Janaína Ribas de Amaral

MELHORIA DA ESTABILIDADE DE VEÍCULO USANDO APRENDIZADO POR REFORÇO

Dissertação submetida como requisito final para a obtenção do título de mestre em Engenharia e Ciências Mecânicas pela Universidade Federal de Santa Catarina e em Engenharia Automotiva pela Technische Hochschule Ingolstadt em regime de cotutela.
Orientador: Prof. Dr. Thiago Antonio Fiorentin
Coorientador: Prof. Dr. Harald Göllinger

Joinville
2019

Janaína Ribas de Amaral

IMPROVEMENT OF VEHICLE STABILITY USING REINFORCEMENT LEARNING

Master's Thesis submitted as a final requirement for obtaining the master's degree in Engineering and Mechanical Sciences by the Federal University of Santa Catarina and in Automotive Engineering by the Technische Hochschule Ingolstadt under cotutela.
Advisor: Prof. Dr. Thiago Antonio Fiorentin
Co-advisor: Prof. Dr. Harald Göllinger

Joinville
2019

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Amaral, Janaina Ribas
Melhoria da estabilidade de veículo usando aprendizado por reforço / Janaina Ribas Amaral ; orientador, Thiago Antonio Fiorentin, coorientador, Harald Göllinger, 2019.
94 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Campus Joinville, Programa de Pós Graduação em Engenharia e Ciências Mecânicas, Joinville, 2019.

Inclui referências.

1. Engenharia e Ciências Mecânicas. 2. Aprendizado por reforço. 3. Controle de estabilidade. 4. Vetorização de Torque. I. Fiorentin, Thiago Antonio. II. Göllinger, Harald. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia e Ciências Mecânicas. IV. Titulo.

Janaína Ribas de Amaral

MELHORIA DA ESTABILIDADE DE VEÍCULO USANDO APRENDIZADO POR REFORÇO

Esta Dissertação foi julgada adequada para obtenção do Título de "mestre", e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia e Ciências Mecânicas da Universidade Federal de Santa Catarina e pela Faculty of Electrical Engineering and Computer Science da Technische Hochschule Ingolstadt (Alemanha).
Área de concentração: Desenvolvimento de Sistemas de Engenharia

Joinville, 24 de janeiro de 2019.

_____
Prof. Régis Kovacs Scalice, Dr.
Coordenador do Programa de Pós-Graduação em Engenharia e Ciências Mecânicas

Banca Examinadora:

_____
Prof. Thiago Antonio Fiorentin, Dr.
Universidade Federal de Santa Catarina - Orientador

_____
Prof. Harald Göllinger, Dr.
Technische Hochschule Ingolstadt – Coorientador (Videoconferência)

_____
Prof. Kleber Vieira De Paiva, Dr.
Universidade Federal de Santa Catarina

_____
Prof. Moisés Ferber de Vieira Lessa, Dr.
Universidade Federal de Santa Catarina

_____
Prof. Andrea Piga Carboni, Dr.
Universidade Federal de Santa Catarina

I dedicate this work to my family, my boyfriend and friends.

ACKNOWLEDGMENT

Firstly, I would like to thank God, for always being present in my life helping me to overcome difficulties.

I would like to express my special thanks to my advisors Thiago Antonio Fiorentin and Harald Göllinger for their dedication, patience, advice, for helping me to solve the problems that have appeared during this work and for sharing their knowledge with me.

I thank UFSC and THI for offering the double master's degree and FAPESC for providing the scholarship to make it possible.

I would like to show gratitude to Mrs Ebenbeck for helping me during the development of this thesis and to the team Schanzer that provided the necessary information for the simulation of the vehicle.

I must express thanks to my parents and my boyfriend Guilherme for supporting me and for having encouraged me always to try to do my best.

I also would like to thank my friends, especially Clarice that has helped me since we were approved for the double master's degree and that shared with me the good moments as well as the difficult ones of this abroad experience.

Finally, I would like to express my gratitude to my grandfather Lourival (in memoriam), for being my first teacher.

“Nobody phrases it this way, but I think that artificial intelligence is almost a humanities discipline. It’s really an attempt to understand human intelligence and human cognition.”
(Sebastian Thrun)

RESUMO

Esta dissertação de mestrado apresenta um estudo sobre o uso do aprendizado por reforço para controlar a vetorização de torque de um pequeno carro elétrico de corrida, a fim de melhorar o manuseio e a estabilidade do veículo. Para isso, utilizou-se o algoritmo de aprendizado por reforço Neural Fitted Q Iteration e realizaram-se amostragens do comportamento do veículo por meio de simulações com o software CarMaker. A função de custo utilizada é baseada na posição dos estados no plano de fase do ângulo de deriva e da velocidade angular de deriva. Para investigar a razão máxima de distribuição de torque que deve ser ajustada para garantir a estabilidade, bem como investigar a eficácia das entradas do controlador no processo de aprendizagem, realizaram-se dois experimentos (A e B) com diferentes estados e diferentes possibilidades de distribuição de torque. Os controladores resultantes são capazes de melhorar o manuseio e a estabilidade do veículo com uma redução significativa no ângulo de deriva do veículo. Porém, o controlador A apresentou melhor desempenho com relação à redução do ângulo de deriva e mostrou que a razão de distribuição de torque de 70% é suficiente para manter o veículo estável.

Palavras-chave: Controle de estabilidade. Aprendizado por Reforço. Vetorização de Torque. Neural Fitted Q Iteration.

RESUMO EXPANDIDO

Introdução

Os requisitos de segurança veicular estão sempre aumentando (VDA, 2018; CARRO NO BRASIL…,2017). Para cumprir com esses requisitos, muitos sistemas estão sendo criados, tanto para melhorar a segurança passiva quanto a ativa. Em paralelo, a área de inteligência artificial está crescendo e ganhando importância (LEE, 2017; COLUMBUS, 2017). De acordo com PwC (2017), os sistemas de assistência ao condutor são um campo de significativo potencial para a aplicação de inteligência artificial. Um exemplo de método de inteligência artificial que já está sendo usado para melhorar a segurança ativa é o aprendizado por reforço. Mas a aplicação desse método ainda pode ser aumentada. Portanto, essa dissertação de mestrado apresenta uma nova aplicação com o uso de um controlador baseado em aprendizado por reforço para melhorar a estabilidade e manuseio de um pequeno veículo elétrico de corrida, por meio do controle da vetorização do torque. Para isso, implementou-se o algoritmo Neural Fitted Q Iteration - NFQ (RIEDMILLER, 2005); foram realizadas simulações para gerar amostras do comportamento do veículo no programa CarMaker; a função de custo foi definida com base na posição dos estados no plano de fase do ângulo de deriva e da velocidade angular de deriva. Para investigar a razão máxima de distribuição de torque que deve ser ajustada para garantir a estabilidade, bem como investigar a eficácia das entradas do controlador no processo de aprendizagem, realizaram-se dois experimentos (A e B) com diferentes estados e diferentes possibilidades de distribuição de torque.

Objetivos

O objetivo dessa dissertação era apresentar uma nova aplicação do aprendizado por reforço em segurança ativa com o desenvolvimento de um controlador para melhorar o manuseio do veículo e a estabilidade do veículo, controlando a vetorização de torque de um pequeno carro elétrico de corrida. Portanto, foi necessário criar um controlador capaz de analisar a resposta do veículo a uma entrada do motorista e verificar se é necessário corrigir o comportamento do veículo, alterando a distribuição de torque nas rodas traseiras, a fim de manter a estabilidade. Os objetivos específicos eram: implementar um algoritmo de aprendizado por reforço, gerar dados para treinamento por meio de simulações do comportamento do veículo, realizar o treinamento para gerar o modelo do controlador e avaliar os resultados do processo de aprendizagem.

Metodologia

Diferente dos controladores de estabilidade tradicionais que dependem de um modelo matemático do veículo, o controlador baseado no aprendizado por reforço aprende a correta distribuição de torque por meio de interações com o ambiente, observando estados, tomando ações e recebendo recompensas e penalidades. Essas informações de estados, ações e recompensas ou penalidades são armazenadas e utilizadas para alimentar um algoritmo que realiza o processo de aprendizagem. Nesse trabalho, o algoritmo escolhido foi o Neural Fitted Q Iteration que utiliza uma rede neural para representar a função de valor da ação. Como entrada do controlador, ou estados, foram escolhidos alguns parâmetros que representam o comportamento longitudinal e lateral do veículo, como a aceleração longitudinal, a taxa de guinada, a velocidade, bem como o ângulo do volante. Uma vez que foi possível realizar dois tipos de experimento, o Experimento A usou como entrada a velocidade absoluta e o B usou as componentes longitudinais e lateral. As possíveis ações do controlador foram definidas como a porcentagem de torque que vai para a roda esquerda. Para cada experimento foram definidos valores diferentes, o A poderia variar de 30% a 70% e o B de 10% a 90%. Como função de custo, função que representa a relação entre as penalidades ou custos com a transição de um estado para outro por meio de uma ação, determinou-se o uso da análise do plano de fase do ângulo de deriva e da velocidade angular de deriva feita por He (2005). De acordo com essa análise, o plano de fase foi dividido em três regiões que determinam a estabilidade do veículo e custos diferentes foram definidos para cada região. A função de valor da ação foi representada por uma rede neural com duas camadas ocultas já que o espaço amostral do estado é grande. A política de tomada de ações foi determinada como ε-greedy, significando que uma porcentagem das ações seria tomada de acordo com a mínima função de valor da ação e uma pequena porcentagem seria tomada randomicamente. Para realizar as amostragens de experiência, foram realizadas simulações do comportamento do veículo usando o método dinâmico multicorpos no programa CarMaker. A manobra selecionada para a simulação foi a Sine with Dwell utilizada pela UN ECE na aprovação do sistema eletrônico de controle de estabilidade em veículos de passageiros. Uma vez que a simulação era realizada, as informações dos estados, ações e custos eram coletadas e armazenadas em uma memória, em seguida, para uma porcentagem desses dados calculava-se a função de valor da ação, então, uma nova rede neural era treinada e a antiga substituída.

Resultados e discussões

No final do processo de aprendizagem, dois controladores foram criados. Para avaliar o desempenho dos controladores, simulações do comportamento do veículo foram realizadas para situações em que o veículo estaria em condição instável e próxima da instabilidade. Essas simulações foram feitas considerando o veículo com e sem controle de estabilidade. Os resultados demonstraram redução do pico do ângulo de deriva tanto para o controlador do Experimento A quanto do Experimento B. Mas, o Controlador A apresentou maior redução do ângulo de deriva. Para ambos os controladores se observou a estabilização da taxa de guinada após o fim da entrada do ângulo do volante, o que valida a função de custo selecionada. No entanto, verificou-se que essa função deveria ser melhorada para regiões estáveis, já que o controlador seleciona valores de distribuição de torque diferentes de 50% quando o veículo está estável.

Considerações finais

Conclui-se que os controladores baseados em aprendizado por reforço foram capazes de melhorar a estabilidade e manuseio de veículo. Como o controlador do Experimento A resultou em maior redução no pico do ângulo de deriva do veículo, isso pode significar que é melhor utilizar como entrada a velocidade absoluta ao invés das suas componentes. Observou-se também que 70% como máximo torque que pode ser distribuído para as rodas é suficiente para manter o veículo estável. Uma vez que o controlador não utiliza o ângulo de deriva como entrada, pode-se dizer que se gerou um controlador simples. De acordo com os resultados, conclui-se que a função de custo, as entradas e a arquitetura da rede neural foram satisfatórias para captar o comportamento não linear do veículo que caracteriza a estabilidade.

Palavras-chave: Controle de estabilidade. Aprendizado por Reforço. Vetorização de Torque. Neural Fitted Q Iteration.

ABSTRACT

This master's thesis presents a study on the use of reinforcement learning to control the torque vectoring of a small electric race car in order to improve vehicle handling and vehicle stability. For this, the Neural Fitted Q Iteration algorithm is used, and the sampling of experiences is done using simulations of the vehicle behaviour in the software CarMaker. The cost function is based on the position of the states on the phase-plane of sideslip angle and sideslip angular velocity. To investigate the maximum ratio of torque distribution that should be set to guarantee stability as well as to investigate the effectiveness of the controller inputs in the learning process, two experiments were done (A and B) with different states and different possibilities of torque distribution. The resulting controllers are able to improve the vehicle handling and stability with a significant reduction in the vehicle sideslip angle. However, controller A presented better performance regarding the reduction of the sideslip angle and showed that 70% as the maximum ratio for the torque distribution is enough to keep the vehicle stable.

Keywords: Stability Control. Reinforcement Learning. Torque Vectoring. Neural Fitted Q Iteration.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AFR – Active front steering
AI – Artificial intelligence
DP – Dynamic programming
ESC – Electronic stability control
GNSS – Global navigation satellite system
MC – Monte Carlo Method
MDP – Markovian decision process
MLP – Multi-layer perceptron
NFQ – Neural Fitted Q Iteration
NN – Neural network
PCA – Principal component analysis
RL – Reinforcement learning
Rprop – Resilient backpropagation algorithm
SCG – Scaled conjugated gradient
SGD – Stochastic gradient descent
TD – Temporal differences method
TVD – Torque vectoring differential

LIST OF SYMBOLS

α	Slip angle	degrees
g	Gravitational acceleration	-
F_y	Lateral force in the tire	N
T_{Lock}	Torque distributed from left to right driveshaft, or vice versa	N.m
T_{input}	Input torque in the differential	N.m
M_z	Yaw moment	N.m
F_L	Longitudinal force in the left rear wheel	N
F_R	Longitudinal force in the right rear wheel	N
ΔF_x	Difference between F_L and F_R	N
t_w	Track width of the vehicle	m
b	Bias	-
x	Inputs of the neural network	-
x_s	Standardized inputs of the neural network	-
w	Weights of the neural network	-
φ	Activation function	-
y	Outputs of the neural network	-
e	Error	-
τ	Learning rate	-
γ	Discount factor	-
s	State	-
s'	Next state	-
a	Action	-
a'	Next action	-
π	Policy	-
R	Reward	-
V	State value function	-
Q	Action value function	-
P	Pattern set	-
D	Set of transitions	-
U	Matrix of eigenvectors of the covariance matrix of the variables	-

$\boldsymbol{z}$	Vector of principal components PC1 and PC2	-
$\boldsymbol{s}$	State vector	-
t	Time	-
β	Vehicle sideslip angle	degrees
$\dot{\beta}$	Vehicle sideslip angular velocity	degrees/s
$k_{\beta\dot{\beta}}$	Slope of the curves from reference region and stability error	-
c	Cost function	-
$TrqRatio$	Ratio of torque to be distributed to left rear wheel	-
T_L	Amount of torque that goes to the left rear wheel	N.m
T_R	Amount of torque that goes to the right rear wheel	N.m

TABLE OF CONTENTS

1 INTRODUCTION

The requirements for vehicle safety are always increasing (VDA, 2018; CARRO NO BRASIL…,2017). With the intention of fulfilling it, the engineers have developed different kinds of systems for passive safety from seat belts to airbags. Regarding the active safety, the phase before the crash, they have made use of controllers and sensors to develop driver assistance systems like autonomous emergency braking (AEB), electronic stability control (ESC) and traction control (VAHIDI; ESKANDARIAN, 2003). According to Guo et al. (2010), these systems work effectively in order to prevent crashes and many countries already consider them as required items of the vehicle.

At the same time, the field of artificial intelligence (AI) is gaining more and more power and recognition in several areas such as economics, medicine and especially engineering (LEE, 2017; COLUMBUS, 2017).

According to PwC (2017), driver assistance systems is an area of significant potential for the application of artificial intelligence. Indeed, many advances were made to improve the active safety of vehicles by using artificial intelligence — for example, advances using controllers based on reinforcement learning algorithms.

