
Data: V25.00-L6.94
Init: V22.22-L3.47
Init: V25.00-L6.94
Learned: V22.22-L3.47
Learned: V25.00-L6.94


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

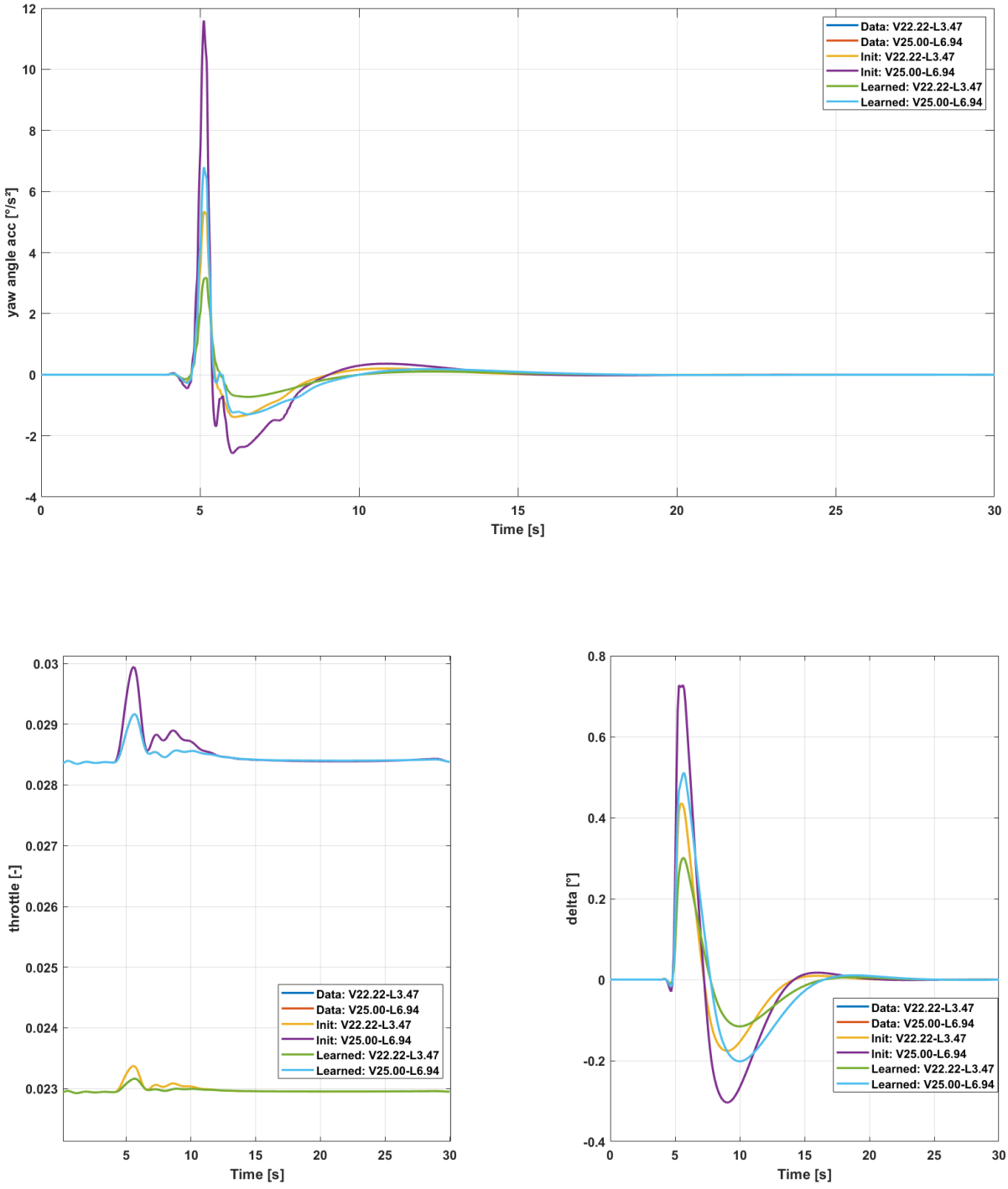


Figure E.1: This figure shows the amount of throttle and the angle of the front wheel of the bicycle model during the lane change maneuvers.