Reinforcement learning is a kind of machine learning used for goal-directed and decision problems (SUTTON; BARTO, 1998). It was already applied to improve longitudinal motion in Adaptive Cruise Control (DESJARDINS; CHAIB-DRAA, 2011; PIETQUIN; TANGO; ARAS, 2011; WEI et al., 2018) and to optimize rules of a fuzzy system that controls vehicle stability adding a yaw moment generated by differential braking (AKBARI; GOHARIMANESH, 2014).

These examples demonstrate improvement in active safety using reinforcement learning, but its application in this field can still be enlarged.

Thus, this master's thesis aims to present a new application proposing the use of a reinforcement learning controller to improve stability and handling of a small rear wheel driven electric vehicle by controlling the torque vectoring.

For this purpose, a reinforcement learning algorithm was implemented, based on the NFQ Reinforcement Learning algorithm (RIEDMILLER, 2005) to create the model of the controller. As these algorithms learn how to take decisions by trial and error, the algorithm learned how to control the torque vectoring experiencing interactions with the environment by means of simulations of the vehicle behaviour that are carried out using the software CarMaker. However, it was necessary

to set costs (penalties) for unstable or handling limit situations to help the algorithm understand when it is needed to add a yaw moment to keep the car stable.

One of the most important sets for the stability controllers is to define the threshold for the addition of the corrective yaw moment. Some controllers compare the current yaw rate of the vehicle with the desired yaw rate computed using a linear or nonlinear vehicle model (GOHARIMANESH; AKBARI, 2017; LEE; HWANG; SUH, 2015). Another approach is to define the handling limit using the phase-plane of sideslip angle and sideslip angular velocity. By looking at this phase-plane, it is possible to have a good characterization of the nonlinear dynamic behaviour of the vehicle. For this reason, it is used in many stability controllers that use brakes, torque vectoring or even a combination of both solutions (GUO et al., 2010; INAGAKI; KSHIRO; YAMAMOTO, 1994; HE, 2005; LU et al., 2016). In this work, the phase-plane of sideslip angle and sideslip angular velocity is used to define the cost function.

To investigate the influence of the selected inputs and actions in the performance of the controller, two experiments are performed, A and B, using different states as input and different values of torque ratio as action.

At the end, two preliminary handling and stability controllers are created, and they are able to select which percentage of torque should be distributed to the wheels when a steering input as Sine with Dwell is given.

Therefore, in this work, an introduction to lateral vehicle dynamics, neural network, machine learning and reinforcement learning is made. In the sequence, the process to create the controller is presented. Then, the learning results are shown and discussed.

1.1 OBJECTIVES

The objective of this master's thesis was to present a new application of reinforcement learning in active safety with the development of a controller to improve vehicle handling and vehicle stability, by controlling the torque vectoring of a small electric race car. For this, it was necessary to create a controller that is able to analyse a race car response to a driver input and verify if it is necessary to correct the car behaviour, by changing the torque distribution in the rear wheels, in order to keep the car stable.

1.1.1 Specific objectives

- Implement an algorithm based on reinforcement learning;
- Generate training data: acquisition of driver inputs like steering wheel angle and car response like speed, acceleration and yaw rate while performing simulations of manoeuvres;
- Run the training data into the algorithm to generate the model of the controller;
- Evaluate the results of the learning process.

2 LITERATURE REVIEW

In this chapter, it is introduced the concepts of lateral vehicle dynamics and handling limit, torque vectoring, neural network, machine learning and reinforcement learning, which are the support for this study. Moreover, a literature review about stability controllers and autonomous driving robots are presented since this was the basis for the development of the proposed controller.

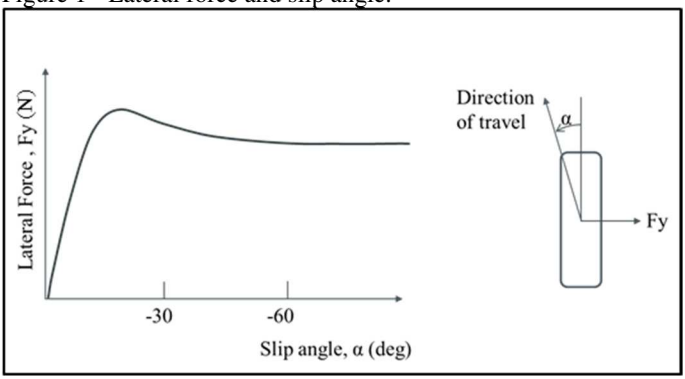
2.1 LATERAL VEHICLE DYNAMICS AND HANDLING LIMIT

When the driver needs to turn the vehicle in a situation of cornering or lane change, for example, he applies a steering angle input to the steering wheel. According to this input, the wheels turn in the desired direction.

According to Gillespie (1992), if the vehicle is at high speed, the tires must develop lateral forces to counteract the lateral acceleration that is present under this situation to control the direction of the vehicle. The generation of the lateral force is based on the lateral slip of the tire (slip angle), on lateral inclination (camber angle) or the two effects combined.

The side slip angle α is the angle between the direction of heading of the tire and its direction of travel (GILLESPIE, 1992), as can be seen in Figure 1. The lateral force generated by the tire is dependent on the slip angle. For small values of sideslip, the relation is linear. However, when the slip side angle increases and the lateral acceleration exceeds 0.3g, the relation becomes nonlinear (HE, 2005), as shown in Figure 1.

Figure 1 - Lateral force and slip angle.



Source: Gillespie (1992).

After a further increase in the side slip motion, the properties of the lateral tire forces saturate (HE, 2005) and the tire presents the behaviour of a locked wheel (GILLESPIE, 1992). The result is a yaw moment that leads to vehicle instability and spin (HE, 2005).

From this point, the vehicle will not respond to steering inputs. Therefore, it is the limit of handling, once that handling is the responsiveness of a vehicle to driver input (GILLESPIE, 1992).

In this section, it is also essential to introduce the concept of the vehicle sideslip angle which is a control parameter in many stability controllers. The vehicle sideslip angle is the angle between the speed and the longitudinal axis of the vehicle.

To control the vehicle and improve handling and stability, three types of systems can be used: active steering, brake-based and torque vectoring.

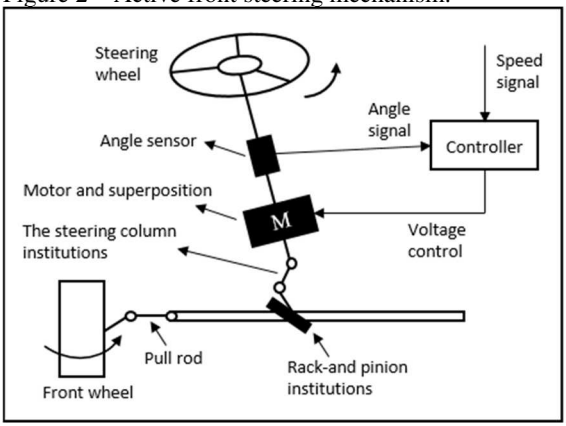
2.1.1 Active steering system

Both rear and front steering angle can be controlled to improve the vehicle dynamics. The active rear steering system controls the rear steering angle, and it is used to minimize the vehicle sideslip angle or to make the car follow a dynamic model. It is done by positioning the rear wheels in the opposite direction to the front wheels. At low speeds, it results in an improvement of manoeuvrability and at high speed enhancing of stability (HE, 2005).

The active front steering (AFS), according to Qinchao, Xuncheng and Fang (2011), is a system that provides an additional steering angle to the steering angle defined by the driver to control the yaw rate. This controller and a servo motor as actuator can be added to the traditional steering system, and there will be a superposition of the driver input and the controller output. The AFS mechanism is shown in Figure 2.

Active steering is an effective tool to improve handling (HE, 2005). However, according to Guo et al. (2010), when the lateral tire forces achieve their adhesion limits and the vehicle presents nonlinear behaviour, the AFS system becomes less effective.

Figure 2 – Active front steering mechanism.

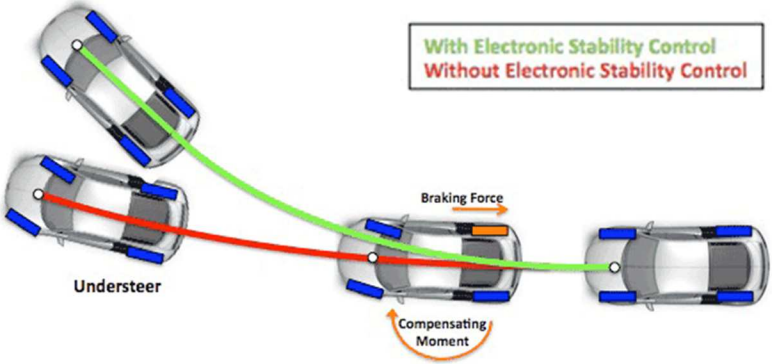


Source: Qinchao, Xuncheng and Fang (2011).

2.1.2 Brake-based system

The brake-based is the most common system used for stability control (HE, 2005). It acts by using the braking force of one or more wheels with the aim of creating an additional yaw moment that stabilizes the vehicle. Figure 3 illustrates the principle of operation of the brake-based system in a case of understeer.

Figure 3 – Principle of operation of the brake-based system.



Source: CVEL (2018).

The advantage of this solution is that it can share system and actuators with the Antilock Braking System (ABS) and that it can create

a high corrective yaw moment. However, its disadvantage is that it influences the longitudinal dynamics. Therefore, as stated by Guo et al. (2010), the brake-based system is indicated for emergency and it is not indicated for steady-state situations.

2.1.3 Torque vectoring

Torque vectoring is a technology used to distribute the torque that comes from the power supply in different values of torque for the wheels to improve traction performance and lateral vehicle dynamics (HE, 2005). In the case of lateral vehicle dynamics, the different torque distribution between left/right or front/rear wheels generates the yaw moment that controls the vehicle stability.

According to He (2005), the front/rear torque vectoring creates a yaw moment caused by difference in the magnitude of lateral forces in the rear and front wheels. Because when the longitudinal force is reduced, the lateral force is also reduced due to the tire property. For left/right distribution, the yaw moment is created as a result of the difference in the longitudinal force in the left and right wheels, as shown in Figure 4.

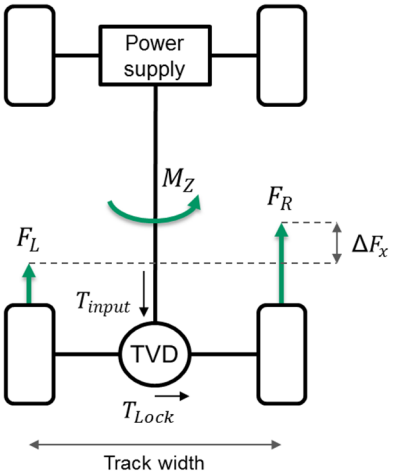
If the vehicle has the same number of driven wheels as electric motors/engine, torque vectoring is easily applied by a controlling system. In the case that the number of electric motors or engine is lower than the number of driven wheels, it is necessary that a locking differential distributes different torques to the wheels.

This torque vectoring differential (TVD) must provide the possibility of controlling the magnitude and the direction of the torque that will be transferred between the wheels, and it can be done by using friction clutches connected to the differential and the driveshafts (HANCOCK et al., 2005).

The TVD can be applied both for left/right and for front/rear torque vectoring. The left/right is more effective to control stability (HE, 2005). For this type, engaging the correct clutch, it is possible to control the torque T_{Lock} that should be distributed from left to right driveshaft, or vice and versa, to generate braking torque in one wheel and driving torque in the wheel on the other side. It creates a longitudinal force in the tires proportional to the torque and the wheel radius. So, the yaw moment M_z will be generated from the difference between the left and right longitudinal force ΔF_x according to Equation 1, where t_w is the track width of the vehicle.

$$M_z = \frac{t_w}{2}\Delta F_x \quad (1)$$

Figure 4 – Principle of operation of torque vectoring.



Source: Li and Wu (2011).

The advantage of this method is that it does not influence acceleration and deceleration tasks, in other words, it does not influence the longitudinal dynamics. It can be used for emergency situations and even to improve handling. Its disadvantage is that the magnitude of the generated yaw moment is lower than the yaw moment generated by the brake-based system.

2.2 ARTIFICIAL NEURAL NETWORK

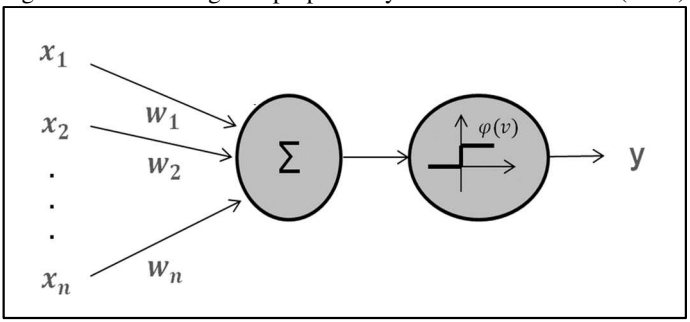
The artificial neural network is a technique that uses the principle of brain neurons to generate a mathematical model (TANCREDI, 2008; KIM, 2017).

The brain associates neurons in order to form information as the neuron itself is not able to perform the storage (KIM, 2017). In the same way, the artificial neural network associates nodes, which represent the neurons, by means of weighted connections, which represent the neurons connections, to store the acquired knowledge (TANCREDI, 2008).

The first mathematical model of the neuron was proposed by McCulloch and Pitts (1943). It represents the output as a function of the weighted sum of the inputs (Figure 5). Inspired in their model, Rosemblatt

(1958) introduced the idea of supervised learning and trial-and-error to update the weights, and it was called perceptron.

Figure 5 – Processing unit proposed by McCulloch and Pitts (1943).



Source: Kim (2017).

In the simple representation of the perceptron, the nodes of the input layer receive the inputs *x*, the inputs are weighted according to the respective connection, and in the output layer, they are added with a bias *b* (Equation 2). After that, this sum is used as independent variable on an activation function φ that provides the output *y* (Equation 3).

$$v = x_1 w_1 + x_2 w_2 + x_3 w_3 + b \tag{2}$$

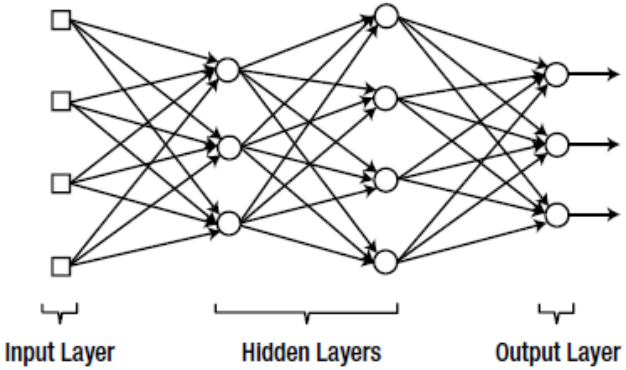
$$y = \varphi(v) \tag{3}$$

According to Haykin (2009), the most common activation functions are threshold logic, sigmoid, linear and rectified linear unit. The activation function is responsible to define if the neural network can solve nonlinear problems.

2.2.1 Layered structure

The artificial neural network can be organized in different types of structures; the mainly used is the layered structure of nodes showed in Figure 6. The first layer is the input layer, which only transmits information and does not carry out any calculation, the last one is the output layer, and the layers between them are the hidden layers (KIM, 2017).

Figure 6 – Layered structure.



Source: Kim, (2017, p.12).

According to Kim (2017), the number of hidden layers is important to classify the neural network, if there is no hidden layer it is called single layer neural network, if hidden layers are added it is called multi-layer neural network. However, there is a specific classification when there is one hidden layer (shallow neural network) and when there are two or more (deep neural network).

2.2.2 Creating the model

Once that the structure is defined, the model is carried out using a training data set containing *input* and *correct output* to perform supervised learning (this topic will be explained in 2.3).