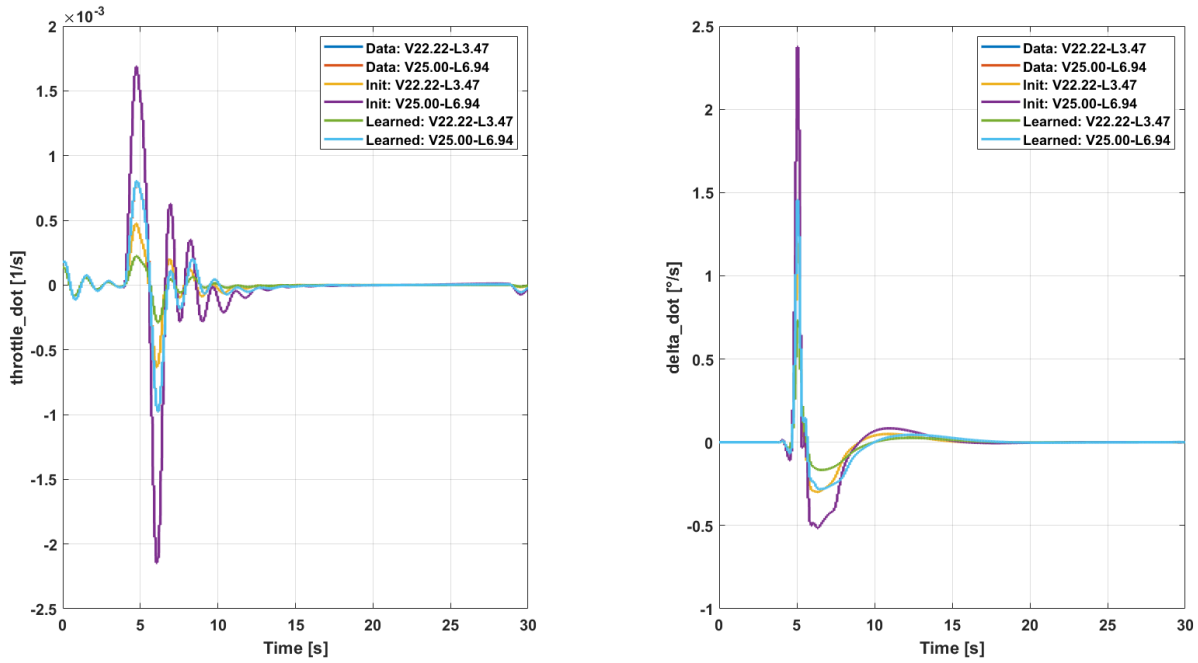


Figure E.2: This figure shows the amount of first derivative of throttle and the first derivative of the angle of the front wheel of the bicycle model is given as input during the lane change maneuvers.