Then, the weights are initialized (it can be done randomly), the inputs are given to the input layer, and the output of the neural network is compared with the *correct output*. The resulting error will be used to update the weight in such a way that the error is minimized. One example of methodology used in order to minimize the error is the gradient descent update represented by delta, δ, as presented in the Equation 4, 5 and 6. These equations are an example of a learning rule.

$$w_{ij} \leftarrow w_{ij} + \Delta w_{ij} \tag{4}$$

$$\Delta w_{ij} = \tau \delta_i x_j \tag{5}$$

$$\delta_i = \varphi'(v_i) e_i \tag{6}$$

where w_{ij} represents the weight between output node i and input node j, Δw_{ij} is the update, x_j is the output from the input node j, τ is the learning rate which can vary from 0 to 1 and indicates how fast the weight will change. $\varphi'(v_i)$ is the derivative of the activation function of the output node i and e_i is the error, that can be the error from the output node i or an average error. The value of Δw_{ij} is calculated for all training data.

In the case of multi-layer neural networks, the error seen in the output layer can be propagated backwards to allow the update of all weights from the neural network, this algorithm is called back-propagation.

2.2.3 Types of update for weight

The weights can be updated by three schemes using supervised learning:

- Stochastic gradient descent (SGD): the update is computed for each training data, and it is immediately incorporated to the weight (Kim, 2017);
- Batch: in this scheme, the errors for all training data are calculated, and then the average error is used to update the weight. It is made just one update for cycle (the cycle containing the calculation for all training data is also called epoch) (TANCREDI, 2008).
- Mini Batch: this scheme is the fusion of the SGD and Batch. The average error is calculated for groups of the training data, and the update is made using one group for step. Kim (2017) exemplifies this scheme in the following way: if the training data have 100 data points and groups of 20 data points are made, it means that in one epoch the weights are going to be updated five times.

According to Kim (2017), the Mini Batch combines the convergence speed of the SGD and the stability of the Batch. The number of epochs necessary to achieve an acceptable error will depend on which scheme is used.

Apart from gradient descent update (section 2.2.2), it is also possible the use of many other concepts as the resilient backpropagation (Rprop) and the scaled conjugate gradient algorithm (SCG). The Rprop was introduced by Riedmiller and Braun (1993). In the Rprop, each weight has its individual update based on the sign of the partial derivative of the error function with respect to the weight. It results in a faster

convergence once that one weight does not influence the update of the other. In the gradient descent, the update is global, and one update can be good for some weights but bad for others.

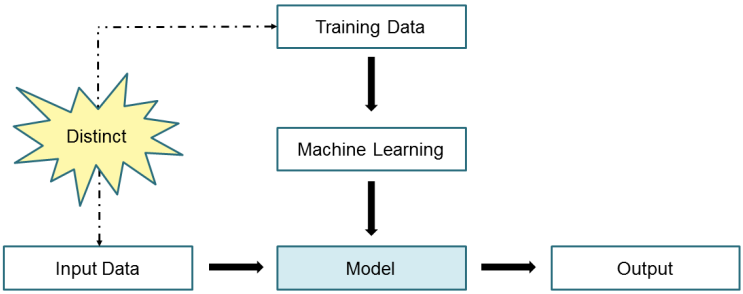
The scaled conjugate was presented by Møller (1993), and it is also an algorithm that aims to accelerate the convergence. The main idea is to update the weight in a direction that is a linear combination of the current gradient descent vector and the previous direction. Moreover, it eliminates the dependence on the user to set the learning rate, which is a parameter that influences the performance of the learning process (MØLER, 1993). The learning rate is calculated at each iteration using the information of the second order derivative of the error function with respect to the weight and the direction of search.

2.3 MACHINE LEARNING

Machine learning is a kind of artificial intelligence that has been used as a powerful tool to solve complex problems in different areas like engineering for autonomous driving, finance to predict the stock market and for medical professionals to carry out diagnoses (PALUSZEK; THOMAS, 2016).

As stated by Kim (2017), machine learning is a technique that generates a model from data, and learning means that the system finds the model itself. A training data is used in the learning process (vertical flow in Figure 7), after that, the resulting model is able to provide a reasonable answer for a given input (horizontal flow in Figure 7). It is important to mention that the training data must be different from the input data.

Figure 7 – Model generation.



Source: Kim, (2017, p. 6).

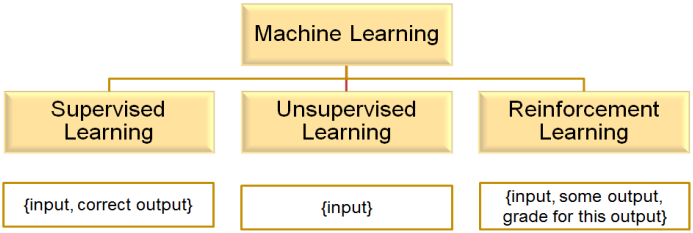
To evaluate the performance to guarantee the generation of a good model, previously unseen data (called validation data set) is given as input for the model, and it is verified the accuracy of the output, this process is called generalization. According to Kim (2017), if the model performance does not depend on the training data or in the input data, the model is generalized. The success of machine learning relies on how well the generalization is made.

The main aspect that influences the generalization is the overfitting, which is characterized by the correct match between training data output and model output. It means that the model learned noise from the training data and will present lower generalizability. Apart from the validation, models with simple structure are also used in order to avoid overfitting.

2.3.1 Types of machine learning

Regarding the method used for training the model, it is possible to classify machine learning in three types, as presented in Figure 8:

Figure 8 – Types of machine learning.



Source: Kim (2017, p. 12).

- Supervised learning: in this case, input and correct output are available to create the model. So, the parameters of the model are adjusted in such a way that the difference between the correct output and the model output is minimized (PALUSZEK; THOMAS, 2016; KIM, 2017).
- Unsupervised learning: this type usually is used to figure out patterns in data that has no “correct” answer (PALUSZEK; THOMAS, 2016; KIM, 2017).
- Reinforcement learning: is about learning what to do by mapping actions and states aiming the maximization of a reward signal. The

training data is composed by input, some output and reward/penalty (SUTTON; BARTO, 1998; KIM, 2017).

The neural network can be used as a model for machine learning.

2.4 REINFORCEMENT LEARNING

Reinforcement learning (RL) is a machine learning approach to solve problems in which an agent needs to learn the optimal strategy of actions, using trial-and-error search observing rewards provided by a dynamic environment and which state is achieved after each selected action (SUTTON; BARTO, 1998; KAELBLING; LITTMAN; MOORE 1996). One of the inventors of reinforcement learning is Richard Sutton, and since the eighties, it has been continuously improved. Also, its importance in the field of machine learning has increased (SUTTON; BARTO, 1998; SUTTON, 1999; WIERING; OTTERLO, 2012).

In this section, the basic concepts about reinforcement learning that are essential to solve the decision problem proposed in this work are presented. Therefore, the elements of reinforcement learning, the agent-environment interaction, the importance of the reward, the markovian decision process, the value function, the function approximation and the Neural Fitted Q Iteration are presented. Also, an explanation about how to solve model-based and model-free problems is presented.

2.4.1 Elements of reinforcement learning

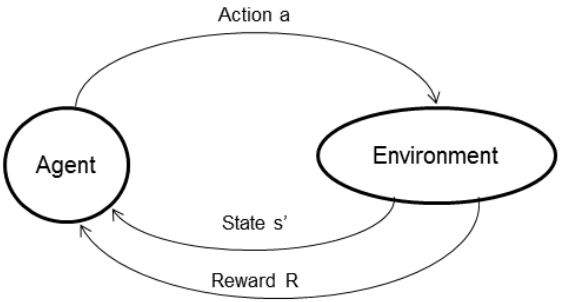
According to Sutton and Barto (1998), the elements of the RL are:

- Policy: is a stochastic rule by which the agent will take an action in each state, the agent behaviour;
- Reward function: indicates the desirability of the states in an immediate sense. The aim of the agent is to maximize the total of rewards in a long-term or minimize the penalties.
- Value function: is the total reward that the agent can expect to achieve in the future starting from some state or taking some action in that state.
- Model: predicts the resultant next state and reward in order to decide among actions in situations not already experienced.

2.4.2 Agent-environment interaction

In the learning process, the one who learns and makes the decisions is the agent. The environment is the thing which the agent interacts with, that presents new situations (states) for the agent and rewards to its actions, this idea is shown in Figure 9.

Figure 9 – Interaction between agent and environment.



Source: Sutton and Barto (1998, p. 52).

2.4.3 Importance of the reward

The only feedback from the environment in the RL problem is the immediate scalar reward signal. The specification of the reward for being in a state or for being in a state and take an action is done by the reward function or also called return. Instead of the reward function, it is also possible to use a cost function that specifies the penalties (negative rewards). These functions indicate implicitly the goal of the RL problem (WIERING; OTTERLO, 2012).

Therefore, by analysing the rewards, it is possible to evaluate and improve the policy (agent behaviour) during learning to achieve an optimal policy (WIERING; OTTERLO, 2012).

2.4.4 Markovian decision process

For the purpose of solving this decision problem in an efficient way, it is necessary to make predictions about states and expected rewards. If the current state carries all information about the past (has the Markov property), it is possible to predict values and make an optimal

decision based only on the knowledge of the current state (SUTTON; BARTO, 1998; WIERING; OTTERLO, 2012).

So, this process is called markovian decision process (MDP) and can be represented by: S – finite set of states; A – finite set of actions; $P_{s,a}$ – state probability matrix (represents the probability of reaching a state after an action); R – reward function; γ – discount factor (it is a parameter that weight the future rewards).

2.4.5 Value function

As stated by Sutton and Barto (1998), some RL algorithms learn to estimate the value functions. The state-value function represents how good is to be in state *s* regarding the amount of future rewards that can be expected starting from the state *s* and following the policy π. On the other hand, the action-value function represents how good is to be in state *s* and take action *a* based on the amount of future rewards that can be expected starting from the state *s*, taking action *a* and following the policy π. For MDPs, the state-value function V^{π} and the action-value function Q^{π} can be calculated using the Bellman Expectation Equations (Equation 7 and Equation 8 respectively):

$$V^{\pi}(s) = E_{\pi}[R_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s] \tag{7}$$

$$Q^{\pi}(s,a) = E_{\pi}[R_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) | s_t = s, a_t = a] \tag{8}$$

where E_{π} means expected value given that the agent follows the policy π. It considers the expected sum of rewards looking ahead in time, but the future rewards are weighted by the discount factor γ that can vary from 0 to 1. This factor is used when the interaction agent-environment continues without limit (SUTTON; BARTO, 1998).

According to Sutton and Barto (1998), the state-value functions can be estimated from experience, in such a way that the agent follows a policy π and maintains an average, for each state encountered, of rewards that have come after the state under analysis. In the case of action-value functions, the average of rewards is related to each action taken in a state.

2.4.6 Solving a model-based problem

As stated by Sutton and Barto (1998) and Wiering and Otterlo (2012), if the environment dynamics can be completely modelled, for

example, all the parameters from the MDP are known, then, it is a model-based RL problem, and it is solved by planning. Dynamic programming (DP) algorithms can be applied in this case, which are algorithms to solve sequential problems that use the value functions to search for good policies.

The DP algorithm divides the problem into two subproblems:

- Policy evaluation (prediction problem): in this step, the state-value function is computed for a policy π in an iterative way. Firstly, the first state-value function for all states (V_0) is arbitrarily chosen, then, in each iteration, the agent sweeps over all states and update V_k or Q_k. If the number of iterations tends to infinity, V_k converges to V^π (SUTTON; BARTO, 1998).
- Policy improvement (control problem): this phase aims to achieve the optimal policy, and it can be solved by two approaches:
  - Policy iteration: the policy π is iteratively improved by acting greedily with respect to the value function, which means that in each state the action with the highest value will be selected (based on the Bellman Optimality Equations) and if this action was not part of the policy π, the policy is updated (SUTTON; BARTO, 1998).
  - Value iteration: it uses the Bellman Optimality Equations as an update rule for the value functions in each iteration (WIERING; OTTERLO, 2012).

In the model-based approach, the value functions can be calculated using Equations 9 and 10, where $\pi(a|s)$ represents the probability of taking action a in state s according to a policy π, R_s^a is the reward associates to a pair state-action, $P_{ss'}^a$ is the probability of achieving state s' by being in state s and taking action a. For $V^\pi(s)$, it is necessary to consider all possible actions that can be taken in s and all possible states that come after these actions are taken. For $Q^\pi(s,a)$, it must be considered all possible states that are achieved after taking the action a.

$$V^\pi(s) = \sum_{a\ e\ A} \pi(a|s)\,(R_s^a + \gamma \sum_{s'e\ S} P_{ss'}^a V^\pi(s')) \tag{9}$$

$$Q^\pi(s,a) = R_s^a + \gamma \sum_{s'e\ S} P_{ss'}^a V^\pi(s') \tag{10}$$

2.4.7 Solving a model-free problem

If the MDP is not known, the RL problem is solved by means of learning, interacting with the environment (SUTTON; BARTO, 1998).

For the solution purpose, many methods are available according to the problem characteristics. When the method is based on the estimate of the value function to select the action, then, it is called value-based. When it is based on learning a parametrized policy to choose the actions, then, it is called policy-based.

According to Chen (2018), value-based methods are used for discrete actions, and policy-based methods are also used for continuous actions. In this work, as only a small number of actions will be considered, it is possible to use a value-based.

Following the value-based approach, once the values for each state or state-action pair is known, it is possible to select one action at each step that leads to the maximum $V^{\pi}(s')$ or $Q^{\pi}(s', a')$, then for finite MDPs, all these actions result in the optimal policy.

The optimal state-value function V^{*} and optimal action-value function Q^{*} which are the expected return for being in state *s* and taking action *a* following the optimal policy are obtained according to the Bellman Optimality Equations (Equations 11 and 12):

$$V^{*}(s) = \max_{\pi} V^{\pi}(s) \tag{11}$$

$$Q^{*}(s, a) = \max_{\pi} Q^{\pi}(s, a) \tag{12}$$

When a cost function is used instead of a reward function, the optimality equations rely on the minimum $V^{\pi}(s)$ and minimum $Q^{\pi}(s, a)$. Therefore, in Equation 11 and 12 the maximum would be replaced by the minimum.

In order to estimate the value functions, it is possible to use methods like Monte Carlo, Temporal differences and Batch reinforcement learning.

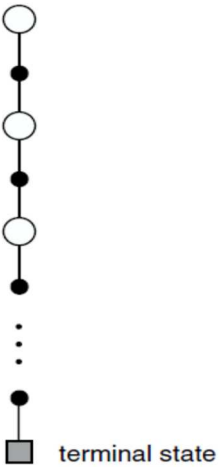
a) Monte Carlo methods (MC):

According to Sutton and Barto (1998) and Wiering and Otterlo (2012), MC methods estimate the value functions by experience. They calculate the empirical mean return of a state *s* by looking straightforward to all the possible successor states of this state from many episode samples

in which a policy π is followed. They wait for the end of the episode to update the value function.

An example of a sample of experience for MC can be seen in Figure 10, in which actions (black circles) are taken in a state (white circles) according to a policy π until the terminal state is achieved. Then, the state-value function for each state is computed at the end of the episode, based on the average between the previous value function and the return seen in the episode.

Figure 10 – Diagram for V^{π} estimation in Monte Carlo.



Source: Source: Sutton and Barto (1998, p. 115).

b) Temporal Differences Learning (TD):

TD methods also learn from episodes of experience, but different from MC, it learns from incomplete episodes, and it can update the value function at each step or at λ steps TD(λ). Moreover, the update of the value function is based on the expectation of the future rewards (value-function of the next state), and it is called bootstrapping (SUTTON; BARTO, 1998; WIERING; OTTERLO, 2012).

Both MC and TD can be controlled by ε-greedy which is a technique that combines steps of greedy selection of actions (exploitation), based on the maximum action-value function, and random selection. It is justified by the fact that states which had a low value for the value function at the first time, would not be selected anymore

according to the greedy selection. However, this state can have some probability of a good return. Then, the random selection is used to increase the capacity of learning once that it does exploration to know other states (SUTTON; BARTO, 1998).

Examples of TD methods are Sarsa (RUMMERY AND NIRANJAN, 1994) and Q-learning (WATKINS, 1989).

c) Batch Reinforcement Learning

According to Wiering and Otterlo (2012), it is a subfield of dynamic programming-based reinforcement learning. Its characteristics are that the agent is not allowed to interact with the system during learning, thus the agent learns from a fixed set of transitions (s,a,R,s') that are sampled from the environment a priori and it does not apply exploration. The idea is to get the best policy out of the available batch of transitions. However, the optimal policy might not be achieved. It is carried out by observing the transitions and updating all the value functions at the same time.

As stated by Wiering and Otterlo (2012), the solution process is divided into 3 phases: exploration of transitions (it could be done in a random way), learning the best policy by applying a Batch RL algorithm and application of the learned policy.

Among the advantages of Batch RL, it is possible to point out the stability and data-efficiency of the learning process that results in fast convergence (WIERING; OTTERLO, 2012).

Nevertheless, important states might not be covered by the sample experiences. Then, aiming to improve the RL solution, the growing batch reinforcement learning process allows increasing the number of samples. It alternates steps of exploration and learning, and in each exploration phase, the set of transitions grows. According to Wiering and Otterlo (2012), this aspect made the growing batch reinforcement learning problem popular. Because it is in between the online update problem, in which the value function is updated each time that the agent sees a single transition, and the pure batch problem in which the update of the value function is done after all the transitions from the sample experience are seen.

Wiering and Otterlo (2012) advise that the batch RL algorithms are more indicated for problems with large state spaces than online learning methods as Q-learning. The online methods present problems of exploration overhead, inefficiency due to stochastic approximation and stability issue when using function approximation.

The batch RL algorithms focus on solving these issues. Examples for these algorithms are Least-Squares Policy Iteration, Fitted Q Iteration and Neural Fitted Q iteration.

According to Sutton and Barto (2018), besides the value-based methods, it is also possible to learn a parameterized policy without the need to use the value function to select an action. They are called policy-based methods. In these methods, the value function can be estimated to help in the update of the policy parameters, but not in the action selection. Methods that use both the value function and the policy approximation are called actor-critic methods.

2.4.8 Function approximation

In cases where the state space is large or complex, continuous state space or images, for example, it would be necessary a lot of memory to store the information, and also, a lot of time to fulfil a table with the value functions for each state-action pair. Therefore, it was proposed the idea of function approximation. It consists in finding a function that approximates the value function so that it would not be necessary to store each value (SUTTON; BARTO, 1998).

According to Sutton and Barto (1998), the approximation function is obtained by means of generalization from states already experienced. This kind of problem is solved using the well-developed supervised learning algorithms like artificial neural network and decision tree.

In supervised learning it is necessary to have the input and the correct output to perform the training, the input can be the state-action pair and output is the action-value function that is estimated from experience, following a policy π. Then, the training is performed. In order to achieve the optimal policy, it alternates between steps of the reinforcement learning to obtain the value functions and steps of supervised learning to improve the function approximation.

2.4.9 Neural Fitted Q Iteration (NFQ)

NFQ is a batch RL algorithm proposed by Riedmiller (2005), it uses a neural network (multi-layer perceptron) to obtain the function approximation for the Q-value function. Riedmiller (2005) considers his work as a realisation of the Fitted Q Iteration proposed by Ernst, Wehenkel and Geurts (2005), which uses a decision tree as function approximation.

The purpose was to solve the problems of long learning time or even failure of learning which is usually due to the local update of neural network multi-layer perceptron that might influence negatively the value in other regions. As the global representation is good regarding generalization, the idea was also to exploit this characteristic (RIEDMILLER, 2005).

The solution was storing all previously experienced transitions in a memory and reusing this data set every time that the Q-function (action-value function) is updated to present explicitly the previous knowledge in each update.

Thus, the basic idea is to update off-line the neural network after an entire set of transitions $D(s,a,s')$ is experienced resulting in faster convergence.

The algorithm for the NFQ is shown in Figure 11. The multi-layer perceptron (MLP) is initialized, then, two main steps are carried out. First, the generation of the pattern set P in which the input is the state and the action taken in each time instance, and the target is the sum of the cost for the transition (s,a,s') and the minimal expected Q-value for s' estimated by the neural network Q_k. Then, the training of the regression pattern within the multi-layer perceptron is performed using, for example, the supervised learning method Rprop (RIEDMILLER, 2005). This procedure is repeated for N epochs until the pattern is learned (RIEDMILLER, 2005).

Figure 11 – NFQ Algorithm.

NFQ_main(){
Input: a set of transition samples D; output: Q-value function Q_N
k=0
Init_MLP() $\rightarrow Q_0$;
Do {
generate_pattern_set $P = \{(input^l, target^l), l = 1, \dots, \#D\}$ where:
$input^l = s^l, a^l,$
$target^l = c(s^l, a^l, s'^l) + \gamma \min_a Q_k(s'^l, a')$
Rprop_training(P) $\rightarrow Q_{k+1}$
k:= k+1
} While (k < N)

Author: Riedmiller (2005, p. 320).

Riedmiller (2005) also explains that some variants can be applied to this basic algorithm. For example, new transitions could be collected and added to the set of transitions *D*. So, the updated set *D* is used to train a new multi-layer perceptron.

2.5 CONTROLLERS PROPOSED IN THE LITERATURE

The reinforcement learning technique has been widely used for controlling the autonomous movement of mobile robots in order to avoid obstacles and reach a goal without any knowledge about the robot kinematic or how the sensors work.

Janusz and Riedmiller (1995) used Q-learning, the offline TD algorithm, combined with a neural network, which acts as function approximation for the Q-value, with the purpose of control two motors of a mobile robot in an unknown environment. The controller is able to output the speed of each motor using as input information of infrared sensors to avoid collisions.

The Q-learning algorithm was applied for Smart and Kaelbling (2002) as well to make a mobile robot complete the task of following a corridor and avoid obstacles. During the learning phase, the robot was controlled by a supplied control and by means of a scanning laser, so that, it was able to detect direction and distance to the goal state and the obstacles. According to the authors, the robot can learn good control policies faster than programmers with moderated experience can write a code that controls the robot movement in the conventional way (without applying machine learning).

Regarding the application of reinforcement learning to control vehicles, Yu and Sethi (1995) proposed a controller that uses reinforcement learning, neural networks and image data to define the steering wheel angle that keeps a vehicle on the road. The image data is processed by means of a neural network to detect the car position. Then, the reinforcement learning algorithm is in charge of taking an action to avoid the car to go off-road, and the action is a steering wheel angle selected using another neural network.

Oh, Lee and Choi (2000) also presented a lateral control architecture based on reinforcement learning to solve the problem of driving a vehicle in high curvature roads at high speeds. The algorithm input is the lateral position error from the lane centre, and its rate of change given by image data and the output is the steering command. The reinforcement learning algorithm used is an online TD, and three neural networks are used for prediction and control.

In 2007, Riedmiller, Montemerlo and Dahlkamp came up with an approach that allows a robot car to learn how to steer in 20 minutes. They used the NFQ reinforcement learning algorithm with the variation that new transitions could be collected by greedily exploiting. The data is provided by real-life experience using sensors that detect position and velocity. Then, when there is a curve in the given trajectory, the algorithm defines an action (steering angle of ±60°, ±10° or 0°) that is performed by means of a motor coupled to the steering wheel. The deviation of the vehicle position from the given track is used as a parameter for the cost function, considering failure a deviation greater than 0.5 m.

Vincent and Sun (2012) proposed a learning controller for an autonomous vehicle that takes over control only in situations that the conventional controller is not able to overcome. They also incorporated a neural network in the solution of the problem, its input is the vehicle dynamics variables given by sensors, and the output is the action to be done.

In 2018, Wang, Chan and de La Fortelle presented a controller based in Q-learning with neural function approximation to perform lane change even when the vehicle faces unforeseen situations.

These works show that robots/vehicles were able to successfully learn a trajectory, performing cornering or which actions to take only by sampled experiences, learning from data.

Reinforcement learning is being used even associated with other controller aiming to self-tuning parameters. For example, Akbari and Goharimanesh (2014) used reinforcement algorithms to optimize the rules of a fuzzy controller for vehicle stability.

He (2005) and Guo et al. (2010) did not present a learning controller to improve vehicle stability. Nevertheless, their idea was also interesting. It was the combination of an active steering controller, a brake-based controller and He (2005) also used a driveline-based controller aiming to overcome the disadvantages of each system. To select the controller that should be active, they used the phase plane of the sideslip angle and its angular velocity, introduced by Inagaki et al. (1994), to check if the vehicle was in a stable or emergency situation.

Another alternative to defining the threshold for the addition of the corrective yaw moment is the comparison between the current yaw rate of the vehicle and the desired yaw rate computed using a linear or nonlinear vehicle model (GOHARIMANESH; AKBARI 2017; LEE; HWANG; SUH, 2015).

3 METHODOLOGY

As previously mentioned, this master's thesis proposes the development of a stability controller based on reinforcement learning for a small electric race car. This controller should be able to observe the current state of the vehicle and define the percentage of torque that should be distributed to each wheel in such a way that an additional yaw moment is generated in cases when the vehicle is in handling limit or unstable situations.

Different from traditional controllers for stability that depends on a mathematical model of the vehicle, the RL controller learnt the torque distribution by interacting with the environment, observing states, taking actions and receiving rewards or penalties.

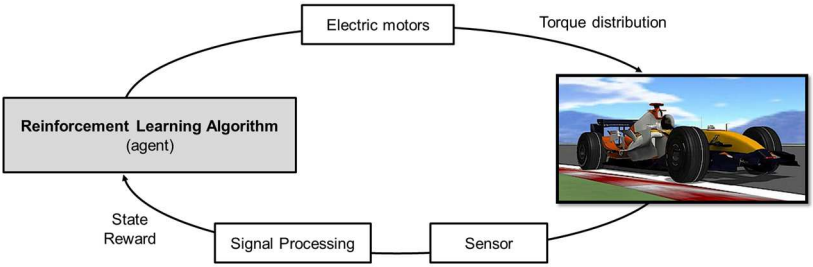
Therefore, in this chapter, the methodology to develop the controller is presented. Firstly, the RL algorithm, the inputs, the actions, the cost function and the policy are defined, in the sequence, it is presented the sampling of experience, the learning process and the final controller architecture.

3.1 REINFORCEMENT LEARNING ALGORITHM

This problem can be seen as a markovian decision problem in which a state is observed, a decision needs to be taken, and the next state depends only on the current state. Therefore, as those decisions should be optimized, an artificial intelligence technique could be used. Looking at the machine learning approaches, in order to apply supervised learning, it would be necessary a correct decision for each state, and without a mathematical model of the vehicle, the correct decision would be difficult to be defined. Unsupervised learning is more used to cluster the data, and it is also more challenging. However, it is possible to establish rewards or penalties for some states and actions and perform simulations with the vehicle to obtain inputs and outputs. Considering these characteristics, the reinforcement learning approach fits better for this problem.

Therefore, the algorithm is the agent, the vehicle represents the environment and the states that describe the vehicle dynamics are detected by sensors, as shown in Figure 12. For this problem, the model of the markovian decision process is not known, then, the Q-value can be estimated using learning. Also, the state space is large, so a function approximation should be used.

Figure 12 – Representation of the reinforcement learning problem.



Source: Author (2019).

As the batch reinforcement learning algorithms are more indicated for large state spaces than MC and TD, due to its better stability (WIERING; OTTERLO, 2012), the selected algorithm to generate the controller model was the batch RL Neural Fitted Q Iteration (NFQ). It was selected also because NFQ was successfully used by Riedmiller, Montemerlo and Dahlkamp (2007) with the goal of making a car learn how to drive. In this work, policy-based methods were not explored due to the time limitation in which the vehicle simulation resource would be available.

So, as explained in 2.4.9, driving simulations should be done to generate the set of transitions $D(s,a,s')$ that contains the states and actions. In the sequence, the pattern set P to train the neural network Q_k should be created. After, the neural network is trained with the regression pattern.

At the end, the neural network was used to help in the selection of the actions in the controller.

3.2 CONTROLLER INPUTS

In order to select the input states, simulations with the race car using the multibody dynamic method (section 3.7.1) were performed in CarMaker. During these simulations, parameters that represent the longitudinal and lateral behaviour of the vehicle and input of the driver were collected: sideslip angle in rad, sideslip angular velocity in rad/s, absolute velocity in m/s, yaw rate in rad/s, steering wheel angle in rad, acceleration in x and in y in m/s^2 (longitudinal and transverse respectively).

With the collected data, it was created the correlation matrix of the states that is shown in Figure 13. It was possible to see that steering wheel

angle, yaw rate and acceleration in y are highly correlated, but correlation among inputs is not recommended when neural networks are used because it might slow the learning (LECUN et al., 2012). Then, the steering wheel angle and the yaw rate were selected as inputs because they are typically used to describe the lateral vehicle dynamics (GILLESPIE, 1992) and these two variables were decorrelated using principal component analysis (PCA).

The PCA is a method that reconstructs the data by using a new orthogonal coordinate system that represents the direction of the higher variance of the data, which is obtained based on the eigenvector of the covariance matrix of the variables (JOLLIFFE, 2002). It results in two new uncorrelated variables PC1 and PC2 that are a linear combination of the yaw rate and steering angle. It is represented in Equation 13 where $\boldsymbol{z}$ is the vector of principal components PC1 and PC2, $\boldsymbol{U}$ is the matrix of eigenvectors and $\boldsymbol{y}$ is the data (yaw rate and steering wheel angle).

$$\boldsymbol{z} = \boldsymbol{U}\boldsymbol{y} \tag{13}$$

The acceleration in x (in longitudinal direction) also was selected because this variable is easily measured. On the other hand, sensors to measure the sideslip angle and sideslip angular velocity usually are expensive (DU; ZHANG, 2011), so these variables were not considered as input. Therefore, the application of the algorithm is simpler.

Regarding the velocity, two approaches could be considered during the research phase regarding the development of this controller, to use the absolute velocity or to use its longitudinal and lateral components. Thus, to investigate the influence of these two approaches in the learning process, two experiments were performed. Experiment A using the absolute velocity and experiment B using the longitudinal and lateral velocity. Table 1 presents the states for each experiment.

Table 1 – Selected states for each experiment.

States - Experiment A	**States - Experiment B**
• Longitudinal acceleration • Steering wheel angle • Yaw rate • Absolute velocity	• Longitudinal acceleration • Steering wheel angle • Yaw rate • Longitudinal velocity • Lateral velocity

Source: Author (2019).

Figure 13 – Correlation matrix of the states.

	Sideslip angle	Sideslip angular velocity	Absolute velocity	Yaw rate	Steering wheel angle	Acceleration in x	Acceleration in y
Sideslip angle	1,00	0,00	0,20	0,50	0,66	-0,60	0,41
Sideslip angular velocity	0,00	1,00	0,00	-0,19	-0,14	-0,01	-0,16
Absolute velocity	0,20	0,00	1,00	-0,07	-0,02	-0,71	-0,10
Yaw rate	0,50	-0,19	-0,07	1,00	**0,98**	-0,46	**0,99**
Steering wheel angle	0,66	-0,14	-0,02	**0,98**	1,00	-0,53	**0,96**
Acceleration in x	-0,60	-0,01	-0,71	-0,46	-0,53	1,00	-0,41
Acceleration in y	0,41	-0,16	-0,10	**0,99**	**0,96**	-0,41	1,00

Source: Author (2019).