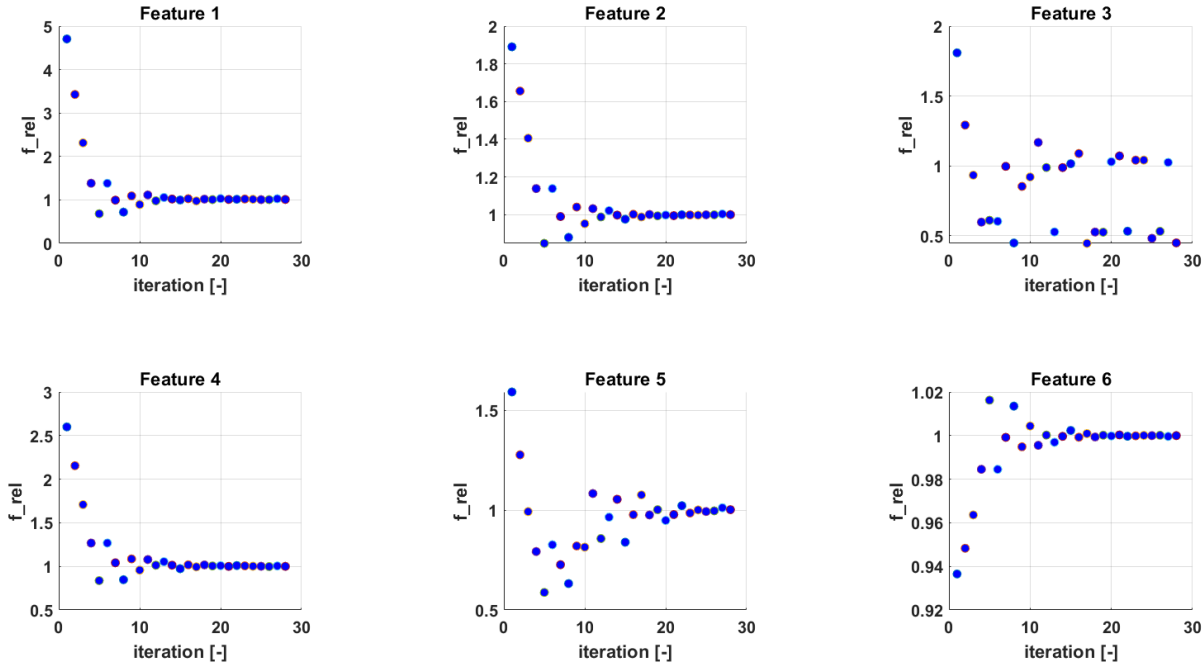


Figure E.3: The convergence of $\boldsymbol{f}_{rel}$ towards one during the learning iterations.

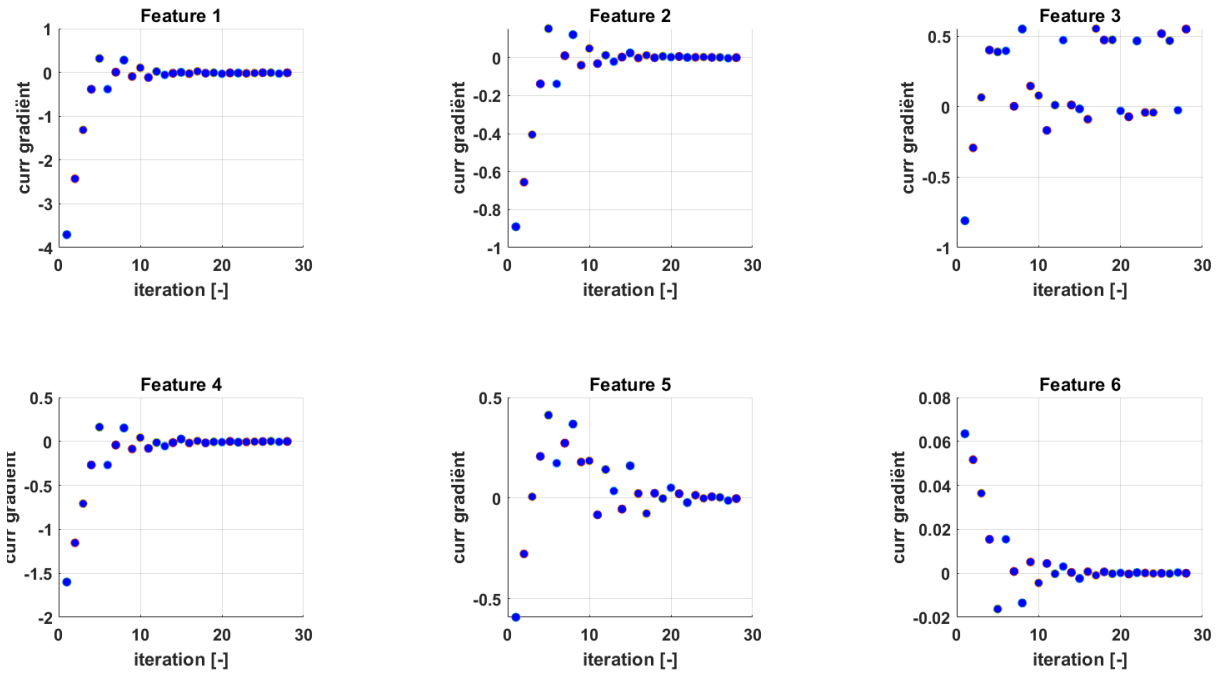


Figure E.4: The gradient $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\theta}}$ estimated by $\boldsymbol{F}_{obs} - \boldsymbol{F}(\boldsymbol{r}_{expected})$ towards zero during the different learning iterations.