It is important to mention that for real application, the degradation on the state signal due to the accuracy and limitations of the measurement system should be studied. For example, to measure the velocity, a GNSS and an inertia platform system should be available. However, the accuracy of these systems might influence in the performance of the controller.

To make the learning process of the neural network faster, it is recommended to standardize the input data (LECUN et al., 2012). Thus, each input variable x^l was standardized according to Equation 14, where $x_{min}{}^l$ and $x_{max}{}^l$ are the minimum and the maximum value of each variable respectively.

$$x_s^l = (x^l - x_{min}{}^l)/(x_{max}{}^l - x_{min}{}^l) \tag{14}$$

After that, the values were rounded to one decimal place, except for the absolute and longitudinal velocity that were rounded to two decimal places.

3.3 POSSIBLE ACTIONS

Regarding the actions, it is not possible to add a steering wheel angle or to actuate in the brakes of the studied vehicle to apply a solution like the AFS or ESC system. Nevertheless, it is possible to apply torque vectoring.

The aim of changing the torque distribution between the wheels is to create the additional yaw moment in the vehicle to bring it back to a stable situation.

Therefore, for both experiments, it was defined that the actions would be the percentage of torque that goes to the left rear wheel, and also discrete actions. The possible percentages for each experiment are presented in Table 2.

Table 2 – Possible actions for the experiments.

	Experiment A	**Experiment B**
Possible percentages of torque sent to left wheel	30%, 40%, 50%, 60%, 70%	10%, 30%, 50%, 70%, 90%

Source: Author (2019).

Different actions were set to the Experiments A and B to investigate the maximum percentage of torque distribution that is necessary to keep the vehicle stable.

3.4 COST FUNCTION

The cost function was based on the analysis of the phase plane β-β' diagram used also by Inagaki et al. (1994), He (2005) and Guo et al. (2010) in their controllers based on theoretical models to define the limits of the vehicle stability.

The phase-plane analysis is a graphical method used to perform the stability analysis of nonlinear dynamic systems (HE, 2005; ZHANG et al., 2011). These systems can be represented by first-order nonlinear differential equations as presented in Equation 15, where $\boldsymbol{s}$ is the state vector, and $\boldsymbol{f}$ is the nonlinear vector function of the states (HE, 2005).

$$\dot{\boldsymbol{s}} = \boldsymbol{f}(\boldsymbol{s}, t) \tag{15}$$

The phase plane is obtained by plotting one state as a function of the other, and at the end, it represents the transient response of the system to initial conditions or constant inputs (HE, 2005). In the resulting graph,

it is possible to see the states trajectory over time and the influence of equilibrium points (points at which the states of the system do not change over time). By looking at the trajectories and the equilibrium points, it is performed the stability analysis.

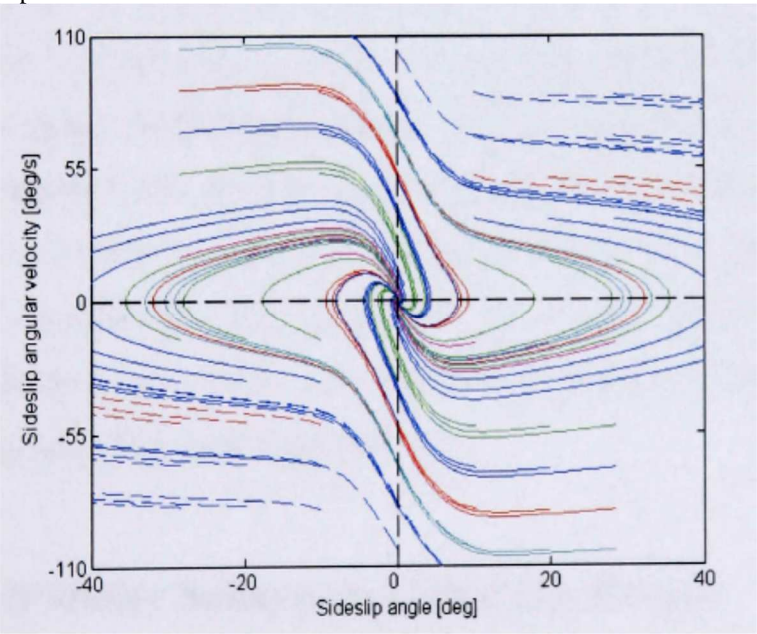
In the case of the vehicle, its lateral vehicle dynamics can be represented by a second order nonlinear system (GILLESPIE, 1992) that could be represented by means of Equation 12 considering two states $s_1 = s$ and $s_2 = \dot{s}$. Nonlinear systems can have more than one equilibrium point, the vehicle system has three equilibrium points, but only one is a stable focus (INAGAKI et al., 1994). Therefore, it is possible to use the phase plane diagram to perform the stability analysis of vehicles.

Regarding the selection of the states, phase plane based on vehicle sideslip angle and sideslip angular velocity or yaw rate and sideslip angle can be created (ZHANG et al., 2011), even though in the last one the states are not one the derivative of the other. For this work, the vehicle sideslip angle and its angular velocity were selected because according to He (2005), it is more effective.

To create the phase-plane diagram of the vehicle, it is important to know the vehicle model and the tire model. However, as this diagram was used only to evaluated how good or how bad the actions of the RL algorithm were, in this work the β-β' phase-plane for the studied vehicle was not created. Instead, the phase-plane from (HE, 2005) was used, once that he developed an extensive study on this diagram.

He (2005) created a $\beta - \dot{\beta}$ phase-plane to control vehicle stability and handling using a nonlinear two degrees of freedom model for the vehicle and the Pacejka Tyre Model (PACEJKA; BESSELINK, 1997) to represent the tires. He performed simulations considering the speed as 100 km/h, the steering angle as zero and the friction coefficient as 1 (nominal road surface). As result, he obtained the graph shown in Figure 14, where the solid lines represent stable trajectories, dashed lines represent unstable trajectories and the point (0,0) is an equilibrium point.

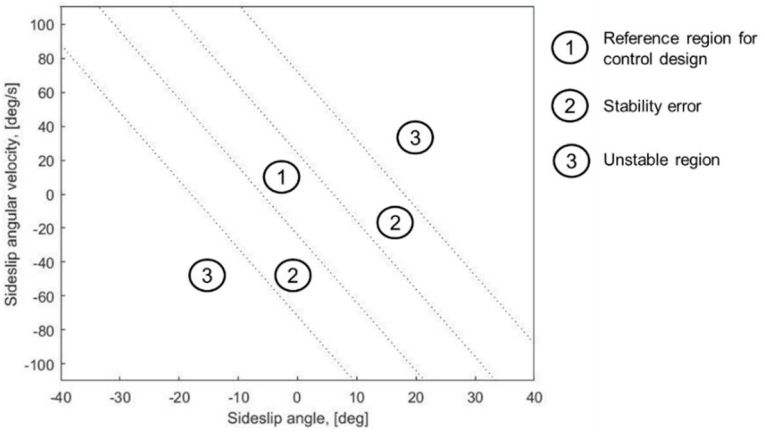
Figure 14 – β-$\dot{\beta}$ Phase-plane diagram obtained by He (2005) with zero steer input.



Source: He (2005, p. 137).

By the analysis of the diagram, He (2005) defined an approximated reference region for control design (Figure 15), being that outside of this region a correction to the behaviour of the car should be made. He also defined a region of "stability error" to be more conservative as the phase-plane changes with the steering input.

Figure 15 – Definition of reference region and stability error for vehicle control.



Source: He (2005, p. 143).

The Equation 16 represents the reference region, and the Equation 17 defines the "stability error", where $k_{\beta\dot{\beta}}$ is the slope of the curve (equal to 4), d and d' are the half width of the region, and their values are 72 and 24 respectively. β is in degrees and $\dot{\beta}$ in degrees/s.

$$\left|\dot{\beta} + k_{\beta\dot{\beta}}\beta\right| < d' \tag{16}$$

$$\left|\dot{\beta} + k_{\beta\dot{\beta}}\beta\right| < d \tag{17}$$

Based on these regions, He (2005) defined how the controller should act. If the vehicle was in the "stability error" region, handling control should be applied. If the vehicle was outside of the reference and "stability error" regions, dynamic stability control should be applied.

In the present work, this idea was used to create the vehicle cost function. It was considered that when the vehicle was in state s, took action a and achieved a state s' that is in region 3 (emergency situation), it is a failure. In this case, the value of the cost function is 1. If the state s' is in the "stability error" region, the value is 0.4. When the state s' is in the reference region (stable situation) and the taken action was 50% of torque distribution, the value of the cost function for the state-action pair is zero. However, when the vehicle is in a stable situation and the torque distributions was different than 50% the cost value was 0.10. This consideration was done because the vehicle should learn not to use torque

distribution different than 50% in a stable situation as it can make the handling unnatural for the driver (HE, 2005). The cost function can be seen in Equation 18.

$$c(s,a,s') = \begin{cases} 1.00, & if\ \left|\dot{\beta} + k_{\beta\dot{\beta}}\beta\right| \geq 72 \\ 0.40, & if\ 24 \leq \left|\dot{\beta} + k_{\beta\dot{\beta}}\beta\right| < 72 \\ 0.10, & if\ \left|\dot{\beta} + k_{\beta\dot{\beta}}\beta\right| < 24\ \&\ action \neq 50\% \\ 0.00, & if\ \left|\dot{\beta} + k_{\beta\dot{\beta}}\beta\right| < 24\ \&\ action = 50\% \end{cases} \quad (18)$$

Therefore, during the learning process, it was necessary the acquisition of the sideslip angle and the sideslip angular velocity.

3.5 Q-VALUE FUNCTION

As the NFQ algorithm was selected, a regression neural network was used to estimate the Q-value, which represents how good it is to be in one state and take some action.

The neural network architecture used was composed of one input layer with five nodes (in Experiment A) and six nodes (in Experiment B), two hidden layers with ten nodes and an output layer with one node. For all hidden nodes the activation function was the logistic sigmoidal (0,1) as used by Riedmiller, Montemerlo and Dahlkamp (2007) and for the output layer, the purelin function was used (linear function). These functions were used because they presented better results in the representation of the Q-value function. Weights and bias were initialized automatically randomly by the software Matlab.

Two hidden layers were used since they are normally better for generalization then using only one hidden layer (THOMAS et al., 2017).

For supervised training of the neural network the scaled conjugate gradient algorithm was used, as this algorithm aims fast learning, it does not depend on parameters defined by the user, and its performance does not degrade as quickly as Rprop when the error is reduced (MATHWORKS, 2018). The maximum number of epochs used for training was 500.

The input to train the neural network was the state observed (defined in 3.2) and action taken in that state. The target was the estimated Q-value for the pair state-action. It was calculated according to Equation 19, where the cost of the transition is added to the minimum Q-value

related to the next state (calculated using the previous neural network) multiplied by the discount factor γ considered as 0.95 (because the future costs are important).

$$Q(s,a) = c(s,a,s') + \gamma \min_{a'} Q(s',a') \tag{19}$$

The training data set was obtained by running simulations of the vehicle behaviour and watching the states, actions and costs. Then, each time a new neural network was trained, the estimated Q-value was improved.

To avoid overfitting, 15% of the collected data was used as a validation data set. During the training process, the mean squared error related to the validation set was monitored, and after the initial phase, if this error increased for a defined number of iterations (meaning that the neural network is overfitting), the training is stopped.

3.6 POLICY

As the data collection for training was performed by means of simulations, it was possible to use the policy ε-greedy with 10% of random choices to explore the state space and 90% of greedy choices (selecting the action with lowest Q value in the state) to improve the policy.

In a real environment, random choices are avoided because they may result in hazardous situations.

3.7 SAMPLING OF EXPERIENCES

The sampling of experiences was performed by means of simulation of the dynamic behaviour of the vehicle in the software CarMaker 6.0.1.

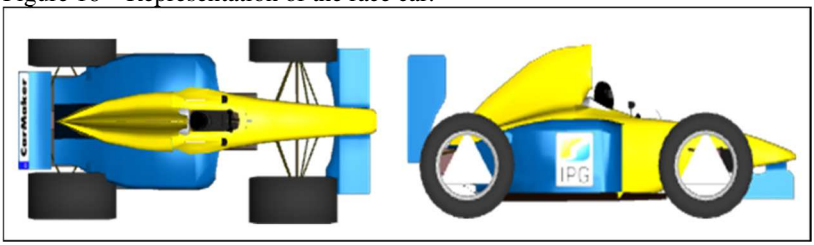
The vehicle studied was a small rear wheel driven (RWD) electric race car similar to a car from Formula Student competition. In Table 3, it is shown the vehicle properties, and in Figure 16 the studied vehicle is illustrated.

Table 3 – Properties of the studied vehicle.

Property	Value
Mass [kg]	191
Tire diameter [m]	0.33
Number of electric motors	1
Maximum torque of the motor [N.m]	250
Maximum rotation of the motor [rpm]	4500
Gear ratio	1.13
Wheelbase [m]	1.60
Track width [m]	1.20

Source: Author (2019).

Figure 16 – Representation of the race car.



Source: IPG (2018).

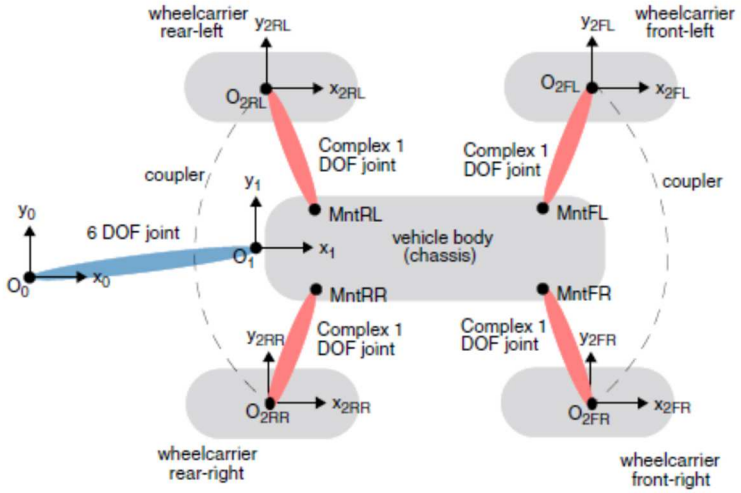
The software CarMaker carries out simulations based on the multibody dynamic method.

3.7.1 Multibody dynamic simulation

According to Font-Llagunes (2016), the multibody dynamic method relies on the representation of the mechanical system by solid bodies or links connected by kinematic constraints or force elements, such as joints and springs. The motion of the system is described by the trajectory of specific points which are solved normally by means of differential-algebraic equations (DAEs) (FONT-LLAGUNES, 2016).

The software CarMaker from IPG makes use of these equations to perform simulations of the vehicle dynamics. The vehicle is represented by five rigid bodies connected by five joints. As illustrated in Figure 17, the chassis body is connected to the wheel carrier bodies with a complex joint of one degree of freedom, and it is also connected to the ground body with a six degrees of freedom joint. It is possible to add kinematical dependence between the rear wheel carriers and the front wheel carriers to express a rigid axle for example (IPG, 2014).

Figure 17 – Vehicle representation by means of multibody dynamics.



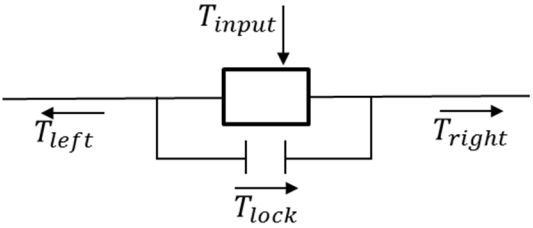
Source: IPG (2014, p. 69).

The CarMaker has an interface with Matlab and Simulink. Therefore, it was possible to add a controller for torque vectoring in the vehicle model in software-in-the-loop configuration. Thus, the controller could read the states (controller inputs), calculate the Q-values using one neural network block for each state-action pair and take actions according to the policy ε-greedy.

To guarantee that different values of torque would be distributed to the wheels, the driveshafts were coupled with the model "Torque Vectoring" available in CarMaker (Figure 18). This lockable model of differential allows the transference of torque between the left and right driveshafts in two directions according to the defined torque ratio. Equation 20 shows the torque T_{Lock} that should be transferred from one side to another, where $TrqRatio$ is the desired percentage of torque that should be distributed to the left wheel and T_{input} is the input torque in the differential.

$$T_{Lock} = T_{input}(1 - 2 \cdot TrqRatio) \tag{20}$$

Figure 18 – Representation of “Torque Vectoring” coupling model for the driveshafts.



Source: IPG (2018).

Aiming to simulate situations in which the vehicle could learn how to behave in order to stay stable regarding its lateral dynamics, the manoeuvre Sine with Dwell was used according to (UN ECE, 2014).

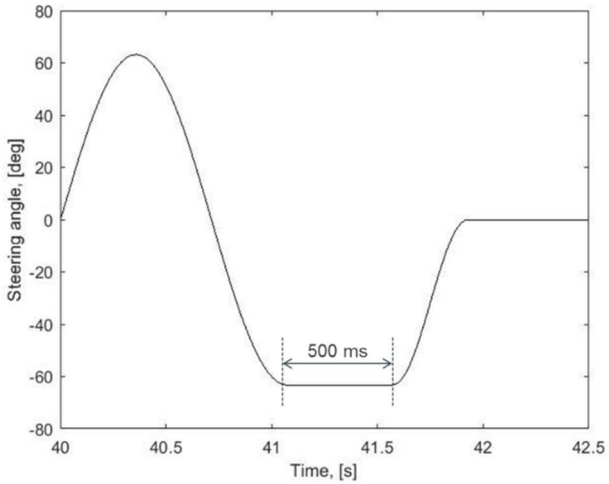
3.7.2 Manoeuvre Sine with Dwell

Sine with Dwell is an open-loop manoeuvre used in the mandatory homologation tests from UN ECE R13H (UN ECE, 2014) for the type-approval of the electronic stability control in passenger cars in Europe. The advantage of it being open loop is that the driver does not apply corrections in the vehicle motion. Then, it guarantees repeatability.

According to UN ECE R13H, in this manoeuvre, it is given a steering wheel input in the shape of a sine wave with a frequency of 0.7 Hz and a dwell of 500 ms as represented in Figure 19.

For the ESC type approval, tests with different amplitudes of the sine wave should be performed. To set the amplitudes, firstly, it is necessary to find the steering angle A that causes in the vehicle a lateral acceleration of 0.3g at 80 km/h. Then, the steering wheel inputs will be a sine wave with amplitudes that vary from 1.5A to 6.5A in steps of 0.5A (clockwise and anticlockwise). After that, it is verified if the vehicle complies with the criteria defined by the standard (Annex A).

Figure 19 – Sine Steer input for Sine with Dwell manoeuvre.



Source: UN ECE (2014, p.81).

In this work, due to a limitation of computational time, the simulations for the sampling of experiences for both Experiment A and B were performed only for four amplitudes:

- 2.5A (±28.74°): amplitude that makes the studied vehicle remain in the reference region of the phase-plane of sideslip angle and its angular velocity;
- 5.5A (±63.23°): smallest amplitude that does not comply with the criteria of UN ECE R13H;
- 6.5A (±74.73°): amplitude related to the highest factor that should be used in the simulation;
- 8A (±91.97°): according to the phase-plane of the vehicle sideslip of the studied vehicle, this is the smallest angle that leads to loss of stability.

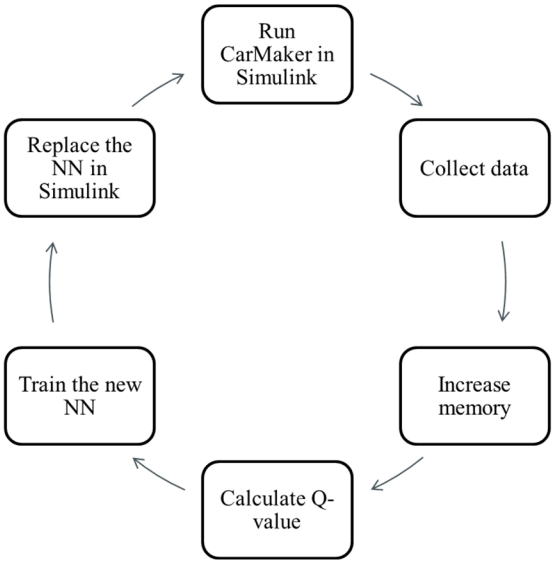
As the vehicle used in the simulations was smaller than a passenger car, the amplitudes for the sine input were also smaller than what would be expected for a passenger car. However, this standard was still used because of its complete procedure and adaptation of the amplitudes regarding the characteristics of the vehicle.

3.8 LEARNING PROCESS

For the first sampling of experiences, it was necessary to add to the controller model in Simulink a neural network trained with random values in a range of [0, 1.5], because the model needs a neural network for the selection of actions during the simulation. However, at this point, there was no collected data from the vehicle behaviour for training. Therefore, the first neural network is trained with random values and does not represent the estimated Q-value. This neural network is also used to calculate the minimum Q-value of the state s' in Equation 19 in order to obtain the target to train the next neural network. For this reason, the learning process must be done by the alternation of steps of sampling of experience with steps of updating of the Q-value function aiming to improve the neural Q-value in each cycle.

After each sampling performed in CarMaker, the sampled data set obtained was added to a memory, and one iteration of NFQ was carried out in Matlab to generate the pattern data set P and to train a new neural network. The new neural network was replaced in Simulink, and another step of sampling of experience was performed. This cycle is shown in Figure 20.

Figure 20 - Representation of the learning process.



Source: Author (2019).

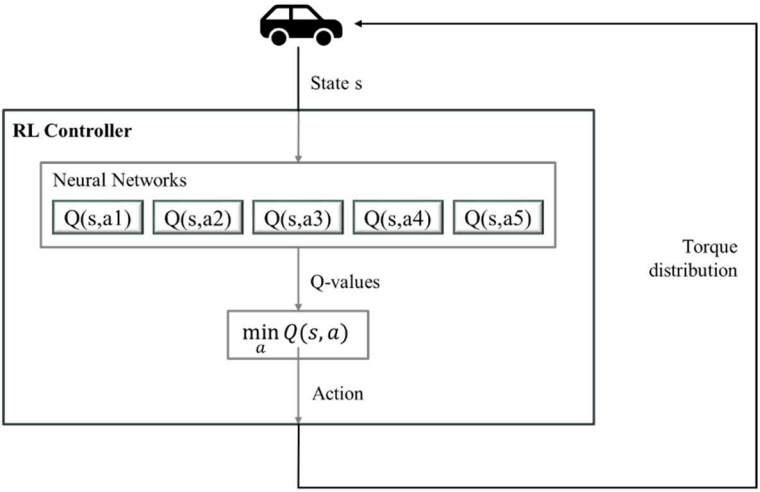
This process was repeated ten times for each sine amplitudes (±28,74°, ±63.23°, ±74.73°, ±91.97°) in a random way to guarantee better learning.

The phase that took most time in this process was the generation of the pattern set P to train the neural network because the number of data increased after the end of each simulation. At the end of the learning process, this phase was taking more than 4 hours.

3.9 FINAL CONTROLLER ARCHITECTURE

After the learning process, the model of the controller was generated, which is able to control vehicle handling and vehicle stability by receiving information about the state. It uses the neural networks to calculate the Q-values for all state-action pairs in that state and then, it selects the action with the lowest Q-value. In other words, the percentage of torque that should be distributed to each wheel to keep the car stable. This process is represented in Figure 21.

Figure 21 - Representation of the final controller.

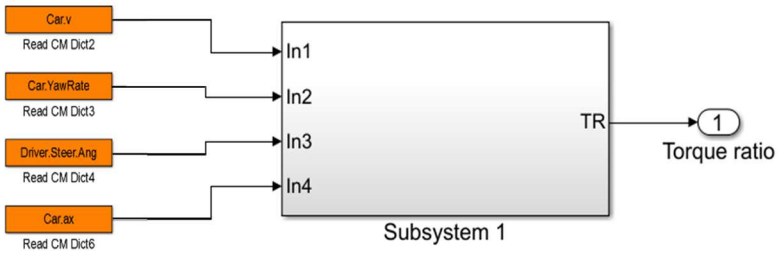


Source: Author (2019).

In Simulink, the stability controller works in software-in-the-loop configuration using blocks to read the states (orange blocks in Figure 22), and they are the input for the Subsystem 1. In this subsystem, represented in Figure 23, the states are normalized and decorrelated using a MATLAB

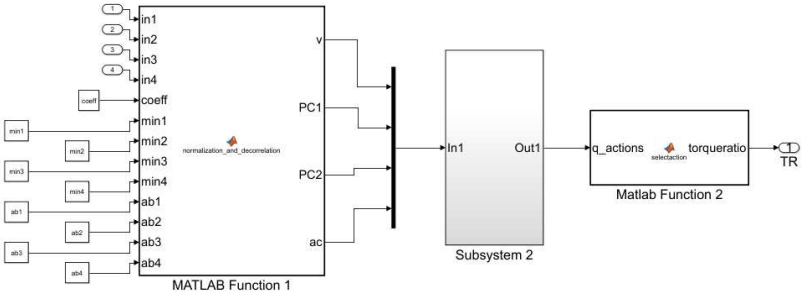
function. For this, it must be given the minimum and the range of each parameter for normalization, and the eigenvector matrix related to the yaw rate and the steering wheel angle for decorrelation.

Figure 22 – Controller for Experiment A.



Source: Author (2019).

Figure 23 – Subsystem 1 from Experiment A.



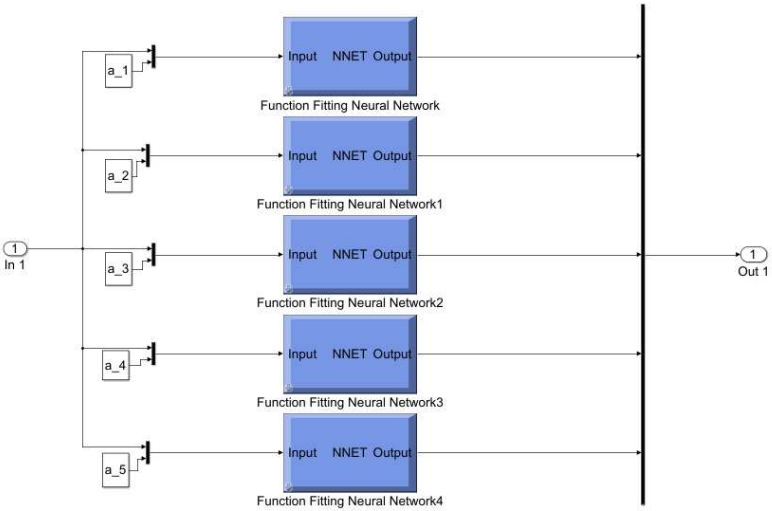
Source: Author (2019).

In the sequence, the output of the MATLAB Function 1 is given as input to the Subsystem 2 that contains the five identical neural networks blocks associated to each possible action to calculate the estimated Q-values, it can be seen in Figure 24.

Then, the Q-values become the input for the MATLAB Function 2 where the action related to the minimum Q-value is identified. This action is the torque ratio for the torque vectoring.

The architecture of the controller from Experiment B is similar to the controller A. The difference is that there are five states instead of four.

Figure 24 – Subsystem 2 from Experiment A.



Source: Author (2019).

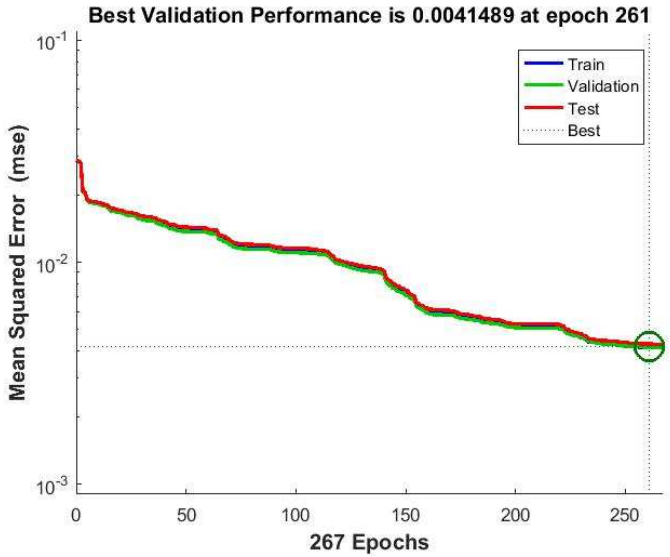
4 ANALYSIS OF THE RESULTS

At the end of the learning process, the neural networks that estimate the Q-value function for the controllers from Experiment A and B were obtained. So, in this section, the performance of the neural networks is presented as well as the evaluation of the controllers.

4.1 PERFORMANCE OF THE NEURAL NETWORKS

During training, in which the weights and bias were calculated, the performance of the neural networks was monitored to avoid overfitting. In Figure 25, it is possible to see that for Experiment A the training stopped at epoch 261 with a minimum square error of 4.15E-03. The R^2 for this neural network is 0.88 regarding the regression for the validation data set.

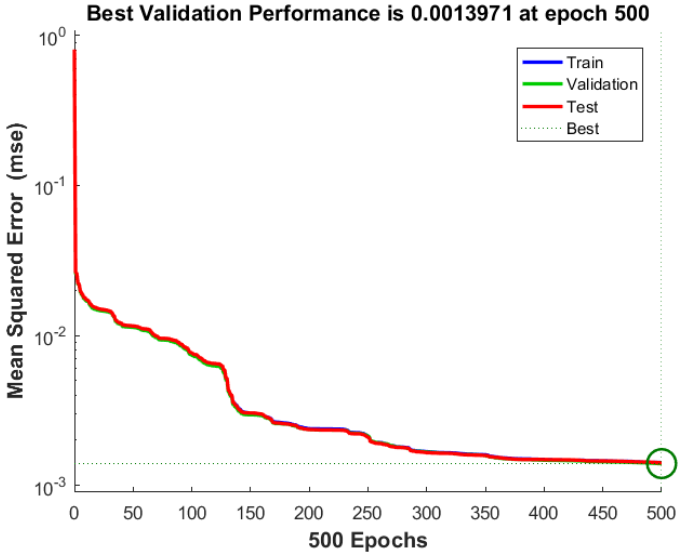
Figure 25 – Performance of the neural network during training – Experiment A.



Source: Author (2019).

In Figure 26, it is possible to see that for Experiment B the training stopped at epoch 500 with a minimum square error of 1.4E-03; The R^2 for this neural network is 0.97 regarding the regression for the validation data set for the Experiment B.

Figure 26 – Performance of the neural network during training – Experiment B.



Source: Author (2019).

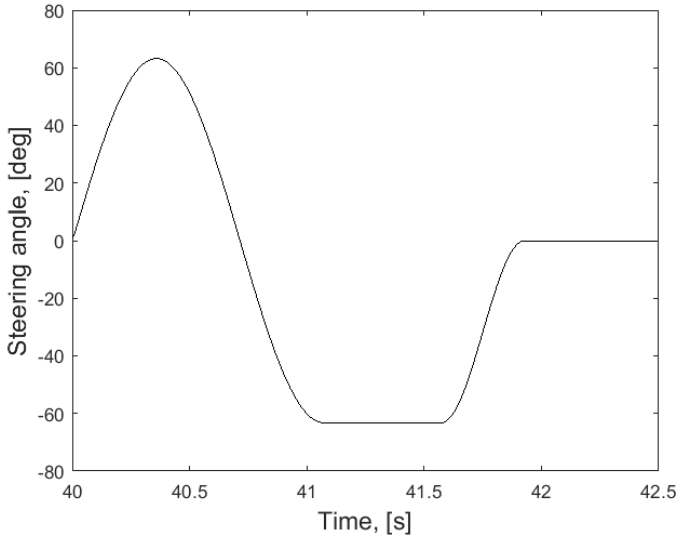
As the overfitting was avoided, the neural networks can generalize when a different input is given.