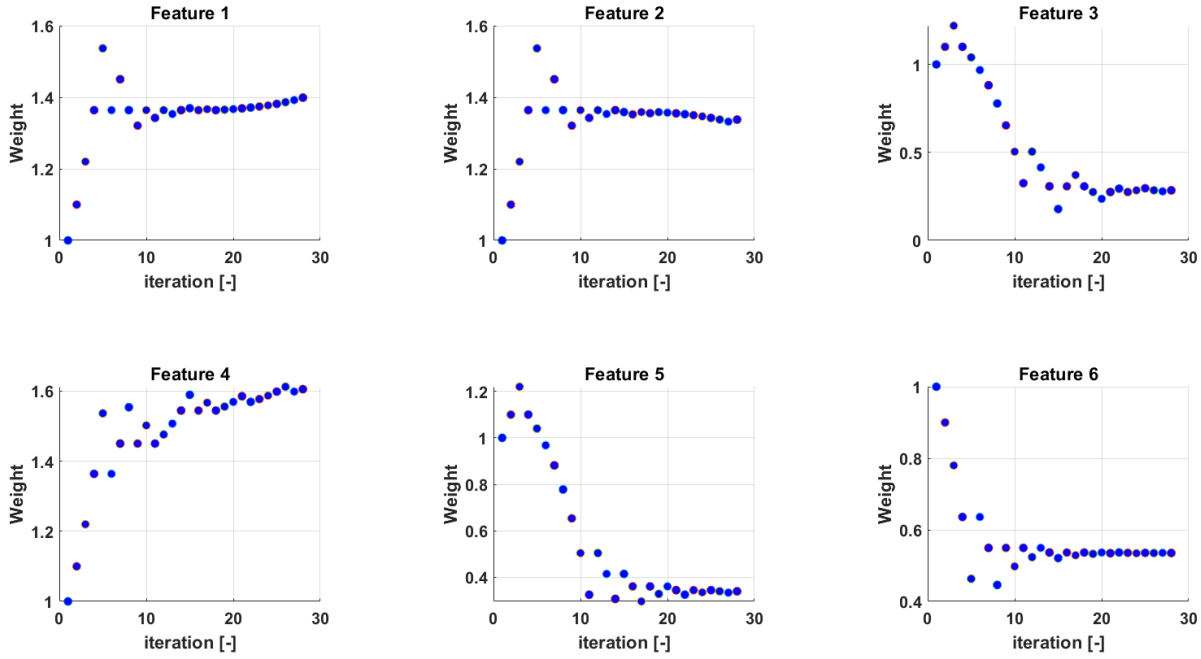


Figure E.5: The learned weights during the different learning iterations.