4.2 EVALUATION OF THE RL CONTROLLERS

The performance of the controllers from Experiment A and B were evaluated by means of comparison between the result of simulations of the vehicle behaviour with and without controller.

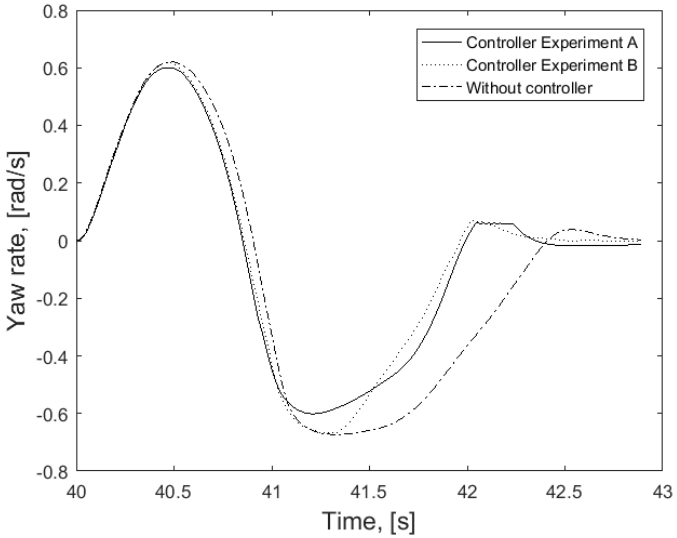
The simulations were carried out in CarMaker and the manoeuvre performed was Sine with Dwell with amplitude 63.23° and 91.97° at 80 km/h. From Figure 27 to Figure 34, the results of the simulations are presented: the steering input, the yaw rate, the phase-plane of the vehicle sideslip and the sideslip angle for 63.23° and 91.97° respectively.

Figure 27 – Steering input for the simulation of 63.23°.



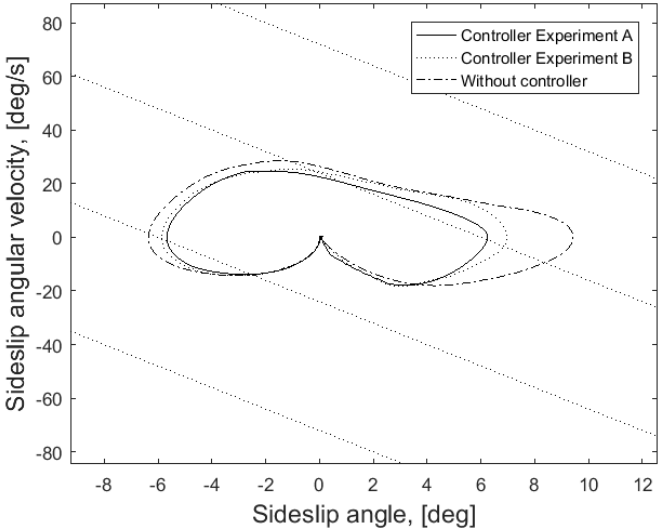
Source: Author (2019).

Figure 28 – Yaw rate for the simulation of 63.23°.



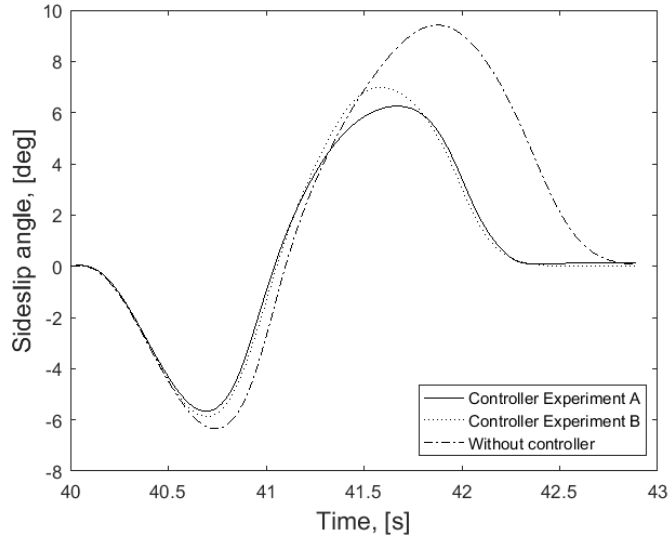
Source: Author (2019).

Figure 29 – Phase plane of the vehicle sideslip for the simulation of 63.23°.



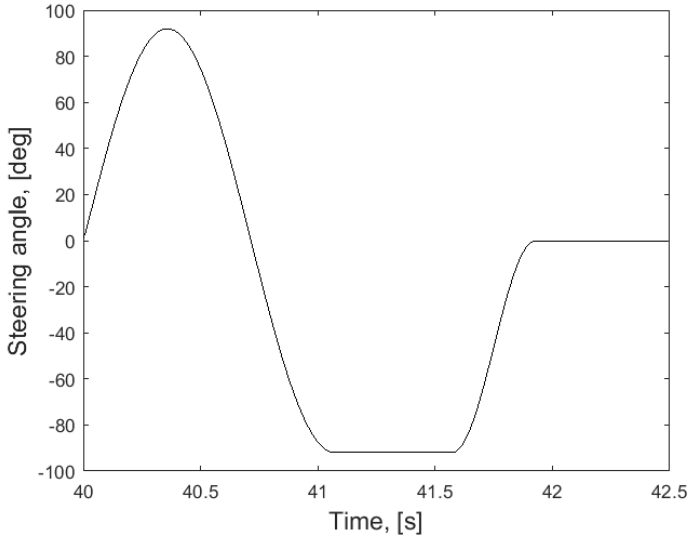
Source: Author (2019).

Figure 30 – Sideslip angle for the simulation of 63.23°.



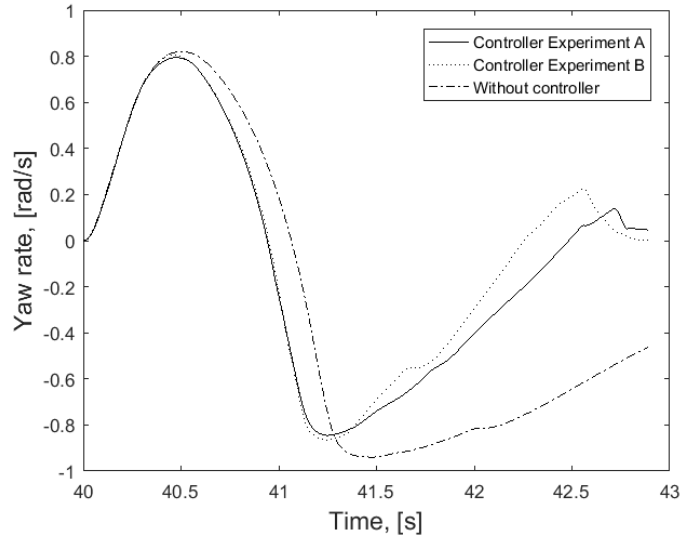
Source: Author (2019).

Figure 31 – Steering input for the simulation of 91.97°.



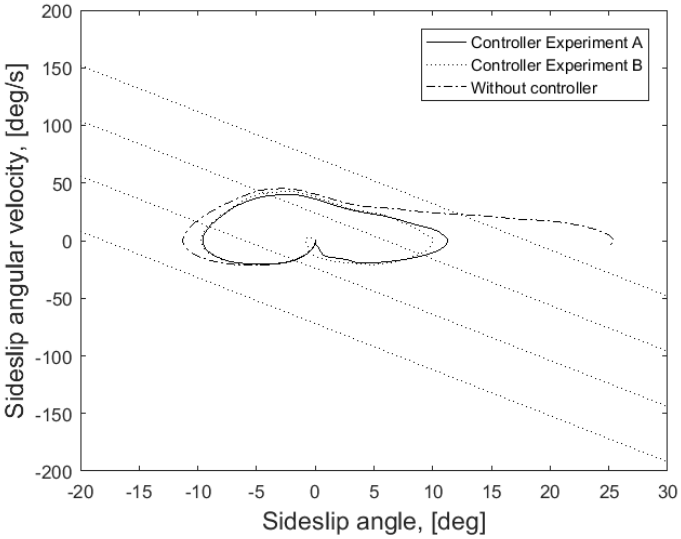
Source: Author (2019).

Figure 32 – Yaw rate for the simulation of 91.97°.



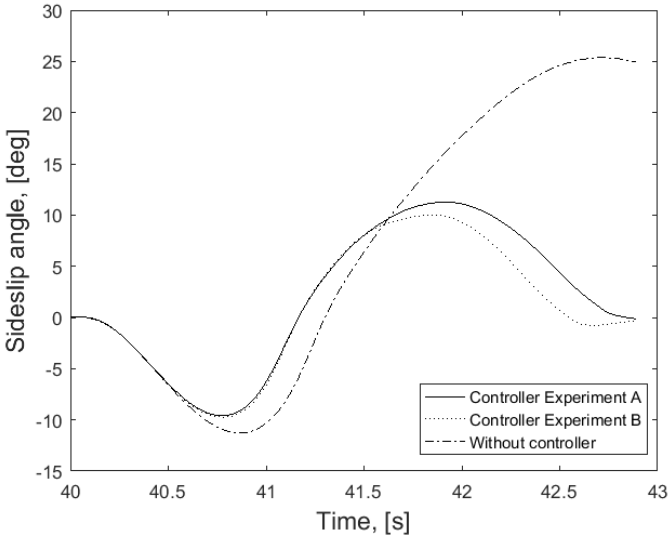
Source: Author (2019).

Figure 33 – Phase plane of the vehicle sideslip for the simulation of 91.97°.



Source: Author (2019).

Figure 34 – Sideslip angle for the simulation of 91.97°.



Source: Author (2019).

In Figure 29, it is possible to see that without the controller, the vehicle would not leave the regions one and two of the phase-plane. However, with the controllers, the peak of the vehicle sideslip angle was reduced in 33,62% using the controller from Experiment A and in 25.85% using the controller from Experiment B, as shown in Figure 30. Therefore, vehicle handling was improved.

Figure 33 shows that without a controller, the vehicle loses stability as the β-β' phase-plane trajectory reaches region 3. Using the controllers, the vehicle remains within the limits of handling, which means stability enhancement.

The enhancement can also be seen in Figure 28 and Figure 32 once that with the Controller A or with the Controller B, the yaw rate stabilizes faster after the end of the steering wheel input that takes place in 41.9 s. This fact validates the use of the model proposed by He (2005) for the phase plane of vehicle sideslip angle and sideslip angular velocity as a cost function.

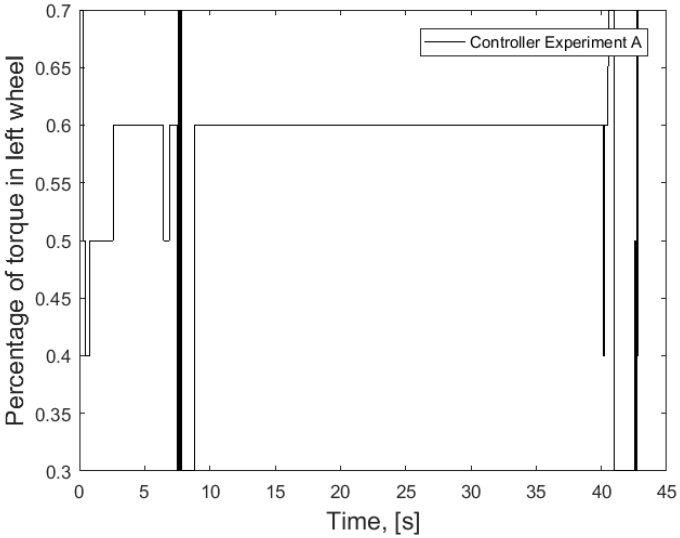
It was also necessary to check if the controllers were able to learn that when the vehicle was in region 1 of the phase-plane diagram, for example, before the steering input that starts at 40 s, it should select as action 50% of torque for each wheel. In Figure 35, that demonstrate the torque distribution over time for the sine wave with the amplitude of 91.97° for Controller A, it is possible to see that 0.1 as transition cost was not enough to make the controllers learn to select 50% of torque distribution before 40 s.

From Figure 35, a problem with pattern recognition can also be detected, because at the beginning of the vehicle motion, between 5 s and 10 s where the steering wheel input is zero, the controller applies torque distribution using the maximum and the minimum values of the range in a stable situation, and it should not happen. This behaviour was detected in Experiment B too; however, more simulations and a modified cost function could solve this problem.

In general, the results demonstrate the success of this new approach to create a stability controller without a mathematical model of the vehicle and having the contribution of the sideslip angle, even though it is not used as an input.

It is also important to point out the lack of results in the literature for direct comparison.

Figure 35 - Torque distribution – Amplitude of 91.97° - Controller A.



Source: Author (2019).

5 CONCLUSION

This master's thesis presents a new application of reinforcement learning in active safety with the development of a controller based on reinforcement learning to improve vehicle handling and vehicle stability by controlling the torque vectoring of a small electric race car. To accomplish that, an algorithm based on reinforcement learning was implemented, training data was generated using simulations of the vehicle behaviour, and the learning process was carried out to create the controller model.

To investigate the maximum ratio of torque distribution that should be set to guarantee stability as well as the effectiveness of the controller inputs in the learning process, two experiments were done (A and B).

From the results of Experiments A and B, it is possible to see that both controllers could enhance the vehicle dynamics. However, the controller from Experiment A presented better results regarding the reduction of the sideslip angle. This may mean that in this case, it is better to use the absolute velocity as input instead of its components. Regarding the maximum percentage of torque distribution, the performance of the Controller A showed that 70% is enough to guarantee the vehicle stability.

The results of the learning process demonstrate that the selected cost function was satisfactory for learning to cope with the vehicle behaviour. The validation of the phase-plane of the vehicle sideslip as cost function can be done by looking at the reduction of the sideslip motion, but mainly by looking at the stabilization of the yaw rate after the end of the steering wheel input. It is even possible to say that the threshold for the addition of the yaw moment defined by the cost function is effective, once that the controllers were able to help the vehicle in handling limit and emergency situations. Although, the cost function should be improved for stable regions to avoid torque ratio different from 50% in stable situations.

The results also show that even though not all the states that describe the lateral vehicle behaviour were used as input of the neural network, it still could find a pattern that represents this behaviour. The architecture of the neural network with two hidden layers and the selected activation functions were also important to identify the nonlinearity. Moreover, using the selected states as input, it is possible to say that it is not necessary to use the sideslip angle as input to capture the nonlinear behaviour to control the stability. Therefore, a simpler controller was created.

The torque vectoring controller was able to keep the vehicle under a stable situation with only 70% as maximum torque distribution. And considering that it has the advantage of not influencing the longitudinal dynamics, it is noticeable that this controller is suitable for this application.

As expected, the Batch RL method NFQ showed to be data-efficient, given the fact that only a few simulations were performed.

5.1 SUGGESTIONS FOR FUTURE WORK

The suggestions for future work are:

- Perform more simulations for sampling of experiences with other steering wheel inputs, velocities, roads properties and manoeuvres to improve the controller performance.
- Study the use of other algorithms of reinforcement learning that have been developed recently.
- Study the influence of the properties of the measurement systems.
- Study the driveability experienced by the driver.
- Perform a sensitivity analysis regarding the change of controller inputs.
- Implement a standard linear controller, such as PID, for comparison.