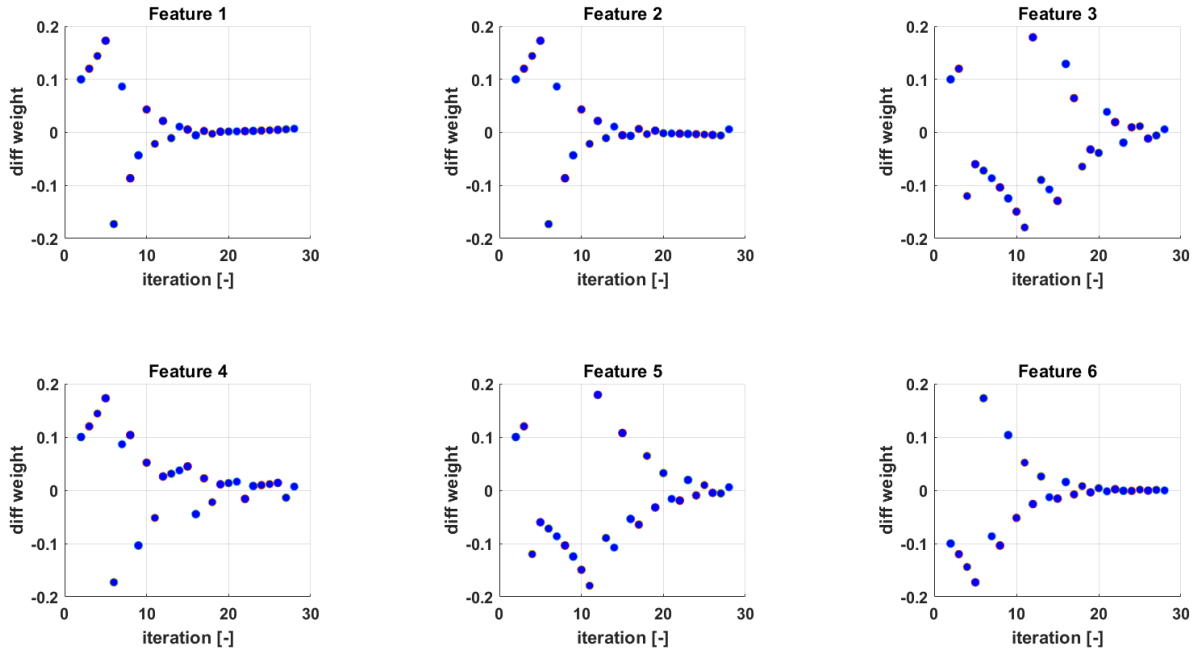


Figure E.6: The difference of $\boldsymbol{\theta}$ with respect to one used in the previous iteration.

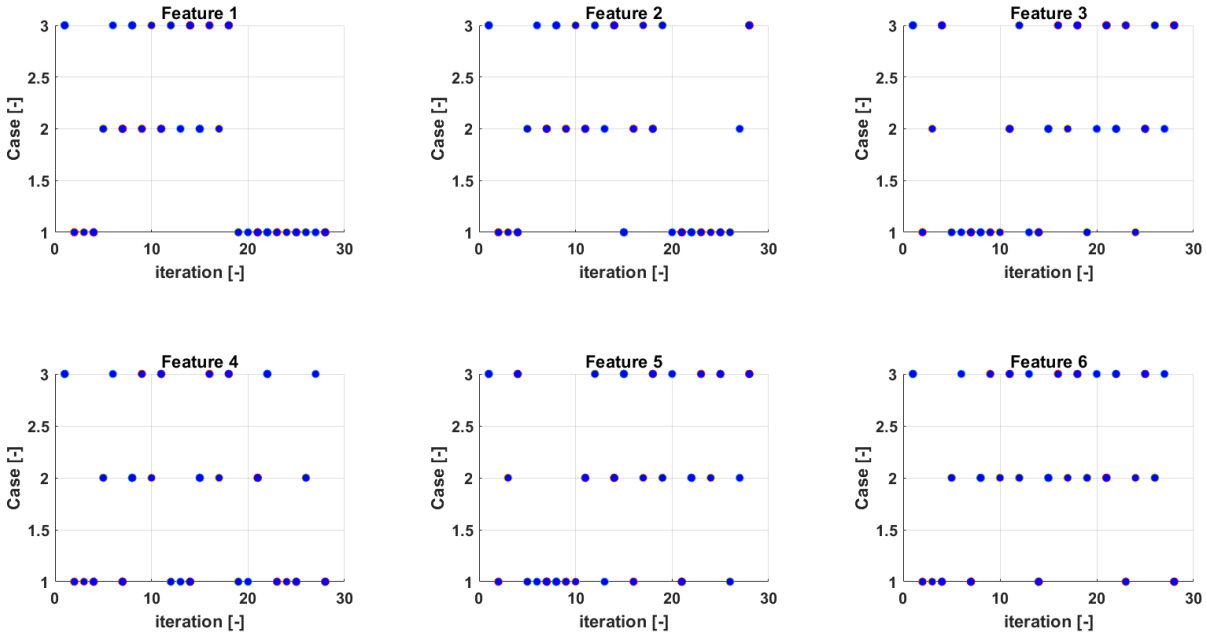


Figure E.7: This figure shows which case of the three available in the RPROP algorithm is chosen during the learning iterations.

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

[21] 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.