REFERENCES

AKBARI, A. A.; GOHARIMANESH, M. Yaw Moment Control Using Fuzzy Reinforcement Learning. In: INTERNATIONAL SYMPOSIUM ON ADVANCED VEHICLE CONTROL, 12., 2014. **Proceedings**… AVEC, 2014.

CARRO no Brasil ganhará novos itens de segurança obrigatórios. **Gazeta do Povo**. 2017. Available in: <http://www.gazetadopovo.com.br/automoveis/carro-no-brasil-ganhara-novos-itens-de-seguranca-obrigatorios-veja-quais-ewttudwagbw4f5vg0c7t6649g>. Access in: 01 Feb. 2018.

CHEN, S. **Comparing Deep Reinforcement Learning Methods for Engineering Applications**. 2018. 140 p. Thesis (Master) - Faculty of Computer Science, Otto-von-Guericke-Universität Magdeburg, 2018.

CLEMSON VEHICULAR ELECTRONICS LABORATORY (CVEL). **Electronic Stability Control.** Available in: <https://cecas.clemson.edu/cvel/auto/systems/stability_control.html>. Access in: 04 Feb. 2018.

COLUMBUS, L. **How Artificial Intelligence Is Revolutionizing Business In 2017**. 2017. Available in: < https://www.forbes.com/sites/louiscolumbus/2017/09/10/how-artificial-intelligence-is-revolutionizing-business-in-2017/#4a386df55463>. Access in: 08 Mar. 2018.

DESJARDINS, C.; CHAIB-DRAA, B. Cooperative adaptive cruise control: A reinforcement learning approach. **IEEE Transactions on Intelligent Transportation Systems**. v. 12, n. 4. IEEE, 2011. p. 1248-1260.

DU, H.; ZHANG, N. Robust vehicle stability control based on sideslip angle estimation. In: BARTOSZEWICZ, A. **Robust Control, Theory and Applications**. IntechOpen, 2011. p.561-576.

ERNST, D.; WEHENKEL, L.; GEURTS, P. Tree-based batch mode reinforcement learning. **Journal of Machine Learning Research**. n. 6. 2005. p. 503–556.

FONT-LLAGUNES, J. M. (ed). **Multibody Dynamics**: Computational Methods and Applications. Switzerland: Springer, 2016.

GERMAN ASSOCIATION OF THE AUTOMOTIVE INDUSTRY (VDA). **Vehicle safety**: New Euro NCAP requirements. Available in: <https://www.vda.de/en/topics/safety-and-standards/Vehicle-safety--New-Euro-NCAP-requirements/Vehicle-safety--New-Euro-NCAP-requirements.html>. Access in: 01 Feb. 2018.

GILLESPIE, T. **Fundamentals of Vehicle Dynamics**. Warrendale: SAE Publications, 1992.

GOHARIMANESH, M.; AKBARI, A. A. Improving lateral dynamic of vehicle using direct yaw moment controller by differential brake torques based on quantitative feedback theory. **Scientia Iranica**: Transaction B, Mechanical Engineering. n. 24(2). Scientia Iranica, 2017. p. 662-672.

GUO, J. et al. Integrated control of Active Front Steering and Electronic Stability Program. In: INTERNATIONAL CONFERENCE ON ADVANCED COMPUTER CONTROL, 2., 2010, Shenyang. **Proceedings**… IEEE, 2010. p.449-453.

HANCOCK, P.J. et al. A comparison of braking and differential control of road vehicle yaw-sideslip dynamics. **Proceedings**… Journal of Automobile Engineering, 219 (3), 2005. Institution of Mechanical Engineers, 2005. pp. 309-327.

HAYKIN, S. **Neural Networks and Learning Machines**. 3. ed. Upper Saddle River: Pearson, 2009.

HE, J. **Integrated Vehicle Dynamics Control Using Active Steering, Driveline and Braking**. 2005. 260 p. Thesis (PhD) – School of Mechanical Engineering, University of Leeds, Leeds, 2005.

INAGAKI, S.; KSHIRO, I.; YAMAMOTO, M. Analysis on vehicle stability in critical cornering using phase-plane method. In: INTERNATIONAL SYMPOSIUM ON ADVANCED VEHICLE CONTROL, 1994, Tsukuba-shi. **Proceedings**… Tokyo: SAE Japan, 1994. p. 287-292.

IPG. **Reference Manual**: CarMaker. IPG, 2014.

IPG AUTOMOTIVE. **Driving innovation and creating knowledge**: Our commitment to the future. Available in: <https://ipg-automotive.com/company/research-teaching/>. Access in: 30 Jul. 2018.

JANUSZ, B.; RIEDMILLER, M. Self-learning neural control of a mobile robot. In: IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORKS, v. 5, 1995, Perth, WA. **Proceedings**… IEEE, 1995. p. 2358-2363.

JOLLIFFE, I. T. **Principal Component Analysis**. 2. ed. New York: Springer, 2002.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement Learning: A Survey. **Journal of Artificial Intelligence Research.** 4. AI Access Foundation and Morgan Kaufmann Publishers, 1996.

KIM, P. **MATLAB Deep Learning**: With Machine Learning, Neural Networks and Artificial Intelligence. Seoul: Apress, 2017.

LECUN et al. Efficient BackProp. In: MONTAVON G.; ORR G. B.; MÜLLER K. R. **Neural Networks**: Tricks of the Trade. Lecture Notes in Computer Science. v. 7700. Berlin, Heidelberg: Springer, 2012. p. 9-48.

LEE, M.; HWANG, K.; SUH, I. S. Independent and Integrated Torque Control of 4-Wheel Drive Electric Vehicle for Automated Driving. In: INTERNATIONAL ELECTRIC VEHICLE SYMPOSIUM AND EXHIBITION, 28., 2015, Goyang. **Proceedings**…EVS, 2015.

LEE, T. B. **Artificial intelligence is getting more powerful, and it's about to be everywhere**. 2017. Available in: <https://www.vox.com/new-money/2017/5/18/15655274/google-io-ai-everywhere>. Access in: 08 Mar. 2018.

LI, L.; WU, Z. Study on Torque Vectoring Differential for Vehicle Stability Control via Hardware in-Loop Simulation. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATION SOFTWARE AND NETWORKS, 3., 2011, Xi'an. **Proceedings**… IEEE, 2011. p. 289-293.

LU, Q. et al. Enhancing vehicle cornering limit through sideslip and yaw rate control. **Mechanical Systems and Signal Processing**, v. 75. Elsevier, 2016. p. 455-472.

MATHWORKS. **Choose a Multilayer Neural Network Training Function**. 2018. Available in: < https://de.mathworks.com/help/nnet/ug/choose-a-multilayer-neural-network-training-function.html>. Access in: 15 Apr. 2018.

MCCULLOCH, W. S.; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. **The Bulletin of Mathematical Biophysics**, v. 5, n. 4. Springer Nature, 1943. p.115-133.

MØLLER, M. F. A scaled conjugate gradient algorithm for fast supervised learning. **Neural Networks**, v. 6, n. 4. Elsevier BV, 1993. p.525-533.

OH, S. Y.; LEE, J. H.; CHOI, D. H. A new reinforcement learning vehicle control architecture for vision-based road following. **IEEE Transactions on Vehicular Technology**. IEEE, 2000. p. 997-1005.

PACEJKA, H. B.; BESSELINK, I. J. M. Magic Formula tyre model with transient properties. **Vehicle System Dynamics**, v. 27. Taylor & Francis, 1997. p. 234-249.

PALUSZEK, M.; THOMAS, S. **MATLAB Machine Learning**. Apress, 2016.

PIETQUIN, O.; TANGO, F.; ARAS, R. Batch reinforcement learning for optimizing longitudinal driving assistance strategies. In: IEEE SYMPOSIUM ON COMPUTATIONAL INTELLIGENCE IN VEHICLES AND TRANSPORTATION SYSTEMS, 2011, Paris. **Proceedings**… IEEE, 2011. p. 73-79.

PRICEWATERHOUSECOOPERS (PwC). **Sizing the prize**: What's the real value of AI for your business and how can you capitalise?. 2017. Available in: <https://www.pwc.com/gx/en/issues/data-and-analytics/publications/artificial-intelligence-study.html>. Access in: 30 Jul. 2018.

QINCHAO, Z.; XUNCHENG, W.; FANG, L. The Overview of Active Front Steering System and The Principle of Changeable Transmission Ratio. In: INTERNATIONAL CONFERENCE ON MEASURING TECHNOLOGY AND MECHATRONICS AUTOMATION, 3., 2011, Shanghai. **Proceedings**… IEEE, 2011. p. 894-897.

RIEDMILLER, M. Neural Fitted Q Iteration – First Experiences with a Data Efficient Neural Reinforcement Learning Method. In: GAMA, J. et al. **Machine Learning:** ECML 2005. ECML 2005. Lecture Notes in Computer Science, v. 3720. Berlin, Heidelberg: Springer, 2005. p. 317-328.

RIEDMILLER, M.; BRAUN, H. A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In: IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORKS, 1993, San Francisco. **Proceedings**… IEEE, 1993. p. 586-591.

RIEDMILLER, M; MONTEMERLO, M; DAHLKAMP, H. Learning to Drive a Real Car in 20 Minutes. In: FRONTIERS IN THE CONVERGENCE OF BIOSCIENCE AND INFORMATION TECHNOLOGIES, 2007. **Proceedings**... IEEE, 2007. p. 645 - 650.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review.** 65, n. 6. 1958. p. 386-408.

RUMMERY, G.A.; NIRANJAN, M. **On-Line Q-Learning Using Connectionist Systems**. Cambridge: Department of Engineering, University of Cambridge, 1994.

SMART, W. D.; KAELBLING, L. P. Effective Reinforcement Learning for Mobile Robots. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS & AUTOMATION, 2002, Washington, DC. **Proceedings**… IEEE, 2002. p. 3404-3410.

SUTTON R. S. Reinforcement Learning: Past, Present and Future. In: McKay B. et al. (eds) **Simulated Evolution and Learning.** SEAL 1998. Lecture Notes in Computer Science, v. 1585. Berlin, Heidelberg: Springer, 1999.

SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning**: an Introduction. 1. ed. Cambridge, MA: MIT Press, 1998.

SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning:** an Introduction. 2. ed. Cambridge, MA: MIT Press, 2018.

TANCREDI, T. P. **Otimização multidisciplinar distribuída aplicada a projetos de engenharia**. 2008. 190 p. Thesis (PhD) – Programa de Pós-graduação em Engenharia Naval, Escola Politécnica da Universidade de São Paulo, São Paulo, 2008.

THOMAS, A. J. et al. Two Hidden Layers are Usually Better than One. In: INTERNATIONAL CONFERENCE ON ENGINEERING APPLICATIONS OF NEURAL NETWORKS, 2017, Athens. **Proceedings**… Springer, 2017. p. 279-290.

UN ECE. **Addendum 12-H**: Regulation No. 13-H. 3. ed. 2014.

VAHIDI, A.; ESKANDARIAN, A. A. Research advances in intelligent collision avoidance and adaptive cruise control. In: IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, vol. 4, no. 3, 2003. **Proceedings**… IEEE, 2003. p. 143-153.

VINCENT, I.; SUN, Q. A combined reactive and reinforcement learning controller for an autonomous tracked vehicle. **Robotics and Autonomous Systems**. v. 60(4). Elsevier, 2012. p. 599-608.

WANG, P.; CHAN, C. Y.; de LA FORTELLE, A. A. Reinforcement Learning Based Approach for Automated Lane Change Maneuvers. In: IEEE INTELLIGENT VEHICLES SYMPOSIUM IV. **Proceedings**… IEEE, 2018.

WATKINS, C. J. C. H. **Learning from delayed rewards**. 1989. Thesis (PhD) - King's College, Cambridge, 1989.

WEI, S.; ZOU, Y.; ZHANG, T.; ZHANG, X.; WANG, W. Design and Experimental Validation of a Cooperative Adaptive Cruise Control System Based on Supervised Reinforcement Learning. **Applied Sciences**. vol. 8, n. 7. MDPI, 2018. p. 1014.

WIERING, M.; OTTERLO, M. V. **Reinforcement Learning**: State-of-the-Art. Berlin Heidelberg: Springer, 2012.

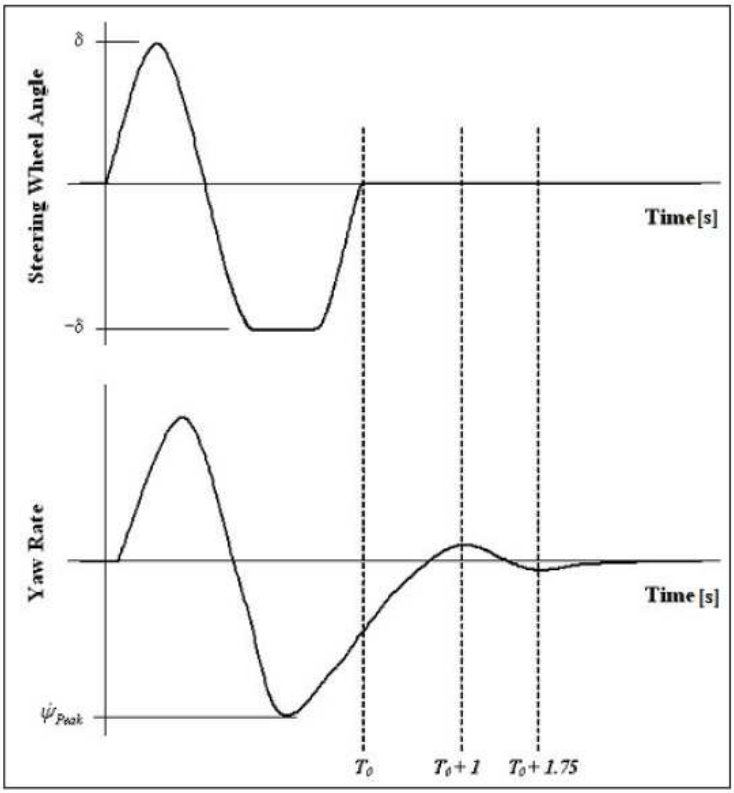
YU, G.; SETHI, I. K. Road-following with continuous learning. In: INTELLIGENT VEHICLES' 95 SYMPOSIUM, 1995, Detroit. **Proceedings**… IEEE, 1995. p. 412-417.

ZHANG, H. Phase Plane Analysis for Vehicle Handling and Stability. **International Journal of Computational Intelligence Systems**. vol. 4. n. 6. 2011. p. 1179-1186.

ANNEX A – Criteria to approve ESC in passenger cars according to UN ECE R13H (UN ECE, 2014)

3.1. The yaw rate measured 1 second after completion of the Sine with Dwell steering input (time T0 + 1 in Figure A.1) shall not exceed 35 per cent of the first peak value of yaw rate recorded after the steering wheel angle changes sign (between first and second peaks) (ψ_{Peak} in Figure A.1) during the same test run.

Figure A.1 - Steering wheel position and yaw velocity information used to assess lateral stability.



Source: UN ECE (2014, p. 74).

3.2. The yaw rate measured 1.75 seconds after completion of the Sine with Dwell steering input shall not exceed 20 per cent of the first

peak value of yaw rate recorded after the steering wheel angle changes sign (between first and second peaks) during the same test run.

3.3. The lateral displacement of the vehicle centre of gravity with respect to its initial straight path shall be at least 1.83 m for vehicles with a GVM of 3,500 kg or less, and 1.52 m for vehicles with a maximum mass greater than 3,500 kg when computed 1.07 seconds after the Beginning of Steer (BOS). BOS is defined in paragraph 5.11.6 (in UN ECE R13H).